Oracle Workflow™ Guide

RELEASE 2.5

January 2000

ORACLE®

Oracle Workflow™ Guide Release 2.5

The part number for this book is A75397–01.

Copyright © 1996, 2000, Oracle Corporation. All rights reserved.

Primary Authors: Siu Chang, Clara Jaeckel

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

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent and other intellectual and industrial property law. Reverse engineering, disassembly or decompilation of the Programs is prohibited.

Program Documentation is licensed for use solely to support the deployment of the Programs and not for any other purpose.

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. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the US Government or anyone licensing or using the Programs on behalf of the US Government, the following notice 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 including documentation, 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 52.227–19, Commercial Computer Software – Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065."

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be 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.

Oracle is a registered trademark and ConText, Enabling the Information Age, Oracle7, Oracle8, Oracle8*i*, Oracle Access, Oracle Application Object Library, Oracle Financials, Oracle Discoverer, Oracle Web Customers, Oracle Web Employees, Oracle Workflow, Oracle Work in Process, PL/SQL, Pro*C, SmartClient, SQL*, SQL*Forms, SQL*Loader, SQL*Menu, SQL*Net, SQL*Plus, and SQL*Report are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Preface ..... **xi**
Audience for This Guide xii
How To Use This Guide xii
Finding Out What's New xiii
Other Information Sources xiv
Online Documentation xiv
Related User Guides xv
User Guides Related to All Products xv
User Guides Related to This Product xvi
Installation and System Administration Guides xvii
Training and Support xix
Do Not Use Database Tools to Modify Oracle Workflow Data xx
About Oracle xx
Your Feedback xxi

Chapter 1 **Overview of Oracle Workflow ..... 1 – 1**
Introduction to Oracle Workflow ..... 1 – 2
Major Features and Definitions ..... 1 – 3
Workflow Processes ..... 1 – 5

Chapter 2 **Setting Up Oracle Workflow ..... 2 – 1**
Oracle Workflow Hardware and Software Requirements ..... 2 – 2
Overview of Setting Up ..... 2 – 4

Overview of Required Set Up Steps for the Standalone Version of Oracle Workflow . . . . . 2 – 4
Overview of Required Set Up Steps for the Version of Oracle Workflow Embedded in Oracle Applications . . . . . 2 – 5
Optional Set Up Steps . . . . . 2 – 6
Other Workflow Features . . . . . 2 – 6
Identifying the Version of Your Oracle Workflow Server . . . 2 – 6
Setup Flowchart . . . . . 2 – 8
Setup Checklist . . . . . 2 – 9
Setup Steps . . . . . 2 – 10
Overview of Oracle Workflow Access Protection . . . . . 2 – 70
Setting Up a Default Access Level . . . . . 2 – 74
Using the Workflow Definitions Loader . . . . . 2 – 76

Chapter 3 **Defining a Workflow Process . . . . . 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 – 7
Opening and Saving Item Types . . . . . 3 – 12
Quick Start Wizard Overview . . . . . 3 – 18
Item Type Definition Web Page . . . . . 3 – 21

Chapter 4 **Defining Workflow Process Components . . . . . 4 – 1**
Workflow Process Components . . . . . 4 – 2
Item Types . . . . . 4 – 2
Allowing Access to an Object . . . . . 4 – 16
Lookup Types . . . . . 4 – 18
Messages . . . . . 4 – 22
Activities . . . . . 4 – 37
Voting Activity . . . . . 4 – 50
Deleting Objects in Oracle Workflow Builder . . . . . 4 – 55

Chapter 5 **Defining a Workflow Process Diagram . . . . . 5 – 1**
Process Window . . . . . 5 – 2
Modifying Fonts in Oracle Workflow Builder . . . . . 5 – 16
Creating a Shortcut Icon for a Workflow Process . . . . . 5 – 17
Roles . . . . . 5 – 19

Chapter 6 **Predefined Workflow Activities . . . . . 6 – 1**
Standard Activities . . . . . 6 – 2
And/Or Activities . . . . . 6 – 2
Comparison Activities . . . . . 6 – 3
Compare Execution Time Activity . . . . . 6 – 3
Wait Activity . . . . . 6 – 4
Block Activity . . . . . 6 – 5
Defer Thread Activity . . . . . 6 – 6
Launch Process Activity . . . . . 6 – 6
Noop Activity . . . . . 6 – 7
Loop Counter Activity . . . . . 6 – 7
Start Activity . . . . . 6 – 9
End Activity . . . . . 6 – 9
Role Resolution Activity . . . . . 6 – 9
Notify Activity . . . . . 6 – 10
Vote Yes/No Activity . . . . . 6 – 11
Master/Detail Coordination Activities . . . . . 6 – 12
Wait for Flow Activity . . . . . 6 – 12
Continue Flow Activity . . . . . 6 – 13
Assign Activity . . . . . 6 – 14
Get Monitor URL Activity . . . . . 6 – 15
Concurrent Manager Standard Activities . . . . . 6 – 16
Execute Concurrent Program Activity . . . . . 6 – 16
Submit Concurrent Program Activity . . . . . 6 – 17
Wait for Concurrent Program Activity . . . . . 6 – 18
Default Error Process . . . . . 6 – 20
System: Error Item Type and Item Attributes . . . . . 6 – 21
Default Error Process . . . . . 6 – 22
Retry–only Process . . . . . 6 – 25

Chapter 7 **Defining PL/SQL Procedures for Oracle Workflow . . . . . 7 – 1**
Standard API for PL/SQL Procedures Called by Function Activities . . . . . 7 – 2
Standard API for an Item Type Selector or Callback Function . . 7 – 7
Standard API for a ”PL/SQL” Document . . . . . 7 – 11

Chapter 8 **Oracle Workflow APIs . . . . . 8 – 1**
Oracle Workflow Procedures and Functions . . . . . 8 – 2
Overview of the Workflow Engine . . . . . 8 – 3
Oracle Workflow Java Interface . . . . . 8 – 4

Additional Workflow Engine Features . . . . . 8 – 5
Workflow Engine APIs . . . . . 8 – 14
CreateProcess . . . . . 8 – 16
SetItemUserKey . . . . . 8 – 18
GetItemUserKey . . . . . 8 – 19
GetActivityLabel . . . . . 8 – 20
SetItemOwner . . . . . 8 – 21
StartProcess . . . . . 8 – 22
LaunchProcess . . . . . 8 – 24
SuspendProcess . . . . . 8 – 26
ResumeProcess . . . . . 8 – 27
AbortProcess . . . . . 8 – 28
CreateForkProcess . . . . . 8 – 30
StartForkProcess . . . . . 8 – 32
Background . . . . . 8 – 33
AddItemAttr . . . . . 8 – 35
SetItemAttribute . . . . . 8 – 36
SetItemAttrDocument . . . . . 8 – 38
getItemTypes . . . . . 8 – 40
GetItemAttribute . . . . . 8 – 41
GetItemAttrDocument . . . . . 8 – 42
getItemAttributes . . . . . 8 – 43
GetItemAttrInfo . . . . . 8 – 44
GetActivityAttrInfo . . . . . 8 – 45
GetActivityAttribute . . . . . 8 – 46
BeginActivity . . . . . 8 – 47
CompleteActivity . . . . . 8 – 49
AssignActivity . . . . . 8 – 52
HandleError . . . . . 8 – 53
SetItemParent . . . . . 8 – 55
ItemStatus . . . . . 8 – 56
getProcessStatus . . . . . 8 – 57
Workflow Core APIs . . . . . 8 – 58
CLEAR . . . . . 8 – 59
GET_ERROR . . . . . 8 – 60
TOKEN . . . . . 8 – 61
RAISE . . . . . 8 – 62
CONTEXT . . . . . 8 – 65
TRANSLATE . . . . . 8 – 67
Workflow Purge APIs . . . . . 8 – 68
Items . . . . . 8 – 70
Activities . . . . . 8 – 71

Notifications . . . . . 8 – 72
Total . . . . . 8 – 73
TotalPERM . . . . . 8 – 74
AdHocDirectory . . . . . 8 – 75
Purge Obsolete Workflow Runtime Data Concurrent Program . . . . . 8 – 76
Workflow Directory Service APIs . . . . . 8 – 77
GetRoleUsers . . . . . 8 – 78
GetUserRoles . . . . . 8 – 79
GetRoleInfo . . . . . 8 – 80
GetRoleInfo2 . . . . . 8 – 81
IsPerformer . . . . . 8 – 82
CurrentUser . . . . . 8 – 83
UserActive . . . . . 8 – 84
GetUserName . . . . . 8 – 85
GetRoleName . . . . . 8 – 86
GetRoleDisplayName . . . . . 8 – 87
SetAdHocUserStatus . . . . . 8 – 88
SetAdHocRoleStatus . . . . . 8 – 89
CreateAdHocUser . . . . . 8 – 90
CreateAdHocRole . . . . . 8 – 92
AddUsersToAdHocRole . . . . . 8 – 94
SetAdHocUserExpiration . . . . . 8 – 95
SetAdHocRoleExpiration . . . . . 8 – 96
SetAdHocUserAttr . . . . . 8 – 97
SetAdHocRoleAttr . . . . . 8 – 98
RemoveUsersFromAdHocRole . . . . . 8 – 99
Workflow Preferences API . . . . . 8 – 100
get_pref . . . . . 8 – 100
Workflow Monitor APIs . . . . . 8 – 101
GetDiagramURL . . . . . 8 – 102
GetEnvelopeURL . . . . . 8 – 104
GetAdvancedEnvelopeURL . . . . . 8 – 106
Oracle Workflow Views . . . . . 8 – 108
WF_ITEM_ACTIVITY_STATUSES_V . . . . . 8 – 108
WF_NOTIFICATION_ATTR_RESP_V . . . . . 8 – 110
WF_RUNNABLE_PROCESSES_V . . . . . 8 – 111
WF_ITEMS_V . . . . . 8 – 112
Workflow Queue APIs . . . . . 8 – 113
EnqueueInbound . . . . . 8 – 116
DequeueOutbound . . . . . 8 – 117
DequeueEventDetail . . . . . 8 – 120

PurgeEvent . . . . . 8 – 122
PurgeItemType . . . . . 8 – 123
ProcessInboundQueue . . . . . 8 – 124
GetMessageHandle . . . . . 8 – 125
DeferredQueue . . . . . 8 – 126
InboundQueue . . . . . 8 – 127
OutboundQueue . . . . . 8 – 128
ClearMsgStack . . . . . 8 – 129
CreateMsg . . . . . 8 – 130
WriteMsg . . . . . 8 – 131
SetMsgAttr . . . . . 8 – 132
SetMsgResult . . . . . 8 – 133
Document Management APIs . . . . . 8 – 134
get_launch_document_url . . . . . 8 – 135
get_launch_attach_url . . . . . 8 – 136
get_open_dm_display_window . . . . . 8 – 137
get_open_dm_attach_window . . . . . 8 – 138
set_document_id_html . . . . . 8 – 139
Overview of the Oracle Workflow Notification System . . . . . 8 – 141
Notification Model . . . . . 8 – 141
Notification APIs . . . . . 8 – 146
Send . . . . . 8 – 148
Custom Callback Function . . . . . 8 – 149
SendGroup . . . . . 8 – 152
Forward . . . . . 8 – 154
Transfer . . . . . 8 – 155
Cancel . . . . . 8 – 156
CancelGroup . . . . . 8 – 157
Respond . . . . . 8 – 158
Responder . . . . . 8 – 159
VoteCount . . . . . 8 – 160
OpenNotificationsExist . . . . . 8 – 161
Close . . . . . 8 – 162
AddAttr . . . . . 8 – 163
SetAttribute . . . . . 8 – 164
GetAttrInfo . . . . . 8 – 166
GetInfo . . . . . 8 – 167
GetText . . . . . 8 – 168
GetShortText . . . . . 8 – 169
GetAttribute . . . . . 8 – 170
GetAttrDoc . . . . . 8 – 171
GetSubject . . . . . 8 – 172

GetBody . . . . . 8 – 173
GetShortBody . . . . . 8 – 174
TestContext . . . . . 8 – 175
AccessCheck . . . . . 8 – 176
WorkCount . . . . . 8 – 177
GetNotifications . . . . . 8 – 178
GetNotificationAttributes . . . . . 8 – 179

Chapter 9 **Oracle Workflow Home Page . . . . . 9 – 1**
Accessing the Oracle Workflow Home Page . . . . . 9 – 2
Setting User Preferences . . . . . 9 – 4

Chapter 10 **Viewing Notifications and Processing Responses . . . . . 10 – 1**
Overview of Notification Handling . . . . . 10 – 2
Reviewing Notifications via Electronic Mail . . . . . 10 – 2
Viewing Notifications from a Web Browser . . . . . 10 – 13
Reviewing a Summary of Your Notifications via Electronic Mail . . . . . 10 – 23
Defining Rules for Automatic Notification Processing . . . . . 10 – 24
Document Management Integration in Notifications . . . . . 10 – 31

Chapter 11 **Monitoring Workflow Processes . . . . . 11 – 1**
Overview of Workflow Monitoring . . . . . 11 – 2
Workflow Monitor . . . . . 11 – 2
Workflow Monitor Access . . . . . 11 – 7

Chapter 12 **Testing a Workflow Definition . . . . . 12 – 1**
Testing Workflow Definitions . . . . . 12 – 2

Chapter 13 **Demonstration Workflow Processes . . . . . 13 – 1**
Sample Workflow Processes . . . . . 13 – 2
Displaying the Process Diagram of a Sample Workflow . . . . 13 – 3
Requisition Process . . . . . 13 – 4
Installing the Requisition Data Model . . . . . 13 – 5
Initiating the Requisition Workflow . . . . . 13 – 7
The Requisition Item Type . . . . . 13 – 11
Summary of the Requisition Approval Process . . . . . 13 – 12
Requisition Process Activities . . . . . 13 – 14

Summary of the Notify Approver Subprocess . . . . . 13 – 19
Notify Approver Subprocess Activities . . . . . 13 – 20
Sample StartProcess Function . . . . . 13 – 22
Example Function Activities . . . . . 13 – 25
Example: Select Approver . . . . . 13 – 25
Example: Verify Authority . . . . . 13 – 28
Example Notification Activity . . . . . 13 – 30
Example: Notify Requisition Approval Required . . . . . 13 – 31
Product Survey Process . . . . . 13 – 33
Installing the Product Survey Data Model . . . . . 13 – 34
Initiating the Product Survey Workflow . . . . . 13 – 35
The Product Survey Item Type . . . . . 13 – 37
Summary of the Survey – Single Process . . . . . 13 – 38
Survey – Single Process Activities . . . . . 13 – 40
Summary of the Survey – Master/Detail Process . . . . . 13 – 41
Survey – Master/Detail Process Activities . . . . . 13 – 43
Summary of the Detail Survey Process . . . . . 13 – 45
Detail Survey Process Activities . . . . . 13 – 46
Document Review Process . . . . . 13 – 48
The Document Management Item Type . . . . . 13 – 48
Summary of the Document Review Process . . . . . 13 – 49
Document Review Process Activities . . . . . 13 – 51
Error Check Process . . . . . 13 – 53
The Periodic Alert Item Type . . . . . 13 – 53
Summary of the Error Check Process . . . . . 13 – 55
Error Check Process Activities . . . . . 13 – 56
Summary of the User Defined Alert Action Process . . . . . 13 – 59
User Defined Alert Action Process Activities . . . . . 13 – 60

Chapter 14 **Workflow Administration Scripts . . . . . 14 – 1**
Miscellaneous SQL Scripts . . . . . 14 – 2
FNDWFPR . . . . . 14 – 4
WFNLADD.sql . . . . . 14 – 4
Wfbkg.sql . . . . . 14 – 4
Wfbkgchk.sql . . . . . 14 – 5
Wfchact.sql . . . . . 14 – 5
Wfchacta.sql . . . . . 14 – 6
Wfchita.sql . . . . . 14 – 6
Wfchitt.sql . . . . . 14 – 6
Wfchluc.sql . . . . . 14 – 7
Wfchlut.sql . . . . . 14 – 7

Wfchmsg.sql . . . . . 14 – 7
Wfchmsga.sql . . . . . 14 – 8
Wfdirchk.sql . . . . . 14 – 8
Wfnlena.sql . . . . . 14 – 9
Wfntfsh.sql . . . . . 14 – 9
Wfprot.sql . . . . . 14 – 9
Wfqclean.sql . . . . . 14 – 10
Wfrefchk.sql . . . . . 14 – 10
Wfretry.sql . . . . . 14 – 11
Wfrmall.sql . . . . . 14 – 11
Wfrmita.sql . . . . . 14 – 11
Wfrmitms.sql . . . . . 14 – 12
Wfrmitt.sql . . . . . 14 – 12
Wfrmtype.sql . . . . . 14 – 12
Wfrun.sql . . . . . 14 – 13
Wfstat.sql . . . . . 14 – 13
Wfstatus.sql . . . . . 14 – 13
Wfstdchk.sql . . . . . 14 – 13
Wfver.sql . . . . . 14 – 14
Wfverchk.sql . . . . . 14 – 14
Wfverupd.sql . . . . . 14 – 14

Appendix A **Oracle Workflow Builder Menus and Toolbars . . . . . A – 1**
Oracle Workflow Builder Menus . . . . . A – 2
Oracle Workflow Builder Toolbars . . . . . A – 7

Appendix B **Oracle Applications Embedded Workflows . . . . . B – 1**
Predefined Workflows Embedded in Oracle Applications and Oracle Self–Service Web Applications . . . . . B – 2
Oracle Support Policy for Predefined Workflows . . . . . B – 11
Customization Guidelines . . . . . B – 11
Resolving Customization Issues . . . . . B – 11
What Is NOT Supported . . . . . B – 12
What Is Supported . . . . . B – 12

Glossary

Index

Preface

Audience for This Guide

Welcome to the *Oracle Workflow Guide*.

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

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

  If you have never used Oracle Workflow, we suggest you attend one or more of the Oracle Workflow training classes available through Oracle University.

See Other Information Sources for more information about Oracle Applications product information.

The *Oracle Workflow Guide* also assumes you have a basic understanding of operating system concepts and familiarity with Oracle8 or Oracle8*i*, PL/SQL, and Oracle WebDB or Oracle Web Application Server/Oracle Application Server technology. If you have not yet been introduced to any of these systems, we suggest you attend one or more of the training classes available through Oracle University.

How To Use This Guide

This guide contains the information you need to understand and use Oracle Workflow to automate business processes to fit your needs.

This preface explains how this guide is organized and introduces other sources of information that can help you. This guide contains 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 begin defining a workflow process.
- Chapter 4 describes how to define the components necessary to build a workflow process.
- Chapter 5 describes how to draw and define a workflow process diagram.
- Chapter 6 describes the standard activities provided with Oracle Workflow.

- Chapter 7 describes the standard APIs for the PL/SQL functions that can be called by Oracle Workflow.
- Chapter 8 provides detailed information about Oracle Workflow's APIs.
- Chapter 9 describes the Oracle Workflow home page, where users and administrators can centrally access all the web–based features of Oracle Workflow.
- Chapter 10 discusses how a user can view and act on a workflow notification.
- Chapter 11 describes how to use the Workflow Monitor to administer or view the status of a workflow process.
- Chapter 12 describes how to launch a workflow process for testing purposes.
- Chapter 13 describes the demonstration workflow processes included with Oracle Workflow.
- Chapter 14 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 appendix also includes the Oracle Workflow support policy.

At the end of this guide, we include a glossary of Oracle Workflow terms.

Finding Out What's New

If you are using the version of Oracle Workflow embedded in Oracle Applications, choose the section that describes new features or what's new from the expandable menu in the HTML help window for Oracle Workflow. If you are using the standalone version of Oracle Workflow, choose the section that describes new features from the contents page of the HTML help.

This section describes:

- New features in this release. This information is updated for each new release of Oracle Workflow.

- Information about any features that were not yet available when this guide was printed. For example, if your system administrator has installed software from a mini pack as an upgrade, this document describes the new features.

Other Information Sources

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

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

Online Documentation

If you are using the version of Oracle Workflow embedded in Oracle Applications, note that all Oracle Applications documentation is available online (HTML and PDF). The technical reference manuals are available in paper format only. The HTML documentation is translated into over twenty languages.

The HTML version of this guide is optimized for onscreen reading, and you can use it to follow hypertext links for easy access to other HTML guides in the library. When you have an HTML window open, you can use the features on the left side of the window to navigate freely throughout all Oracle Applications documentation.

- You can use the Search feature to search by words or phrases.
- You can use the expandable menu to search for topics in the menu structure we provide. The Library option on the menu expands to show all Oracle Applications HTML documentation.

You can view HTML help in the following ways:

- From an application window, use the help icon or the help menu to open a new Web browser and display help about that window.
- Use the documentation CD.
- Use a URL provided by your system administrator.

Your HTML help may contain information that was not available when this guide was printed.

This guide is also available online in Windows Help format. The Windows Help version 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.

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.

Related User Guides

Oracle Workflow is used by other Oracle Applications products to provide embedded workflows. Therefore, if you are using the version of Oracle Workflow embedded in Oracle Applications, you may want to refer to other user guides to learn more about the embedded workflows.

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

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

User Guides Related to All Products

Oracle Applications User's Guide

This guide explains how to navigate the system, enter data, and query information, and introduces other basic features of the GUI available with this release of the Oracle Applications–embedded version of Oracle Workflow (and any other Oracle Applications product).

You can also access this guide online by choosing "Getting Started and Using Oracle Applications" from the Oracle Applications help system.

Oracle Applications Implementation Wizard User 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 Oracle Developer forms so that they integrate with Oracle Applications.

Oracle Applications User Interface Standards

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

User Guides Related to This Product

Oracle General Ledger User Guide

This guide provides information about journal entry, budgeting, and multi–company accounting and consolidation.

Oracle Purchasing User Guide

This guide provides information about entering and managing purchase orders and requisitions.

Implementing Oracle Self–Service Human Resources (SSHR)

This guide provides information about setting up the self–service human resources management functions for managers and employees. Managers and employees can then use an intranet and Web browser to have easy and intuitive access to personal and career management functionality

Oracle Payables User Guide

This guide provides information about entering and managing suppliers, invoices, and payments.

Oracle Projects User Guide

This guide provides information about entering and managing projects, budgets, expenditures, costing, and billing.

Oracle Receivables User Guide

This guide provides information about entering and managing customers, receipts, collections, and transactions.

Oracle Business Intelligence System Implementation Guide

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

BIS 11*i* User Guide Online Help

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

Oracle Financials Open Interfaces Guide

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

Oracle Applications Flexfields Guide

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

Installation and System Administration Guides

Oracle Applications Concepts

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

Installing Oracle Applications

This guide provides instructions for managing the installation of Oracle Applications products. In Release 11*i*, much of the installation process is handled using Oracle One–Hour Install, which minimizes the time it takes to install Oracle Applications and the Oracle 8*i* Server technology stack by automating many of the required steps. This guide contains instructions for using Oracle One–Hour Install and lists the tasks you need to perform to finish your installation. You should use this guide in conjunction with individual product user guides and implementation guides.

Upgrading Oracle Applications

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

Using the AD Utilities

Use this guide to help you run the various AD utilities, such as AutoInstall, AutoPatch, AD Administration, AD Controller, Relink, and others. It contains how–to steps, screenshots, and other information that you need to run the AD utilities.

Oracle Applications Product Update Notes

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

Oracle Applications System Administrator's Guide

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

Oracle Application Object Library/Workflow Technical Reference Manual

This reference manual contains database diagrams and a detailed description of database tables, forms, reports, and programs for Oracle Application Object Library, Oracle Workflow, and related applications. This information helps you convert data from your existing applications, integrate Oracle Workflow with non–Oracle applications, and write custom reports for Oracle Workflow.

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

Training and Support

Training

We offer 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. We also offer Net classes, where training is delivered over the Internet, and many multimedia–based courses on CD. 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.

Do Not Use Database Tools to Modify Oracle Workflow Data

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 guides.

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

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. Oracle Applications provides the E–business Suite, a fully integrated suite of more than 70 software modules for financial management, Internet procurement, business intelligence, 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, enabling 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 application products, along with related consulting, education and support services, in over 145 countries around the world.

Your Feedback

Thank you for using Oracle Workflow and this guide.

We value your comments and feedback. This guide contains a Reader's Comment Form which you can use to explain what you like or dislike about Oracle Workflow or this 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**.

CHAPTER

1

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 take action. 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 Oracle8 Server, to express any business rule that affects a workflow process. See: Workflow Processes: page 1 – 5.

Delivering Electronic Notifications

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

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 – 5.

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 Oracle8 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 or a Java API. Based on flexibly–defined workflow rules, the engine determines which activities are eligible to run, and then runs them. The Workflow Engine supports sophisticated workflow rules, including looping, branching, parallel flows, and subflows.

Workflow Definitions Loader

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

Complete Programmatic Extensibility

Oracle Workflow lets you include your own PL/SQL procedures or external functions as activities in your workflows. Without modifying your application code, you can have your own program run whenever

the Workflow Engine detects that your program's prerequisites are satisfied.

Electronic Notifications

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

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

Electronic Mail Integration

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

Internet–Enabled Workflow

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

Monitoring and Administration

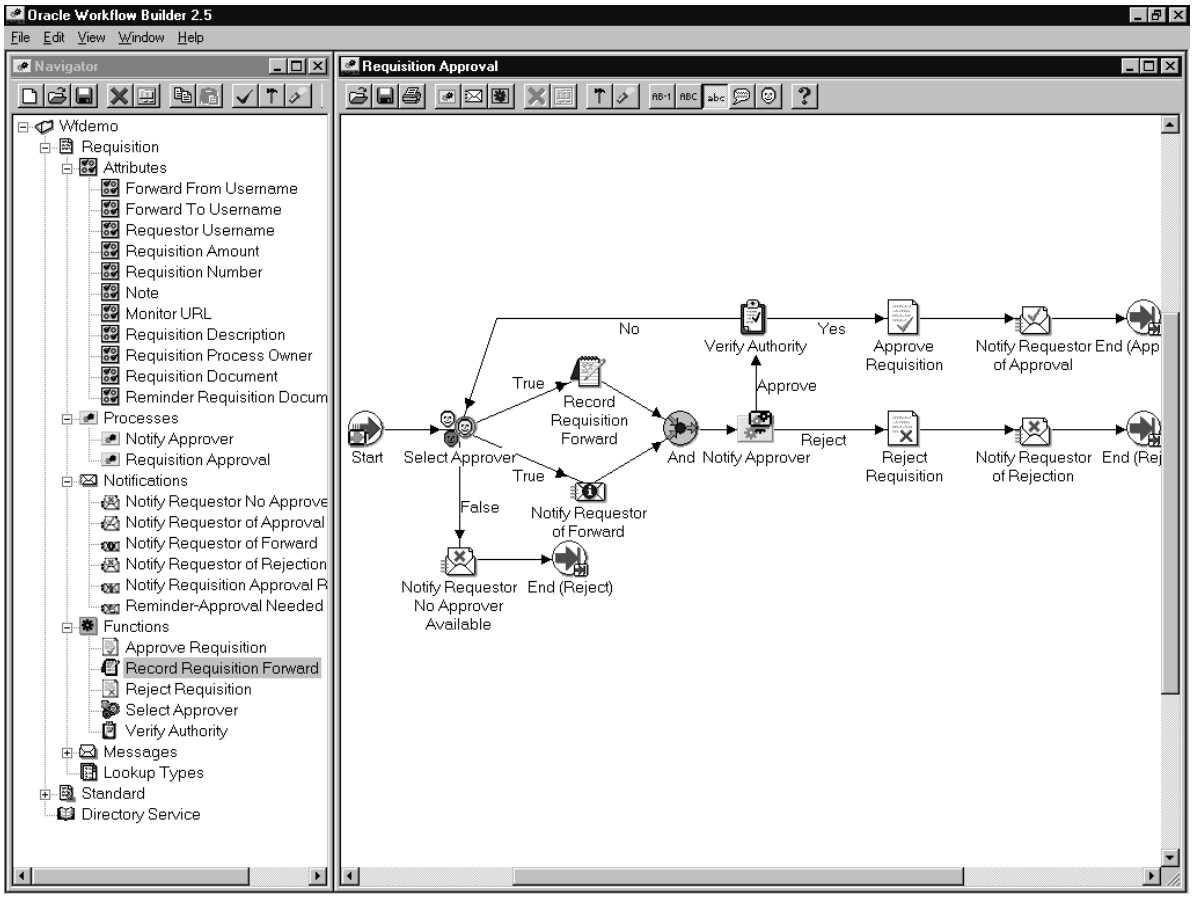
Workflow administrators and users can view the progress of a work item in a workflow process by connecting to the Workflow Monitor using a standard Web browser that supports Java. The Workflow Monitor displays an annotated view of the process diagram for a particular instance of a workflow process, so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

Workflow Processes

Oracle Workflow manages business processes according to rules that you define. The rules, which we call a workflow process definition, include the activities that occur in the process and the relationship between those activities. An activity in a process definition can be an automated function defined by a PL/SQL stored procedure or an external function, a notification to a user or role that may optionally request a response, 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 Oracle Net8 (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, Windows 98, or Windows NT
  - At least 17.3 MB of available disk space to install Oracle Installer, Oracle Workflow Builder, and Oracle Net8.

Attention: Oracle Net8 requires and only supports the use of Microsoft's TCP/IP drivers.

- The Workflow Server requires Oracle WebDB 2.0 or higher, Oracle Web Application Server 3.0.1 or higher, or Oracle Application Server 4.0.7 or higher to be previously installed.
- The E–mail notifications component contains a program that can send mail through Oracle Internet Messaging 4.2, 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 Internet Messaging 4.2 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 WebDB or Oracle Application Server 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 or higher and Abstract Windowing Toolkit (AWT).

Overview of Setting Up

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.

Overview of Required Set Up Steps for the Standalone Version of Oracle Workflow

1. Set up the default Oracle Workflow user preferences for your entire enterprise using the Global Preferences web page. The Global Preferences web page also lets you define your workflow administrator role and your Workflow web agent. See: Setting Global User Preferences: page 2 – 12.
2. 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 4individual 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 – 17.
3. Create a view called WF_LANGUAGES that identifies the languages defined in your Oracle8 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 – 25.
4. Define an environment variable called http_proxy if you plan to use the Notification Mailer. See:Setting the http_proxy Environment Variable: page 2 – 29.
5. Define an environment variable called WF_RESOURCES if your Workflow server is installed on a UNIX platform. See: Setting the WF_RESOURCES Environment Variable: page 2 – 30.
6. If you wish to integrate a certified document management system with Oracle Workflow, define a node for the document management system in the Document Nodes web page. See: Defining Document Management Repositories: page 2 – 31.
7. 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 – 34.

Overview of Required Set Up Steps for the Version of Oracle Workflow Embedded in Oracle Applications

1. Set up the default Oracle Workflow user preferences for your entire enterprise using the Global Preferences web page. The Global Preferences web page also lets you define your workflow administrator role and your Workflow web agent. See: Setting Global User Preferences: page 2 – 12.
2. Set the system profile option called Socket Listener Activated to Yes. See: Setting the Socket Listener Activated Profile Option: page 2 – 28.
3. Define an environment variable called http_proxy if you plan to use the Notification Mailer. See: Setting the http_proxy Environment Variable: page 2 – 29.
4. If you wish to integrate a certified document management system with Oracle Workflow, define a node for the document management system in the Document Nodes web page. See: Defining Document Management Repositories: page 2 – 31.
5. 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 – 34.

Attention: Although your Oracle Workflow installation automatically sets up the following for you, you may want to refer to their appropriate sections for additional background information:

- Directory services: page 2 – 17
- WF_LANGUAGES view: page 2 – 25
- Path to the language–dependent resources file: page 2 – 30

Optional Set Up Steps

1. If you are using the version of Oracle Workflow embedded in Oracle Applications, you can partition the WF_ITEM_ACTIVITY_STATUSES, WF_ITEM_ACTIVITY_STATUSES_H, WF_ITEM_ATTRIBUTE_VALUES, and WF_ITEMS tables for performance gain. See: Partitioning Workflow Tables: page 2 – 10.
2. Set up the Notification Mailer program if you want to allow your users to receive notifications by E–mail. See: Implementing the Notification Mailer: page 2 – 38.
3. Customize the company logo that appears in Oracle Workflow's web pages. See: Customizing the Logo on Oracle Workflow's Web Pages: page 2 – 68.
4. You can modify the templates for your electronic mail notifications. See: Modifying Your Message Templates: page 2 – 56.
5. You can include additional icons to your Oracle Workflow Icons subdirectory to customize the diagrammatic representation of your workflow processes. Use custom symbols for each activity you define. See: Adding Custom Icons to Oracle Workflow: page 2 – 69.

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 – 70.

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 – 76.

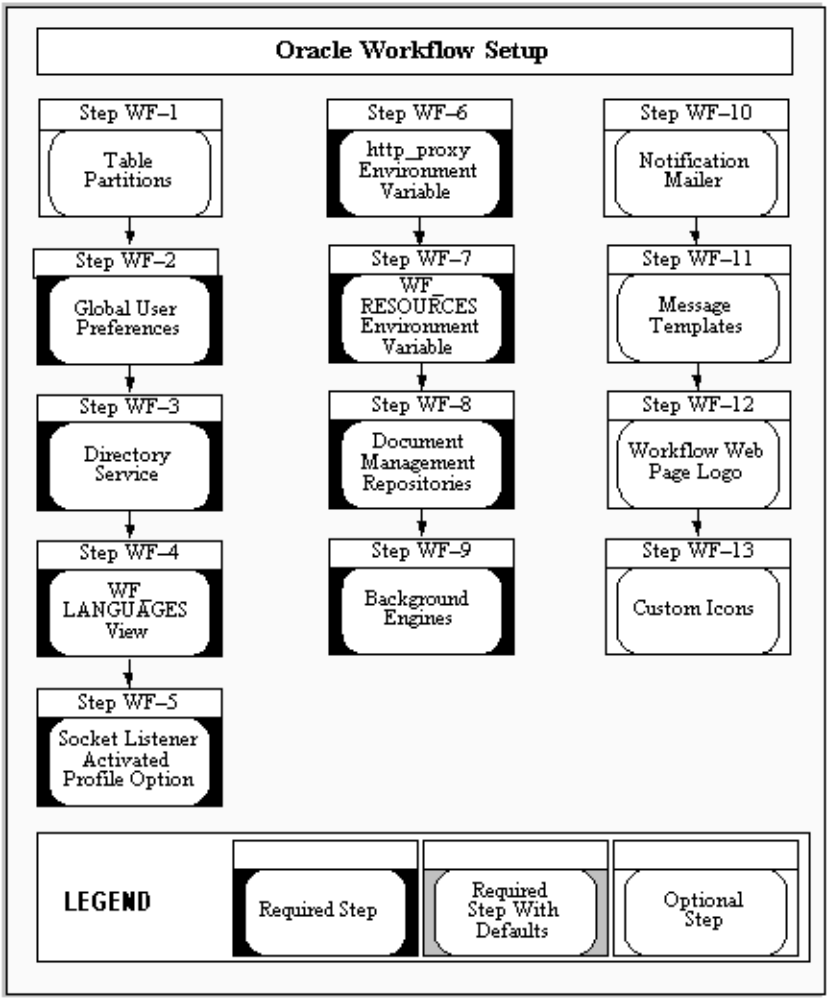
Identifying the Version of Your Oracle Workflow Server

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 SQLPLUS and run a script called wfver.sql. See: wfver.sql: page 14 – 14.

In addition, all Oracle Workflow modules, such as the Workflow Definitions Loader, Oracle Workflow Builder, Notification Mailer, and the Workflow Monitor, automatically verify that the module is compatible with the version of the Oracle Workflow server that it is operating against. This version compatibility check helps to prevent problems such as running Oracle Workflow Builder 2.5 against an Oracle Workflow 2.0.3 database.

Setup Flowchart

Some of the steps outlined in this flowchart and setup checklist are **Required** and some are Optional. You need to perform **Optional** steps only if you plan to use the related feature or complete certain business functions.



Setup Checklist

The following table lists Oracle Workflow setup steps. A reference to whether the step is pertinent to the standalone or embedded version of Oracle Workflow or both and whether the step is optional or required is provided.

	Step No.	Required	Step	Standalone/ Embedded/ Both
	Step 1	Optional	Partitioning Workflow Tables: page 2 – 10	Embedded
	Step 2	Required	Setting Global User Preferences: page 2 – 12	Both
	Step 3	Required	Setting Up an Oracle Workflow Directory Service: page 2 – 17	Standalone
	Step 4	Required	Creating the WF_LANGUAGES View: page 2 – 25	Standalone
	Step 5	Required	Setting the Socket Listener Activated Profile Option: page 2 – 28	Embedded
	Step 6	Required	Setting the http_proxy Environment Variable: page 2 – 29	Both
	Step 7	Required	Setting the WF_RESOURCES Environment Variable: page 2 – 30	Embedded
	Step 8	Required	Defining Document Management Repositories: page 2 – 31	Both
	Step 9	Required	Setting Up Background Workflow Engines: page 2 – 34	Both
	Step 10	Optional	Implementing the Notification Mailer: page 2 – 38	Both
	Step 11	Optional	Modifying Your Message Templates: page 2 – 56	Both
	Step 12	Optional	Customizing the Logo on Oracle Workflow's Web Pages: page 2 – 68	Both
	Step 13	Optional	Adding Custom Icons to Oracle Workflow: page 2 – 69	Both

Setup Steps

Step 1 Partitioning Workflow Tables

If you are using the version of Oracle Workflow embedded in Oracle Applications, you can optionally run a script called wfupartb.sql to partition certain Workflow tables. This step is highly recommended for performance gain.

The script partitions the following tables and recreates the associated indexes:

Table	Indexes
WF_ITEM_ACTIVITY_STATUSES	WF_ITEM_ACTIVITY_STATUSES_PK
	WF_ITEM_ACTIVITY_STATUSES_N1
	WF_ITEM_ACTIVITY_STATUSES_N2
WF_ITEM_ACTIVITY_STATUSES_H	WF_ITEM_ACTIVITY_STATUSES_H_N1
	WF_ITEM_ACTIVITY_STATUSES_H_N2
WF_ITEM_ATTRIBUTE_VALUES	WF_ITEM_ATTRIBUTE_VALUES_PK
WF_ITEMS	WF_ITEMS_PK
	WF_ITEMS_N1
	WF_ITEMS_N2
	WF_ITEMS_N3

Table 2 – 1 (Page 1 of 1)

Before running wfupartb.sql, you should back up these four tables so that you can restore them in case the script fails.

To run the script, you must have sufficient free space on the table and index tablespaces. During the creation of the partitioned tables, the script requires slightly more diskspace than the underlying tables, in the same tablespace where the underlying tables are located. Similarly, sufficient free space is required for the index tablespace.

Additionally, you should allow sufficient time for the script to run. The amount of time needed depends on the amount of data in the tables. When the tables already contain existing data, such as after an upgrade from a previous release, the script requires more time than it does when the tables are empty, such as after a fresh installation of Oracle Workflow. To minimize the time required, run the script as early as possible in your setup process.

Attention: If you are running wfupartb.sql through SQL*Net, then you must set the TWO_TASK variable before you begin.

The script is located in the admin/sql subdirectory under $FND_TOP. Use the script as follows:

```
sqlplus <apps_user>/<apps_passwd> @wfupartb <fnd_user> \
<fnd_passwd> <apps_user> <apps_passwd>
```

For example:

```
sqlplus apps/apps @wfupartb applsys apps apps apps
```

If the script fails, you must perform any necessary cleanup manually. Since the script's operations are DDL operations running in nologging mode, rollback is not possible.

Context: You need to perform this step only once.

Step 2 Setting Global User Preferences

You can control how you interact with Oracle Workflow by specifying user preferences that you can set from the User Preferences web page. As a workflow administrator, you also have access to the Global Preferences web page, which you can use to globally set default user preference values for the entire enterprise. An individual user can override a default user preference at any time by changing the value of the user preference in the User Preferences web page. Both web pages are accessible from the Oracle Workflow Home page, but only a workflow administrator has access to the Global Preferences page.

Attention: The Language, Territory, and Notification preference settings in the Global Preferences and User Preferences web pages are valid only if your directory service views map the Language, Territory, and Notification_Preference columns to the Oracle Workflow preferences table. If you map to some other preference source or set a hardcoded value to these columns, any changes you make to the preferences via the preferences web pages are ignored. See: Setting Up an Oracle Workflow Directory Service: page 2 – 17.

Context: You need to perform this step only once.

See: Setting User Preferences: page 9 – 4.

▶ **To Set Global User Preferences**

1. Use a web browser to connect to the Oracle Workflow home page, then choose the Global Preferences link:

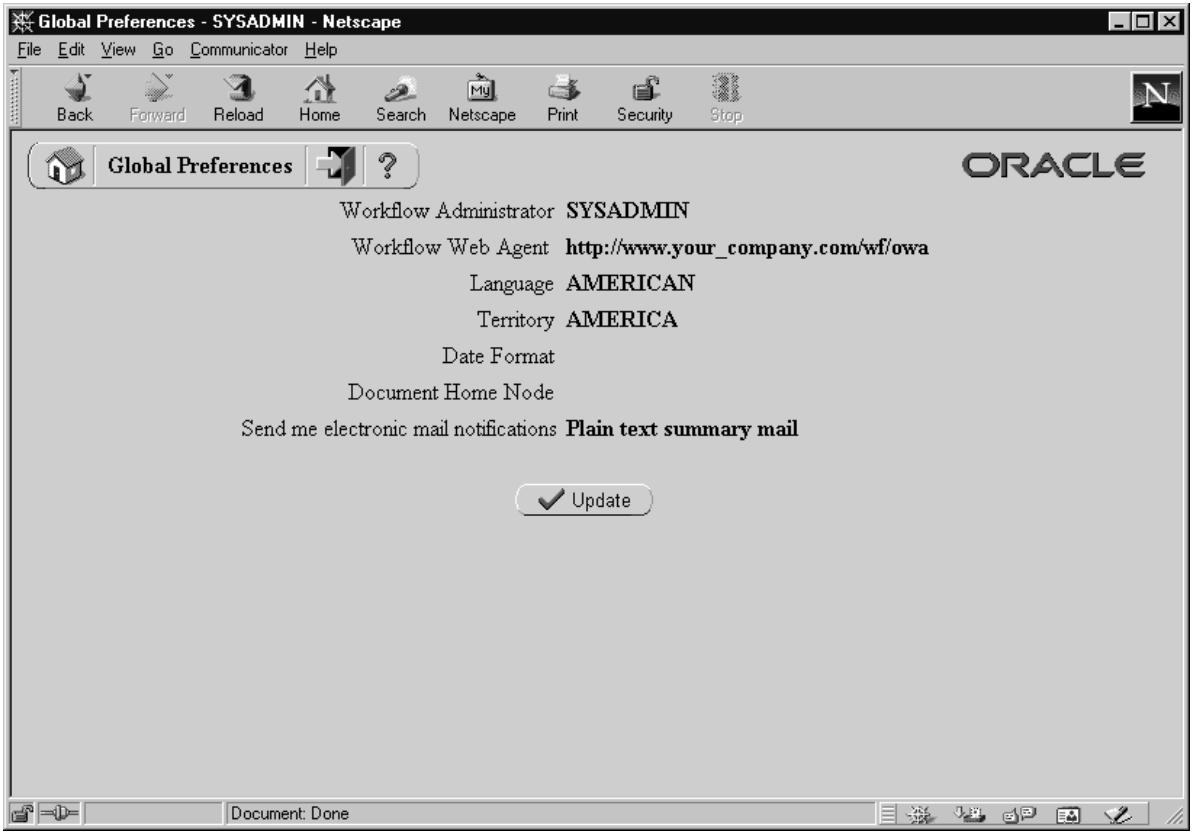
   `<webagent>/wfa_html.home`

   Alternatively, you can connect directly to the Global Preferences web page:

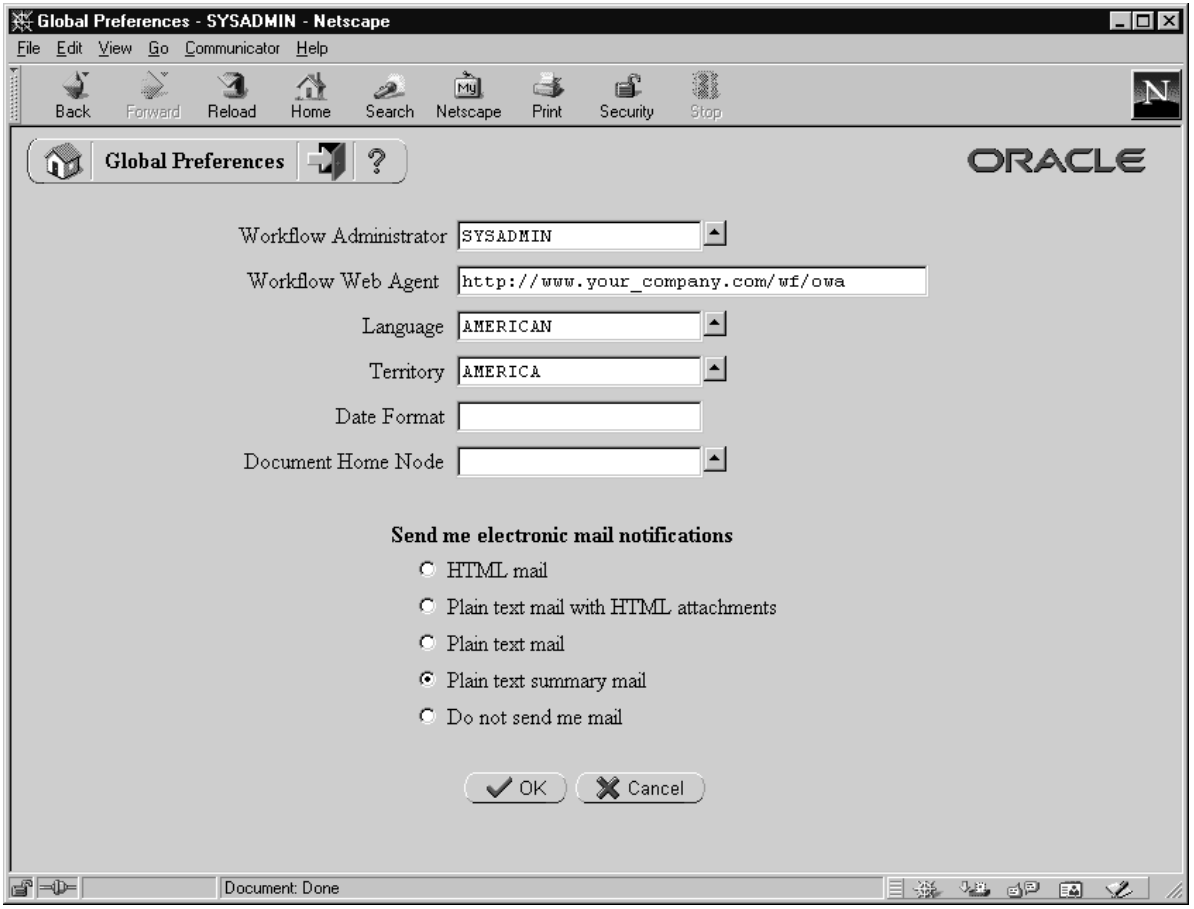
   `<webagent>/wf_pref.edit?edit_defaults=Y`

   `<webagent>` represents the base URL of the web agent you configured for Oracle Workflow in your Web server.

Attention: These are secured pages, so if you have not yet logged on as a valid user in the current web session, you will be prompted to do so before the page appears.



2. The Global Preferences web page displays a summary of your current global preferences. Choose Update to modify these preferences.



3. In the Workflow Administrator field, use the list of values to select the role to which you want to assign workflow administrator privileges. Any user associated with this 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.

   If you want all users and roles to have workflow administration privileges, such as in a development environment, enter an asterisk (*) in the Workflow Administrator field. See: Setting Up an Oracle Workflow Directory Service: page 2 – 17.

4. In the Workflow Web Agent field, enter the base URL of the Oracle web agent you defined for Oracle Workflow in Oracle WebDB, Oracle Web Application Server, or Oracle Application Server.

Caution: The list of values fields that are implemented in many of Oracle Workflow's web pages will not function properly unless you specify the base URL of your Oracle Workflow web agent in this field.

- The base URL should look like this if you are using Oracle WebDB as your Web server:

  http://<*server.com:portID*>/<*DAD_name*>

  `<server.com:portID>` represents the server and TCP/IP port number on which your Web Listener accepts requests and `<DAD_name>` represents the name of the DAD you configured for the Oracle Workflow database schema.

- The base URL should look like this if you are using Oracle Web Application Server as your Web server:

  http://<*server.com:portID*>/<*plsqlagent*>/plsql

  `<plsqlagent>` represents the name of the PL/SQL Agent you configured for the Oracle Workflow database schema.

- The base URL should look like this if you are using Oracle Application Server as your Web server:

  http://<*server.com:portID*>/<*plsql_agent_virtual_path*>

  `<PLSQL_agent_virtual_path>` represents the application virtual path of the PL/SQL agent you configure for the Oracle Workflow database schema.

See your Oracle WebDB, Oracle Web Application Server, or Oracle Application Server documentation for more information.

Attention: If you are using the version of Oracle Workflow embedded in Oracle Applications, you should also edit the APPS_WEB_AGENT profile option.

5. In the Language and Territory fields, use the list of values to select the NLS_LANGUAGE and NLS_TERRITORY combination that defines the default language–dependent behavior and territory–dependent formatting of your users' notification sessions.
6. In the Date Format field, enter an Oracle8–compliant date format that defines the default date format for the workflow database sessions of all users.
7. In the Date Format field, specify an Oracle8–compliant date format that defines the default date format for the workflow database sessions of all users. An example of an Oracle8–compliant date

format is DD–Mon–RRRR. If you do not specify a date format, then the date format defaults to DD–MON–YYYY.

Note: Oracle Workflow may include a time element when relevant for certain displayed dates, even if you do not include a time format with your date format. If you specify a time format along with your date format, then in those situations when Oracle Workflow displays a time element, you will see two time elements following your date.

8. In the Document Home Node field, use the list of values to select the default Document Management repository that users access when they attempt to attach or access an attached document management document.
9. In the 'Send me electronic mail notifications' section, check a notification preference:
   - HTML mail—send notifications as HTML E–mail. Users must read their mail using an HTML E–mail viewer.
   - Plain text mail with HTML attachments—send notifications as plain text E–mail but include the HTML–formatted version of the notifications as attachments.
   - Plain text mail—send notifications as plain text E–mail.
   - Plain text summary mail—send a summary of all notifications as plain text E–mail. Users must use the Notifications web page to take action on individual notifications.
   - Do not send me mail—do not send the notifications as E–mail. Users must view the notifications and take action from the Notifications web page.
10. Check OK once you are satisfied with your changes.

Note: These global language, territory, document home node, and notification preferences are saved to the Oracle Workflow Preferences table for a special user name called –WF_DEFAULTS–. The workflow administrator role and workflow web agent information is saved to the Workflow Resources table.

Step 3 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
- WF_USER_ROLES

In addition, Oracle Workflow provides three local tables called WF_LOCAL_USERS, WF_LOCAL_ROLES, and WF_LOCAL_USER_ROLES that contain columns similar to those defined in the WF_USERS, WF_ROLES, and WF_USER_ROLES views, respectively. You can use these tables to store users and roles not included in your existing directory repository by calling the appropriate Directory Service PL/SQL API. See: Workflow Directory Service APIs: page 8 – 77.

Attention: You should avoid selecting from DUAL to incorporate additional users and roles into the directory service views as this allows you to violate the unique constraint on certain columns of the views and reduces performance with unnecessary joins between the 'select from DUAL' statements.

Context: You need to perform this step only once.

See: Predefined Directory Services: page 2 – 22

See: Ad Hoc Users and Roles: page 5 – 19

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`, `MAILHTML`, or `MAILATTH` allows users to receive and respond to notifications by plain text E–mail, HTML–formatted E–mail or by plain text E–mail with HTML attachments, respectively. A value of `QUERY` allows users to query notifications from the Notifications Web page. 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. See: Overview of Notification Handling: page 10 – 2 and Notification Preferences: page 2 – 39.

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

  Note: You can map the Notification_Preference column over the Oracle Workflow preferences table using the statement below. The benefit of this is that you can then globally set the default notification preference for all users in your enterprise using the Global Preferences web page and let individual users override that default value by changing their notification preference in the User Preferences web page. See: Global Preferences: page 2 – 12, User Preferences: page 9 – 4 and get_pref: page 8 – 100.

  ```
  NVL(wf_pref.get_pref(USR.USER_NAME,'MAILTYPE'),
  'MAILHTML')
  ```

- Language—The value of the Oracle8 NLS_LANGUAGE initialization parameter that specifies the default language–dependent behavior of the user's notification session. Refer to your Oracle8 user's guide or installation manual for the list of supported language conventions.

  Note: You can globally set the language for all the users in your enterprise, by specifying a language in the Global

Preferences web page. Individual users may override that default value by changing their language in the User Preferences web page. See: See: Global Preferences: page 2 – 12, User Preferences: page 9 – 4 and get_pref: page 8 – 100.

Note: You can map the Language column over the Oracle Workflow preferences table using the statement below. The benefit of this is that you can then globally set the default Language for all users in your enterprise using the Global Preferences web page and let individual users override that default value by changing their Language in the User Preferences web page. See: Global Preferences: page 2 – 12 and User Preferences: page 9 – 4.

```
NVL(wf_pref.get_pref(USR.USER_NAME,'LANGUAGE'),
FNDL.NLS_LANGUAGE)
```

Attention: Make sure that the E–mail templates used by the Notification Mailer to send notifications has been translated by Oracle to the language you wish to set. The E–mail templates are delivered in a file called wfmail.wft under the subdirectory $ORACLE_HOME/wf/res/<*lang*>. You can check the appropriate language subdirectory to verify if the templates have been translated to the language you wish to set. See: Modifying Your Message Templates: page 2 – 56.

- Territory—The value of the Oracle8 NLS_TERRITORY initialization parameter that specifies the default territory–dependant formatting used in the user's notification session. Refer to your Oracle8 user's guide or installation manual for the list of supported territory conventions.

  Note: You can map the Territory column over the Oracle Workflow preferences table using the statement below. The benefit of this is that you can then globally set the default Territory for all users in your enterprise using the Global Preferences web page and let individual users override that default value by changing their Territory in the User Preferences web page. See: Global Preferences: page 2 – 12, User Preferences: page 9 – 4 and get_pref: page 8 – 100.

  ```
  NVL(wf_pref.get_pref(USR.USER_NAME,'TERRITORY'),
  FNDL.NLS_TERRITORY)
  ```

- 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.
- Expiration_Date—The date at which the user is no longer valid in the directory service.

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. Those columns that are preceded by an asterisk (*) are similar to the matching column described for `the WF_USERS` view:

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—The internal name of the role. The Name column must be sourced from a column that is less than or equal to 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.

- *Display_Name
- Description
- Notification_Preference
- Language
- Territory
- Email_Address—if the E–mail address is null for a given role, the Notification Mailer sends an individual E–mail to each user within the role.
- Fax
- Orig_System
- Orig_System_ID
- Expiration_Date

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. Oracle8 is 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 8 – 77.

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 14 – 8.

▶ 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 maps over Oracle Human Resources tables,

Oracle Application Object Library tables, various Oracle Applications tables, and the WF_LOCAL tables.

Oracle Workflow provides a sql script that defines the WF_USERS, WF_ROLES, and WF_USER_ROLES views that 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 *admin/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 RDBMS. You base your views on the tables `DBA_USERS`, `WF_LOCAL_USERS`, `DBA_ROLES`, and `WF_LOCAL_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 and any users stored in `WF_LOCAL_USERS`. For each DBA user, the originating system is called `ORACLE`, and the originating system ID is the `USERNAME` column in `DBA_USERS`. The default notification preference for each DBA user is `MAILHTML`.

The `WF_ROLES` view includes all users in the `WF_USERS` view, all roles defined in the `WF_LOCAL_ROLES` table, and all roles in `DBA_ROLES`, where `role_name` begins with `WF`. For each DBA role, the originating system is `ORACLE` and the originating system ID is the `ROLE` column in `DBA_ROLES`. The default notification preference for each DBA role is `MAILHTML`.

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 and the users and roles of your directory repository are not stored in any existing database tables, you can enter your users and roles information in the WF_LOCAL tables and map your directory service to these 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 your custom directory repository.

  The originating system in the WF_USERS view is called WF_LOCAL_USERS, 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_ROLES 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.

Step 4 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 character 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.

Context: You need to perform this step only once.

See: Oracle8i National Language Support Guide

▶ **To display user defined objects in Oracle Workflow Builder in other languages (for standalone version only)**

1. Install Oracle Workflow Cartridge.
2. Create the WF_LANGUAGES view in your workflow server.
3. Run the script wfnlena.sql to enable the language of interest. See: wfnlena.sql: page 14 – 9.
4. Run the script WFNLADD.sql to create rows for the enabled language in each workflow object translation table. See: WFNLADD.sql: page 14 – 4.
5. Set the NLS_LANG environment variable for the new language.

For example, in UNIX, use the command:

```
setenv NLS_LANG = 'language.territory_characterset'
```

For Windows NT, run the `regedit32` command and locate the `NLS_LANG` setting under the `HKEY_LOCAL_MACHINE/SOFTWARE/ORACLE` hierarchy. Double click on `NLS_LANG`, then set the variable to the new value and save your edit.

6. Create a translated version of your workflow process definition and save it as a flat file (.wft).
7. Load the translated .wft file to your workflow database, making sure that the current NLS_LANG setting is correct.

   Note: In Oracle Workflow Release 2.0.3, both the Workflow Definitions Loader and the Workflow Resource Generator use the current setting of NLS_LANG to determine which language to load.

   In Oracle Workflow Release 2.5, to determine the language to load, the Workflow Definitions Loader uses the language specified in the .wft file, while the Workflow Resource Generator accepts a language parameter. If you do not specify a language parameter, the Workflow Resource Generator defaults to the current setting of NLS_LANG.

▶ **To display an Oracle Workflow web session in other languages (for standalone version only)**

- If you have multiple languages installed for Oracle Workflow, as a workflow administrator, you can specify the default language that your users' web sessions display by setting the Language parameter in the Global User Preferences web page. Individual users can override the default language by setting the Language parameter in the User Preferences web page. See: Setting Global User Preferences: page 2 – 12 and Setting User Preferences: page 9 – 4.

  Note: Oracle Workflow web sessions ignore the NLS settings that you specify in your Oracle Web Application Server or Oracle Application Server DAD.

▶ **To display E–mail notifications in other languages**

1. Determine if Oracle has translated the E–mail notification templates to the language you wish to set by checking for a file called wfmail.wft in the appropriate language subdirectory

$ORACLE_HOME/wf/res/<*lang*>. See: Modifying Your Message Templates: page 2 – 56.

2. If the E–mail templates are available for the language you wish to display, you can set your users and roles' default language setting to that language in the Global Preferences web page. See: Setting Global User Preferences: page 2 – 12.

Step 5 Setting the Socket Listener Activated Profile Option.

The Notification Details web page can now display an attached form icon to support form attributes in a notification message. Oracle Applications users can launch the Oracle Workflow Notification Worklist from their Oracle Applications menus.

From the Worklist, users can select a notification link to display the contents of a notification in the Notification Details page. If the notification details display an attached form icon, users can click on that icon to launch an Oracle Applications form.

Before Oracle Workflow can launch the form from the Notification Details page, it must check for appropriate context information with Oracle Applications. To accomplish this, the socket listener on the form side must be active.

You can activate the Oracle Applications socket listener by setting the Socket Listener Activated profile option to Yes using the System Profile Values Window.

In addition, the Workflow Administrator needs to specify the following token values in $FND_TOP/resource/<*language*>/wfcfg.msg for the Java plugin:

```
WF_CLASSID              <Class ID for Jinitiator>
    (Required if you are using Microsoft Internet Explorer.)
WF_PLUGIN_DOWNLOAD      <Plugin location>
    (Such as http://<server>/OA_JAVA/.)
WF_PLUGIN_VERSION       <Plugin version>
    (Such as 1.1.7.18.)
```

Run the Workflow Resource Generator to load the contents of wfcfg.msg into the WF_RESOURCES table.

Context: You need to perform this step only once.

See: Overview of Setting User Profiles: *Oracle Applications System Administrator's Guide*

See: To run the Workflow Resource Generator: page 8 – 62

Step 6 Setting the http_proxy Environment Variable

The Notification Mailer is now capable of fetching the URL content of a URL attribute, if the attribute has Attach Content checked in its Attribute property page. If the referenced URL is outside your organization's firewall, the Notification Mailer must go through your organization's proxy server to fetch the URL content. Before starting your Oracle Workflow Database and the Notification Mailer, set up an environment variable called http_proxy to point to your proxy server.

For example, in UNIX, use the following command to set http_proxy to a proxy server called www–proxy.yourcompany.com:80:

```
setenv http_proxy "http://www-proxy.yourcompany.com:80/"
```

If your Oracle Workflow database is a remote database, edit your Oracle Net8 listener.ora file to include the http_proxy environment variable in your SID description. For example, in the following SID description, http_proxy is set to `www-proxy.yourcompany.com:80`:

```
(SID_DESC=
    (SID_NAME=V804)
    (ORACLE_HOME=/oracle/app/oracle/product/8.0.4)
    (ENVS='http_proxy=www-proxy.yourcompany.com:80')
)
```

Context: You need to perform this step only once.

Step 7 Setting the WF_RESOURCES Environment Variable

If you are using the standalone version of Oracle Workflow and your Workflow server is installed on a UNIX platform, 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.

Attention: Do not enclose environment variable values in double quotes (" ") as it is not supported.

You do not need to set this environment variable if your Workflow server is installed on the Windows NT platform. The Workflow server installation on Windows NT automatically sets a WF_RESOURCES registry setting that identifies the path of the wf<*language*>.res file.

You also 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.

Context: You need to perform this step only once.

Step 8 Defining Document Management Repositories

Oracle Workflow provides seamless open integration with Oracle's third party document management (DM) integration partners.

Oracle Workflow communicates with these DM systems through a web server agent interface. You should be able to identify your DM system's web server using the syntax *<protocol>://<server:port>/*.

To register the DM system with Oracle Workflow, use the Oracle Workflow Document Nodes web page to identify the DM system as a document node. Oracle Workflow uses the document node as the gateway to a specific document management system and uses the information defined in the node to construct the URLs that display the document management system documents.

Context: You need to perform this step only once.

▶ To Define a Document Node

1. Use a web browser to connect to the Oracle Workflow home page and choose the Document Nodes link, if you have workflow administrator privileges:

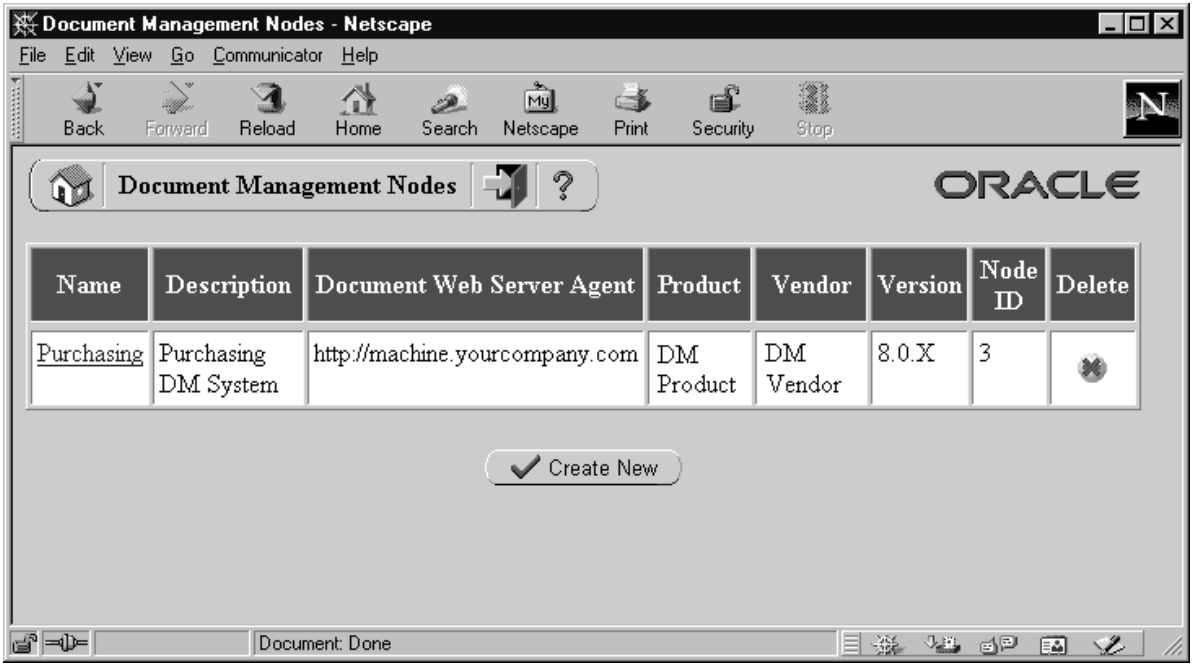
   `<webagent>/wfa_html.home`

   Alternatively, if you have workflow administrator privileges, you can connect directly to the Document Management Nodes web page:

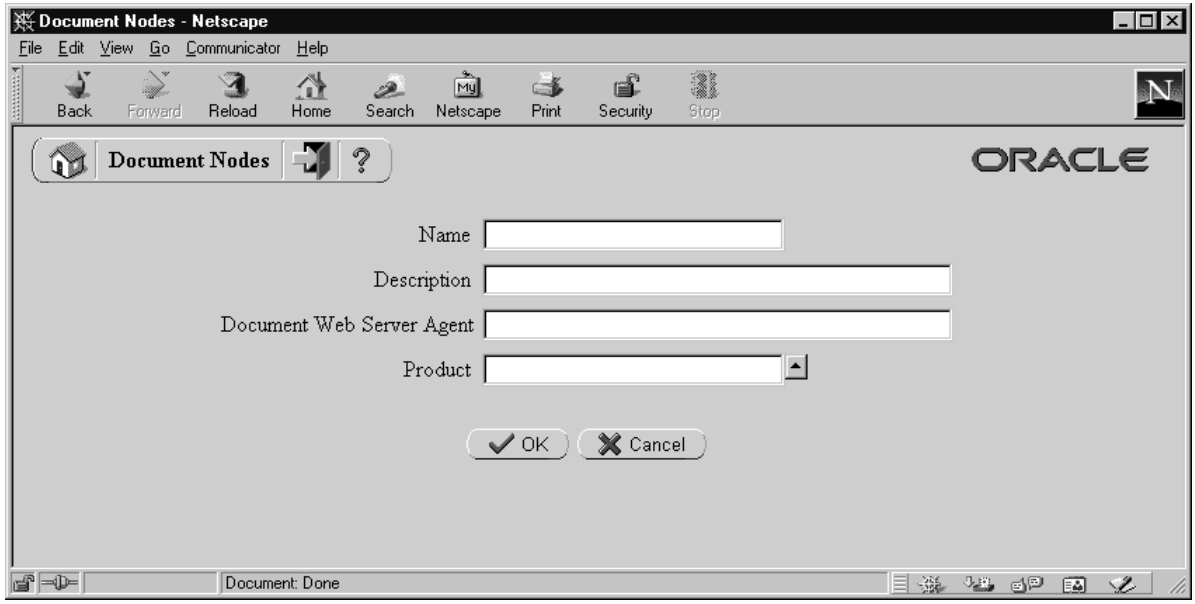
   `<webagent>/fnd_document_management.dm_nodes_display`

   `<webagent>` represents the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 12.

   Attention: These are secured pages, so if you have not yet logged on as a valid user in the current web session, you will be prompted to do so before the page appears.



2. The Document Management Nodes web page displays a summary of any previously defined document management nodes. The summary displays the DM system node name, DM system version number, node ID, node description, and web server for the DM system. Also shown is the name of the DM system product, the DM product vendor and the DM system version number.
3. You can click on a node name link to edit the details of a node or choose Create Node to define a new node in the Document Nodes web page.



4. In the Document Nodes web page, enter or edit the name and description of the node.
5. Enter or edit the web server agent for the document management system. The syntax should be *<protocol>://<server:port>*.
6. Choose the document management product name for this node. This is the name that represents the node for all users who wish to link documents to their workflow processes.
7. Choose OK to update or define the node.

   Once you define a node, the browser returns you to a reloaded Document Management Nodes web page. Oracle Workflow assigns a node ID for the new node.
8. You can also delete a node by clicking on the Delete icon in the Delete column.

   Caution: Before you delete a node, you should ensure that no Document–type attributes currently reference the node.

Step 9 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).

Context: You need to perform this step only once.

See: Activity Cost: page 4 – 40

See: Timeout Transitions: page 5 – 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 8 – 33.

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 – 35

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

See: 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 SQLPLUS or in the next function activity so that other activities are processed as expected.

Step 10 Implementing the Notification Mailer

The Notification Mailer is a program that performs E–mail send and response processing for the Oracle Workflow Notification System. You need to perform this step only if you wish to have your workflow users receive their notifications via E–mail, as well as from the Notifications Worklist web page. The Notification Mailer 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 Internet Messaging 4.2, 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.

The E–mail notifications are based on standard templates defined in Oracle Workflow Builder. The templates describe the syntax the reply should follow and list the information needed to confirm the notification. The generated E–mail message also includes any custom site information, the due date, and any information necessary to process the response. See: Modifying Your Message Templates: page 2 – 56.

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 – 54. 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.

Context: You need to perform this step only once.

See: Reviewing Notifications via Electronic Mail: page 10 – 2

Full MIME Support

Oracle Workflow fully supports MIME (Multi–purpose Internet Mail Extensions) encoded messages. This means that the Notification Mailer can exchange messages with workflow users containing languages with different character sets and multi–media encoded content.

Notification Preferences

Oracle Workflow allows you to determine how you view notifications by setting a notification preference in the User Preferences web page. See: Setting User Preferences: page 9 – 4.

Often times, the functionality of a user's mail reader determines what the user's notification preference should be. Some mail readers can only display plain text, others can display HTML formatting, while still others can only display HTML formatting in an attachment. Five notification preferences are available:

- Plain text mail (MAILTEXT)—The notification message appears as plain text, with no attachments. See: Plain Text E–mail: page 2 – 40.
- HTML mail (MAILHTML)—The notification message appears as HTML–formatted text, with at least one other attachment that is a link to the notification in the Notifications web page. If the notification message has 'Content–Attached' message attributes, these attributes appear as additional attachments to the message. See: HTML–Formatted E–mail: page 2 – 41.

Attention: If you wish to view notifications with HTML formatting, but your mail reader is not able to interpret HTML formatting in the mail message body, change your notification preference to 'Plain text mail with HTML attachments' (MAILATTH). The MAILATTH preference delivers an HTML–formatted version of the notification as an attachment to the plain text notification.

- Plain text mail with HTML attachments (MAILATTH)—The notification message appears as plain text, with at least two other attachments. One attachment is an HTML–formatted version of

the message, and the other is a link to the notification in the Notifications web page. If the notification message has 'Content–Attached' message attributes, these attributes appear as additional attachments to the message. See: Plain Text E–mail with an HTML Attachment: page 2 – 42.

- Plain text summary mail (SUMMARY)—The message is a plain text summary of all open notifications. To respond to the individual notifications in the summary, you must access the notifications from the Notifications web page.
- Do not send me mail (QUERY)—The Notification Mailer does not send you E–mail notifications. Instead you must query and respond to your notifications from the Notifications web page.

Attention: You can always query and respond to your notifications from the Notifications web page, even if you set your notification preference to send you mail.

See: Reviewing Notifications via Electronic Mail: page 10 – 2

See: Viewing Notifications from a Web Browser: page 10 – 13

See: Reviewing a Summary of Your Notifications via Electronic Mail: page 10 – 23

Plain Text E–mail

If the performer of a notification has a notification preference of plain text mail (MAILTEXT), the notification is flagged as such in the Workflow Notification table. When the Notification Mailer polls the Notification table and identifies that flag, it generates a plain text E–mail notification from the information in the table and sends it to the performer role. The Notification Mailer uses the Text Body defined for the message in the Oracle Workflow Builder message property page to generate the plain text E–mail. It token replaces all attribute values referenced in the message body with plain text values. For example:

- PL/SQL Document attributes are token replaced with a plain text version of a PL/SQL document.
- Document Management–type document attributes are token replaced with the plain text display name of the document attribute. To display the contents of the Document Management document, you need to view your notification from the Notifications Worklist web page. See: Viewing Notifications from a Web Browser: page 10 – 13 and To Access a DM Document Sent by a Notification: page 10 – 31.

- URL attributes are token replaced with the display name of the URL attribute, followed by a colon and the URL:

  <URL_Attribute_Display_Name>:<URL>

Attention: Message attributes that have Attach Content checked in their Attributes property page, are attached as plain text to their parent notification. Note that this may render some attachments unreadable if the attachment includes special formatting or your plain text E–mail reader does not recognize attachments. To view these attachments, you should display your notifications in the Notifications Worklist web page. See: Viewing Notifications from a Web Browser: page 10 – 13.

A recipient of a plain text E–mail notification responds by manually replying to the notification and entering response values following the instructions provided in the notification. See: To Respond to a Plain Text E–mail Notification Using Templated Response: page 10 – 4 and To Respond to a Plain Text E–mail Notification Using Direct Response: page 10 – 6.

HTML–Formatted E–mail

If the performer of a notification has a notification preference of HTML mail (MAILHTML), the notification is flagged as such in the Workflow Notification table. When the Notification Mailer polls the Notification table and identifies that flag, it generates an HTML–formatted E–mail notification from the information in the table and sends it to the performer role. The performer role should use an E–mail reader that can interpret and display HTML content within a message body.

> **Note:** If your E–mail reader cannot interpret HTML formatting in a message body, you should set your notification preference to plain text mail with HTML Attachments (MAILATTH).

The Notification Mailer uses the HTML Body defined for the message in the Message Body property page to generate the HTML E–mail. If no HTML Body is defined, it uses the Text Body to generate the HTML mail. The Notification Mailer token replaces all message attributes referenced in the message body with HTML–formatted values. For example:

- PL/SQL document attributes are token replaced with HTML text or plain text between <pre>...</pre> HTML tags.
- Document Management document attributes are token replaced with HTML anchor. When you select such an anchor, your E–mail reader takes you to the target document management

integration screen to display the referenced document. See: To Access a DM Document Sent by a Notification: page 10 – 31

- URL attributes are token replaced with HTML anchors. When you select such an anchor, your E–mail reader takes you to the target URL page.

  Note: Message attributes that have Attach Content checked in their Attributes property page, are appended as HTML–formatted attachments to their parent message. For example:

  – If the message attribute is a URL attribute or a PL/SQL document attribute, the full URL content or fully generated PL/SQL document is fetched and attached to the message. The Notification Mailer does not have any special handling of image, video, or audio URL content.

  – If the message attribute is a Document Management document attribute, a URL anchor to the document management integration screen is attached. You need to select the anchor to display the login page of the document management integration screen. Once you log in, the screen displays the referenced document.

An HTML–formatted notification always includes at least one attachment. The attachment is called Notification Detail Link. When you select this attachment, your E–mail reader opens a browser window that displays your notification in the Notification Details web page. You can respond directly to your notification from this web page, bypassing the need to process your response through the Notification Mailer.

Alternatively, you can respond to your HTML–formatted notification by clicking on a link that represents the response in the HTML message body. The response link generates a plain text E–mail response that includes a response template modified with the predefined response value that you select. See: To Respond to an HTML E–mail Notification: page 10 – 10.

Plain Text E–mail with an HTML Attachment

If the performer of a notification has a notification preference of plain text mail with HTML attachments (MAILATTH), the notification is flagged as such in the Workflow Notification table. When the Notification Mailer polls the Notification table and identifies that flag, it generates a plain text E–mail notification with HTML attachments from the information in the table and sends it to the performer role.

The performer role should use an E–mail reader that supports HTML attachments.

The Notification Mailer uses the Text Body defined for the message in the Message Body property page to generate the plain text body of the E–mail. It also generates an HTML version of the notification message and sends it as an attachment called HTML Message Body to the plain text E–mail. The Notification Mailer generates the content of the HTML attachment from the HTML Body defined for the message. If no HTML Body is defined, it uses the Text Body to generate the HTML mail. The Notification Mailer token replaces all message attributes referenced in the plain text message body with plain text values and token replaces all message attributes referenced in the attached HTML message with HTML–formatted values. See: Sending Plain Text E–mail: page 2 – 40 and Sending HTML–Formatted E–mail: page 2 – 41.

> **Note:** If your E–mail reader supports HTML–formatting in the message body, then the HTML attachment will also appear in line in the message body.
>
> **Note:** Message attributes that have Attach Content checked in their Attributes property page, are appended as HTML–formatted attachments. For example:
>
> – If the message attribute is a URL attribute or a PL/SQL document attribute, the full URL content or fully generated PL/SQL document is fetched and attached to the message.
>
> – If the message attribute is a Document Management document attribute, a URL anchor to the document management integration screen is attached. You need to select the anchor to display the login page of the document management integration screen. Once you log in, the screen displays the referenced document. See: To Access a DM Document Sent by a Notification: page 10 – 31

The notifications received by a user whose notification preference is 'Plain text with HTML attachments' always contain at least two attachments. The first attachment is HTML Message Body and the other is Notification Detail Link. When you select Notification Detail Link, your E–mail reader opens a browser window that displays your notification in the Notification Details web page. You can respond directly to your notification from this web page, bypassing the need to process your response through the Notification Mailer. See: To Respond to a Plain Text E–mail Notification with an HTML Attachment: page 10 – 11.

Alternatively, a recipient of this type of notification can respond in one of two other ways:

- Manually reply to the notification and enter response values following the instructions provided in the notification. See: To Respond to a Plain Text E–mail Notification Using Templated Response: page 10 – 4 and To Respond to a Plain Text E–mail Notification Using Direct Response: page 10 – 6.
- Display the HTML Message Body attachment and click on the HTML link that represents the response. The response link generates a plain text E–mail response that includes a response template updated with the predefined response value you select.

Starting the Notification Mailer

You can install and set up the Notification Mailer to run against UNIX Sendmail, Oracle Internet Messaging 4.2, or a MAPI–compliant mail application on Windows NT. However, before doing so, you must set up a least one mail account dedicated to 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.

Users can receive E–mail notifications using any E–mail reader that is MAPI–compliant running on Windows NT or that Oracle Internet Messaging or UNIX Sendmail can provide a gateway to.

See: Setting the http_proxy Environment Variable: page 2 – 29

See: To Start the Notification Mailer for UNIX Sendmail or Oracle Internet Messaging 4.2 (For the Standalone Version of Oracle Workflow): page 2 – 44

See: To Start the Notification Mailer for UNIX Sendmail or Oracle Internet Messaging 4.2 (For the Version of Oracle Workflow Embedded in Oracle Applications): page 2 – 45

See: To Start the Notification Mailer for MAPI–Compliant Mail Applications: page 2 – 46

See: To Create a Configuration File for the Notification Mailer: page 2 – 47

See: To Run a Perpetual Shell Script for the Notification Mailer: page 2 – 54

See: Response Processing: page 2 – 54

▶ **To Start the Notification Mailer for UNIX Sendmail or Oracle Internet Messaging 4.2**

For the Standalone Version of Oracle Workflow:

1. The Notification Mailer resides on your server in the *$ORACLE_HOME/bin* subdirectory. To run the Notification Mailer, use the following command syntax:

   wfmail.<xxx> -f <config_file>

   Replace *<xxx>* with `ofc` to use the Oracle Internet Messaging 4.2 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.

   Attention: To use the Oracle Internet Messaging 4.2 version of the Notification Mailer, you must be able to connect to a full installation of the Oracle Internet Messaging 4.2 Messaging Option mail server. The Internet Messaging 4.2 mail server installation does not have to reside on the same $ORACLE_HOME or even on the same machine as Oracle Workflow. However, you must ensure that all of Oracle Internet Messaging 4.2's message files are present in the Notification Mailer's running environment. To accomplish this, you can create links under your $ORACLE_HOME where the Notification Mailer resides, to the $ORACLE_HOME/oacore21 and $ORACLE_HOME/office subdirectories of your Oracle Internet Messaging 4.2 installation area. Alternatively, you can manually create the oacore21 and office subdirectories under your $ORACLE_HOME and copy over the files from the $ORACLE_HOME/oacore21 and $ORACLE_HOME/office subdirectories of your Internet Messaging 4.2 installation area.

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 for *config_file* can be:

   `@<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>*\bin.

2. Start the Notification Mailer program by entering the following command in an MS–DOS prompt window:

   `<drive>:\<ORACLE_HOME>\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>\wf\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 Internet Messaging 4.2, the account information would be an Oracle Internet Messaging 4.2 database account of the format, *username/password@connect_string* (or *alias*). For Sendmail, the account information would be the full file path of the mail spool file where incoming messages are stored, such as /var/mail/applmgr3. Note that this should correspond to the account from which you start the Notification Mailer, in this example, applmgr3.

Attention: If you want to start the Sendmail version of the Notification Mailer, you must also specify the full path of the Sendmail executable directory in your PATH environment variable.

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 a unique node name for each Notification Mailer configuration file 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 `wf`.
   - **FROM**—The value that appears in the From: field of the message header when a notification message is delivered to a user. The default is `Oracle Workflow`.

- **SUMMARY_ONLY**—(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', 'MAILATTH', 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', 'MAILATTH', 'MAILHTML', or 'SUMMARY'. See: Setting Up Users and Roles from a Directory Repository: 2 – 17.

  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 an 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.

- **DIRECT_RESPONSE**—Specify N or n to send plain text notifications requiring a templated response to users with a notification preference of 'MAILTEXT' or 'MAILATTH'. Specify Y or y to send plain text notifications requiring a direct response to users with these preferences. The default is N.

  For the templated response method, users must reply using a template of response prompts and enter their response values between the double quotes following each prompt. For the direct response method, users must enter their response values directly as the first lines of a reply. See: Reviewing Notifications via Electronic Mail: page 10 – 2.

- **ALLOW_FORWARDED_RESPONSE**—Indicate whether to allow a user to respond to an E–mail notification that has been forwarded from another role. Valid values include `Y` or `N`. The default is `Y`.

  If the value is N, the Notification Mailer will check if the "From:" E–mail address of the notification response exactly matches the

E–mail address of the recorded recipient role (or the E–mail address of a user in that role). If the two E–mail addresses match exactly, meaning the notification was not forwarded or was forwarded according to a valid routing rule, the Notification Mailer treats the response as a valid response. If the two E–mail addresses do not match exactly, meaning the notification was simply forwarded using the E–mail Forward command, the Notification Mailer does not process the response and treats it as unsolicited mail.

If the value is Y, the Notification Mailer never checks the "From:" E–mail address of the notification response and always allows the response to be processed.

Warning: Note that there are limitations when you set ALLOW_FORWARDED_RESPONSE to N. For example, suppose a notification is sent to a distribution list mail alias that does not have a USER/ROLE relationship in the Oracle Workflow directory service. If any user from the distribution list responds to the notification, the Notification Mailer will always treat their notification response as unsolicited mail, because the "From:" E–mail address, which is an individual user's E–mail address, will never match the distribution list mail alias.

- **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 `30`.
- **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**—(Required) 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 Workflow Web Agent specified in the Global Preferences web page, but you can override this default by entering a different value for this parameter. See: Setting Global User Preferences: page 2 – 12.
- **HTMLFILE**—The filename of the default attachment. The default is `attach.html`.
- **HTMLDESC**—A description of the default attachment. The default is `HTML`.
- **HTMLTYPE**—The html attachment type number. This argument is required only if you are using Oracle Internet Messaging 4.2 and want to include an HTML attachment with your E–mail notification. The default is `10003`.

Attention: The HTMLFILE, HTMLDESC, and HTMLTYPE parameters should only be used to specify the default attachment of the Notifications web page. These parameters are not intended for attaching other types of documents to notification messages. To include other types of documents in a notification, you can embed a URL in the body of the message that resolves to a copy of your document. See: To Define a Message Attribute: page 4 – 29.

- **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 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: Only a message response that contains a notification ID is checked by the tag file. If the Notification Mailer receives a message response that does not contain a notification ID, it moves the message response to the discard folder and sends a 'Warning' message to the sender that it received unsolicited mail. See: Workflow Warning Mail Message: page 2 – 66.

The format used in the tag file is

`Status "Matching string"`

where *`Status`* can be the value: `ERROR, IGNORE,` or `UNAVAIL` and *`"Matching string"`* is the text to look for in the From: line, Subject: line, or body of the message. The Notification Mailer handles a message assigned one of these status values as follows:

- `IGNORE`–moves the message to the discard folder and continues waiting for a valid reply to the open notification. The notification's status is still OPEN and its mail status is still SENT.

- ERROR–moves the message to the discard folder and initiates an error process, if one is defined. The notification's status is still OPEN, but its mail and activity status are both updated to ERROR. Ideally, the workflow administrator corrects the problem and resends the notification by updating its mail status to MAIL.
- UNAVAIL (or any other user defined tag)–moves the message to the discard folder and continues waiting for a reply to the notification since the notification's status is still OPEN, but its mail status is updated to UNAVAIL. The UNAVAIL status is purely informative, as no further processing occurs with this notification.

The Notification Mailer can also assign an INVALID status to a message response, if the returned response value is not a valid value in the assigned lookup (result) type. In this case, it moves the message to the discard folder, and sends an 'Invalid' message but does not alter the notification's status or mail status, so that it continues to wait for a valid reply. See: Workflow Invalid Mail Message: page 2 – 62.

Attention: It is important that you uniquely identify bounced messages and auto–replies from normal responses in the tag file. If you do not identify bounced and auto–reply messages, the Notification Mailer can mistake these as invalid responses, send an 'Invalid' message and continue to wait for a reply. In both cases a perpetual loop would occur where the Notification Mailer keeps sending out an 'Invalid' message and the 'Invalid' message bounces back or is auto–replied.

As an 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 --"
```

Attention: If a message response matches more than one string in the tag file, it gets tagged with the status of the first string it matches in the file. That is, the Notification Mailer performs a top to bottom comparison against the tag file. Due to this behavior, you should prioritize your strings listing the ERROR tags first, followed by the UNAVAIL and then IGNORE tags.

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.

See: Defining Rules for Automatic Notification Processing: page 10 – 24

▶ **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 Internet Messaging 4.2 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.

Step 11 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 message templates in the System: Mailer item type.

Context: You need to perform this step only once.

Workflow Open Mail (Templated) Message

If you select the templated response method, the Notification system uses the Workflow Open Mail (Templated) message as a template for E–mail notifications that require a response. The notification 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, and any comments from the sender of the message or, if the notification is forwarded from another user, any comments from the forwarder.

> **Note:** To select the templated response method, set DIRECT_RESPONSE=N in the configuration file for the Notification Mailer. See: To Create a Configuration File for the Notification Mailer: page 2 – 47.

The response instructions in the plain text message body describe how to reply manually using the templated response method. This message is used for notifications sent to performers with a notification preference of MAILTEXT or MAILATTH. The response instructions in the HTML–formatted message body describe how to reply using the automatically generated response template. This message is used for notifications sent to performers with a notification preference of MAILHTML, and is also attached to notifications sent to performers with a notification preference of MAILATTH.

The Workflow Open Mail (Templated) 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.

MAILTO The content of the HTML tag that a recipient would click on to respond to a notification. This attribute is used only for HTML E–mail notifications.

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

The boilerplate text for a plain text message body is as follows:

```
Oracle Workflow Notification
&COMMENT

_____________Start of Response Template_____________

Response Template for &NOTIFICATION

To submit your response, reply to this message, including
this response template with your reply.  Copy and paste from
this message if necessary to obtain an editable copy of the
template. Insert your response value between the quotes
following each response prompt.
```

```
&RESPONSE
_____________End of Response Template_____________

Notification Details:
&BODY

Due Date: &DUE_DATE
```

The boilerplate text for a HTML–formatted message body is as follows:

```
<HTML> <HEAD> <TITLE> Oracle Workflow Notification </TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" >
<P><B><FONT SIZE=+1>&COMMENT</FONT> </B>
<P>&BODY
<P><B>Please click on one of the following choices to
automatically generate an E-mail response.  Before sending
the E-mail response to close this notification, ensure all
response prompts include a desired response value within
double quotes.</B>
<P>&MAILTO
</BODY>
</HTML>
```

Workflow Open Mail (Direct) Message

If you select the direct response method, the Notification system uses the Workflow Open Mail (Direct) message as a template for E–mail notifications that require a response. The notification 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, and any comments from the sender of the message or, if the notification is forwarded from another user, any comments from the forwarder.

> **Note:** To select the direct response method, set DIRECT_RESPONSE=Y in the configuration file for the Notification Mailer. See: To Create a Configuration File for the Notification Mailer: page 2 – 47.

The response instructions in the plain text message body describe how to reply using the direct response method. This message is used for notifications sent to performers with a notification preference of MAILTEXT or MAILATTH. The response instructions in the

HTML–formatted message body describe how to reply using the automatically generated response template. This message is used for notifications sent to performers with a notification preference of MAILHTML, and is also attached to notifications sent to performers with a notification preference of MAILATTH.

> **Note:** Responses that are generated automatically from an HTML–formatted notification or attachment always use a response template, regardless of which response method you select in the DIRECT_RESPONSE parameter.

The Workflow Open Mail (Direct) 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.
MAILTO	The content of the HTML tag that a recipient would click on to respond to a notification. This attribute is used only for HTML E–mail notifications.

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

The boilerplate text for a plain text message body is as follows:

```
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
```

The boilerplate text for a HTML–formatted message body is as follows:

```
<HTML> <HEAD> <TITLE> Oracle Workflow Notification </TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" >
<P><B><FONT SIZE=+1>&COMMENT</FONT> </B>
<P>&BODY
<P><B>Please click on one of the following choices to
automatically generate an E-mail response.  Before sending
the E-mail response to close this notification, ensure all
response prompts include a desired response value within
double quotes.</B>
<P>&MAILTO
</BODY>
</HTML>
```

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 FYI Mail template, where attributes preceded by an ampersand (&) are token substituted with runtime values when the notification is sent. The boilerplate text for the plain text message body is as follows:

```
Oracle Workflow Notification (FYI)
&COMMENT

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

The boilerplate text for the HTML–formatted message body is as follows:

```
<HTML><HEAD></HEAD>
<BODY BGCOLOR="#FFFFFF"><b>Oracle Workflow Notification
(FYI)</b>
<br>&COMMENT
<hr>
<P>&BODY
</BODY>
</HTML>
```

Workflow Canceled Mail Message

The Workflow Canceled Mail message informs the recipient that a previously sent notification is canceled. It has the following message attributes, with values that are drawn from the message definition associated with the canceled 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 boilerplate text for the plain text message body is as follows:

```
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
```

The boilerplate text for the HTML–formatted message body is as follows:

```
<html><Head></Head><body>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.
<hr>
&BODY
</body></html>
```

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 boilerplate text for the plain text message body is as follows:

```
Oracle Workflow Notification
&COMMENT

WARNING: 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
```

The boilerplate text for the HTML–formatted message body is as follows:

```
<html><Head></Head><body>WARNING: Your previous response to
this message was invalid (see error message below).  Please
resubmit your response.
<P>Error Message: &MAIL_ERROR_MESSAGE
<BR>Error Stack: &MAIL_ERROR_STACK
<HR><P><B><FONT SIZE=+1>&COMMENT</FONT> </B>
<P>&BODY
<P><B>Please click on one of the following choices to
automatically generate an E-mail response.  Before sending
the E-mail response to close this notification, ensure all
response prompts include a desired response value within
double quotes.</B>
<P>&MAILTO
</BODY> </HTML>
```

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 boilerplate text for the plain text message body is as follows:

```
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
```

The boilerplate text for the HTML–formatted message body is as follows:

```
<html><Head></Head><body>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.
<hr>
&BODY
</body></html>
```

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:

SUMMARY Summary report.

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

SYSDATE The current date.

The boilerplate text for the plain text message body is as follows:

```
Summary of Notifications for '&USER_NAME'
(Please use the Notification Viewer to see details or
respond.)
-------------------------------------------------------------

&SUMMARY
```

The boilerplate text for the HTML–formatted message body is as follows:

```
<HTML><HEAD></HEAD><BODY>
<P><FONT size=+1>Summary of Notifications for
'&USER_NAME'</FONT>
<BR><i>Please use the Notification Viewer to see details or
respond.</i>
<HR>

&SUMMARY
</BODY>
</HTML>
```

Workflow Warning Mail Message

The Notification system uses the Workflow Warning Mail message as a template to send a message to a user if it receives unsolicited mail from that user. It has the following message attributes, with values that are drawn from the unsolicited mail:

UBODY The text of the unsolicited mail message body.

USUBJECT The text of the unsolicited mail subject line.

UFROM The address of the user that sent the unsolicited mail.

The boilerplate text for the plain text message body is as follows:

```
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
```

The boilerplate text for the HTML–formatted message body is as follows:

```
<html><head></head><body>
<b>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
auto-generated reply created when responding to the original
message. This contains the 'NID' line which is necessary for
identification.  If you are not responding to a
notification, please do not send mail to this account.</b>
<hr>
<P>From: &UFROM
<BR>Subject: &USUBJECT

<P>&UBODY
</body></html>
```

Step 12 Customizing the Logo on Oracle Workflow's Web Pages

To use Oracle Workflow's web pages and the Workflow Monitor at your site, you must have Oracle WebDB or Oracle Application Server installed. Refer to your web server documentation for additional information.

Once your web server is installed and set up, you can customize the company logo that appears on Oracle Workflow's web pages.

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 or higher and Abstract Windowing Toolkit (AWT) to connect to the Workflow Monitor.

▶ To Customize Oracle Workflow's Web Pages

You can customize the company logo that appears in the upper right corner of Oracle Workflow's web pages.

1. Copy or rename your company logo file (in .gif format) to **WFLOGO.gif.**
2. Move the file to the physical directory that your web server's `/OA_MEDIA/` virtual directory points to.

   Note: If you are using Oracle Workflow embedded in Oracle Applications, the mapping of `/OA_MEDIA/` is completed as part of the Oracle Applications installation and setup steps.

   Note: If you are using the standalone version of Oracle Workflow, the mapping of `/OA_MEDIA/` is completed after you install the Oracle Workflow server and you set up the Workflow Monitor.

Context: You need to perform this step only once.

Step 13 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 Wf subdirectory within your ORACLE_HOME for Windows 95, 98, 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:

  `/OA_JAVA/oracle/apps/fnd/wf/icons`

  Note: `/OA_JAVA/` is a virtual directory mapping that you defined in your web server when you installed Oracle Workflow.

Context: You need to perform this step only once.

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 a 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 to download workflow process definitions from the database to a file, check the value for the environment variable WF_ACCESS_LEVEL on your workflow server. See: Using the Workflow Definitions Loader: page 2 – 76.

  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 version of Oracle Workflow embedded in Oracle Applications, 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 – 2 (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, Windows 98, 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 l*evel.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 Windows 95, 98 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 Help 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.

When you upload or upgrade a process definition, the Workflow Definitions Loader automatically validates the process definition to ensure that it conforms to specific process design rules. It performs the same validation as the Oracle Workflow Builder Verify feature. See: To Validate a Process Definition: page 5 – 15.

Attention: When you upload or upgrade a workflow definition onto an existing definition in a database, it is possible that an object in the upload/upgrade definition has a Display Name that is already in use by a different object in the target database. If this occurs, the Workflow Definition Loader automatically resolves the display name conflict by adding a '@' character to the beginning of conflicting display names in the target database. The upload/upgrade definition is then applied as is and a warning message is generated.

▶ **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 Oracle Net8 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.

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

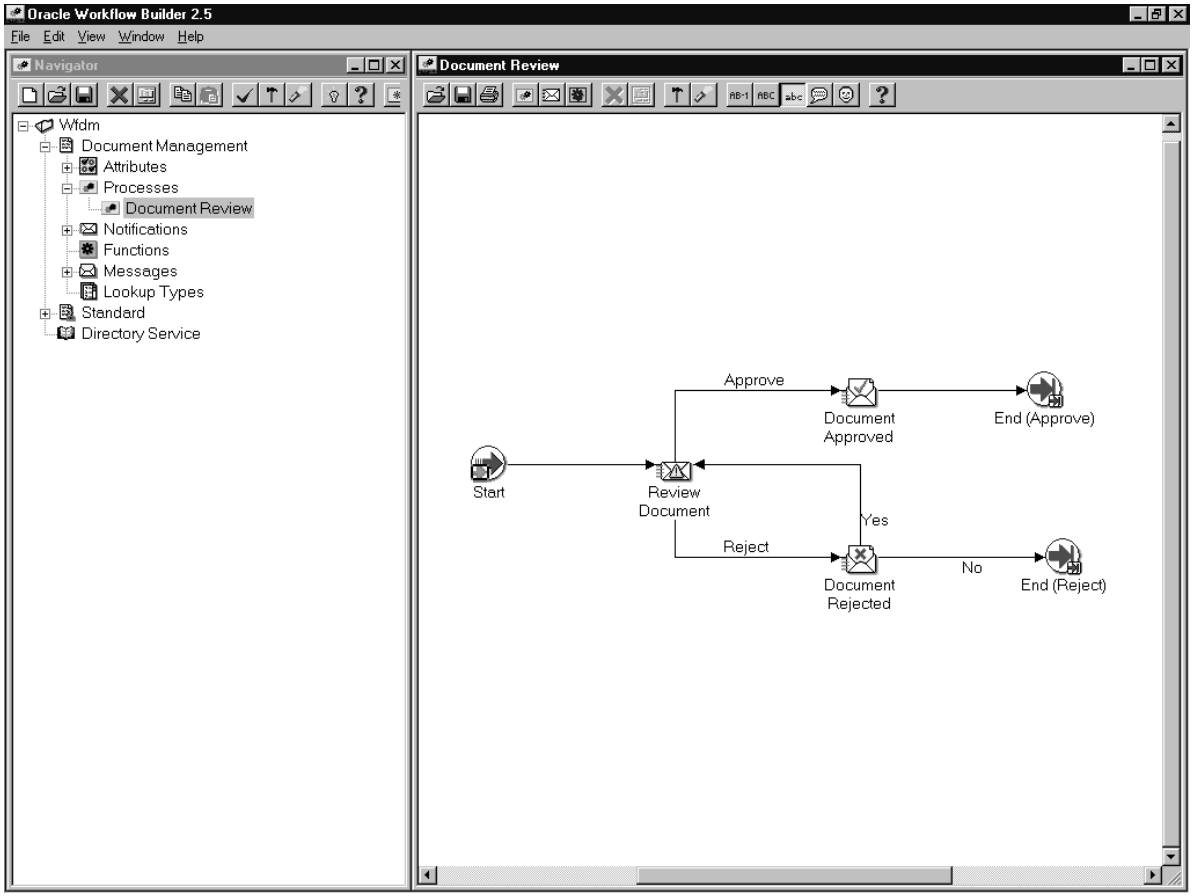
3

Defining a Workflow Process

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

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 – 7.



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

The Navigator Tree Structure

The Navigator window displays a navigator tree hierarchy for each data store that you open or load into Oracle Workflow Builder. A data store (primary branch) is a database connection or flat file that holds your workflow process definition. Within each data store there is at least one item type heading (secondary branch) that represents the grouping of a particular set of processes and its component objects. The following 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 4 – 2.
- Processes—lists the process activities or workflow process definitions for the current item type. See: Process Window: page 5 – 2 and Activities: page 4 – 37.
- 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 4 – 37.
- 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 4 – 37.
- 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 4 – 22.
- Lookup Types—lists the lookup types associated with the current item type. A lookup type has one or more values called lookup codes associated with it. A lookup type is a list of values that can be referenced by a message, or by a notification, function, or process as its possible result type. See: Lookup Types: page 4 – 18.

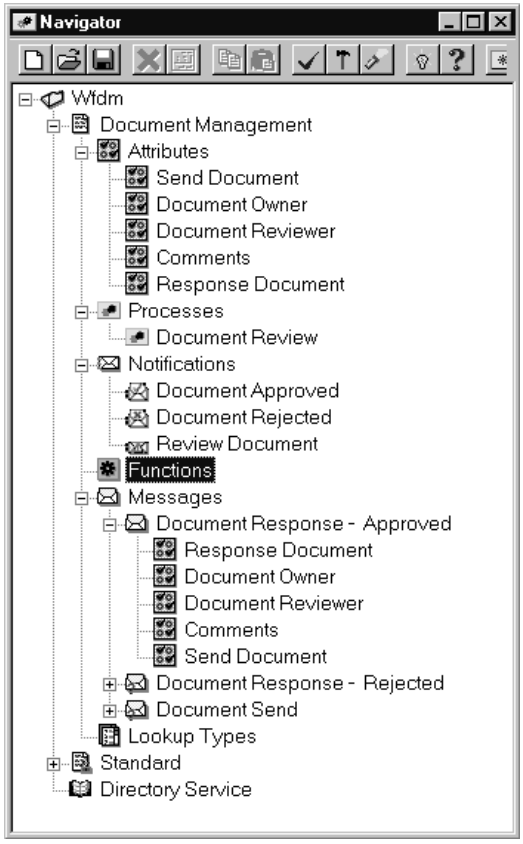
  Note: Each data store also contains a Directory Service branch. The Directory Service branch lists all the directory service roles that you load from your Oracle Workflow database. See: Roles: page 5 – 19.

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

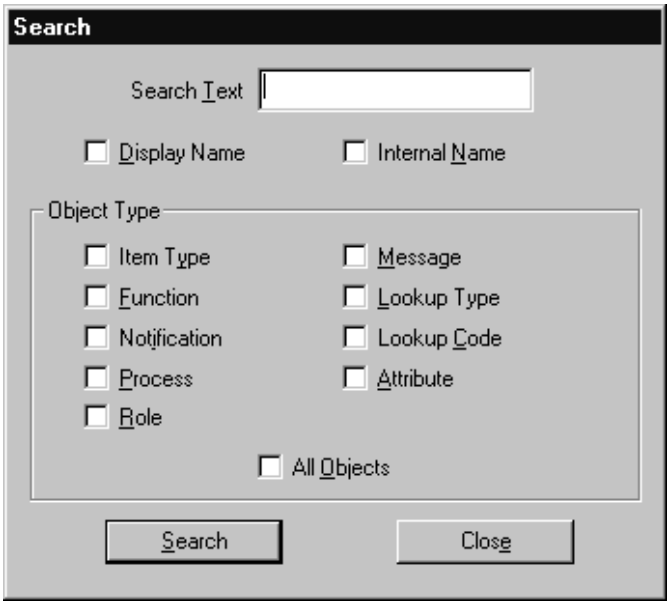
Viewing the Navigator Tree

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

The Navigator window also contains a toolbar that you can use to perform actions within the Navigator window. See: Navigator Toolbar: page A – 7.


Navigator
Wfdm
Document Management
Attributes
Send Document
Document Owner
Document Reviewer
Comments
Response Document
Processes
Document Review
Notifications
Document Approved
Document Rejected
Review Document
Functions
Messages
Document Response - Approved
Response Document
Document Owner
Document Reviewer
Comments
Send Document
Document Response - Rejected
Document Send
Lookup Types
Standard
Directory Service

▶ To Find an Object in the Navigator Tree



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

Creating Process Definitions in Oracle Workflow Builder

Before using Oracle Workflow Builder, you should plan what your process needs to accomplish. In particular, determine what activities need to occur, the order of the activities, what results dictate the different branches of the process, who needs to be informed and what they need to know. Oracle Workflow provides several demonstration workflow examples. See: Sample Workflow Processes: page 13 – 2.

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

- Top–Down Design—If you prefer to approach your design from a high level, you can first sketching out the process diagram with activities, then go back later to create the supporting objects for each activity. See: To Create a Process Definition from Top–Down: page 3 – 10.
- Bottom–Up Design—If you prefer to take a more programmatic approach to your design, you can first define each of the supporting objects of your process before attempting to create a higher level process diagram. See: To Create a Process Definition From Bottom–Up: page 3 – 8.

Quick Start Wizard

The Quick Start Wizard helps you build a process definition from scratch using a process definition template. The Quick Start Wizard creates a new item type for your process, prompting you for the minimum required information. It then creates an outline process diagram from which you can flesh out with more activities. Once the Quick Start Wizard sets up the template, you can use either the top–down or bottom–up approach to complete the design. See: To Use the Quick Start Wizard: page 3 – 18.

Versioning and Dates of Effectivity

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

process itself is an activity, so a process definition always remains constant until the process instance completes.

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

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

Using the Edit Button in a Property Page

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

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

▶ To Create a Process Definition From Bottom Up

1. To start Oracle Workflow Builder, double–click 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 of the Start menu.
2. Choose New from the File menu to create a workspace for your new process definition.

Suggestion: Alternatively, you can use the Quick Start Wizard to first create the framework for your new process definition. Once the Quick Start Wizard creates your new item type and new process activity, you can skip to step 4 below to begin defining the supporting objects for the new item type and process activity. See: To Use the Quick Start Wizard: page 3 – 18.

3. Create a new item type. The item type classifies the work item to be managed by the process. See: To Create an Item Type: page 4 – 6.

4. You can define item type attributes to fully describe your item type and have the activities in your process refer to these attributes for information. See: To Define an Item Type or Activity Attribute: page 4 – 8.

5. Create new lookup types. See: To Create Lookup Types: page 4 – 19.

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

6. Create new messages. See: To Create a Message: page 4 – 24.

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

   You can also create message attributes for the message. You can incorporate message attributes of type 'Send' into a message that are token substituted at runtime to provide dynamic content. You can also define message attributes of type 'Respond' to prompt the notification recipient for a response. See: To Define a Message Attribute: page 4 – 29.

7. Create a new process activity, notification activity or function activity. You may also use predefined standard activities associated with the Standard item type. See: Activities: page 4 – 37 and Standard Activities: 6 – 2.

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

8. Diagram the process.

   Display the Process window for your process activity to diagram the activities and transitions that define your workflow process. You can drag activities from the navigator tree into the Process window. See: Diagramming a Process: page 5 – 4.

9. Save your work by choosing Save or Save As from the File menu. See: To Save Your Work: page 3 – 15.

10. In a database accessible by your Oracle Workflow server, create the PL/SQL stored procedures called by your PL/SQL function activities. You can do this through SQL*PLUS or the Oracle Procedure Builder. See: Workflow APIs: page 8 – 3 and Standard API for PL/SQL Procedures Called by Function Activities: page 7 – 2.

See Also

To Modify a Process Definition: page 3 – 11

Deleting Objects in Oracle Workflow Builder: page 4 – 55

Item Type Definition Web Page: page 3 – 21

▶ To Create a Process Definition from Top Down

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 of the Start menu.

2. Use the Quick Start Wizard to create the framework for your new process definition. Specify the requested information for the new item type and new process activity. See: To Use the Quick Start Wizard: page 3 – 18.

3. A Process window appears, that shows a Start and an End activity node. Create your process diagram by defining new activity nodes to place between the Start and End nodes. See: To Define Nodes in a Process: page 5 – 7.

   You may also use predefined standard activities associated with the Standard item type. See: Standard Activities: 6 – 2.

4. Model your process by drawing transitions between your activities. See: Diagramming a Process: page 5 – 4.

5. Save your work by choosing Save or Save As from the File menu. See: To Save Your Work: page 3 – 15.

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

See Also

To Modify a Process Definition: page 3 – 11

Deleting Objects in Oracle Workflow Builder: page 4 – 55

Item Type Definition Web Page: page 3 – 21

▶ To 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 95/98 program group. If you are using Windows 95/98 or NT 4.0, you can also select the Oracle Workflow Builder icon from the appropriate program folder of the Start menu.
2. Choose Open from the File menu to open a connection to the database or file that contains the process definition you want to modify. See: To Access Process Definitions in an Existing Data Store: page 3 – 12.
3. Select and expand the existing item type associated with the process definition you want to modify.
4. You can modify an item type, item type attribute, lookup, message, message attribute, process activity, notification activity, function activity, or activity attribute. See: To Create an Item Type: page 4 – 6, To Define an Item Type or Activity Attribute: page 4 – 8, To Create Lookup Types: page 4 – 19, To Create a Message: page 4 – 24, To Define a Message Attribute: page 4 – 29, or Activities: page 4 – 37.

5. You can also modify the process diagram by displaying the Process window for your process activity. See: Diagramming a Process: page 5 – 4.
6. Save your work by choosing Save or Save As from the File menu. See: To Save Your Work: page 3 – 15.

See Also

Deleting Objects in Oracle Workflow Builder: page 4 – 55

Item Type Definition Web Page: page 3 – 21

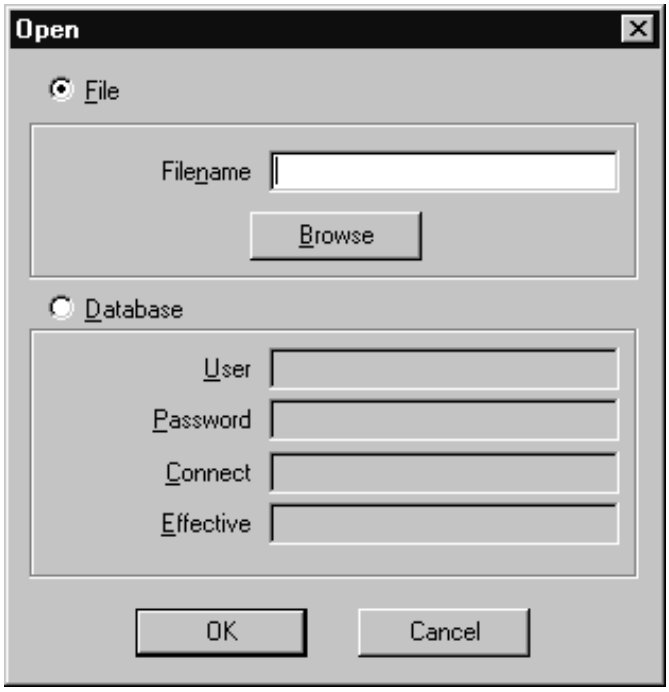
Opening and Saving Item Types

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.

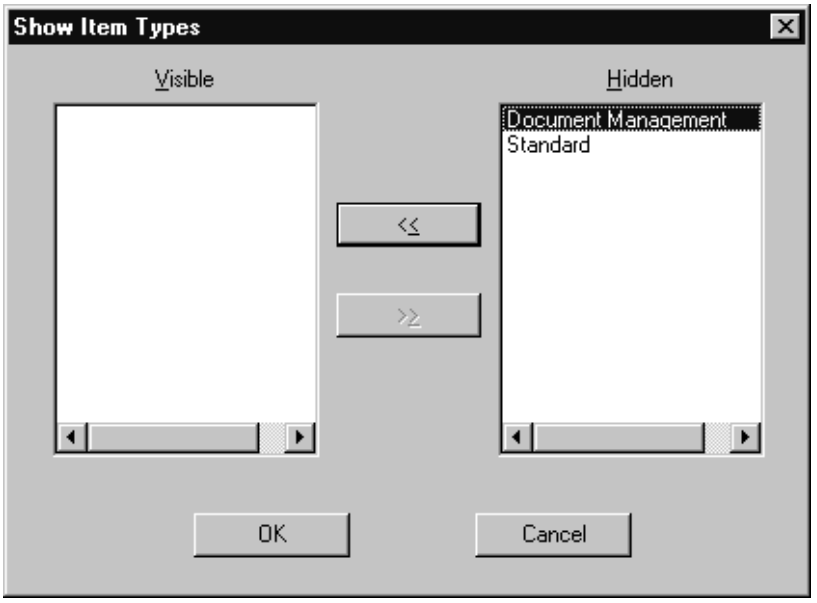


2. Select database or file to connect to the source containing the item type to which your process definition is associated.
3. To open a File: Provide the complete file path and choose OK, or use Browse to locate and open the file (extension .wft).

   Note: You can also drag and drop a .wft file from the Microsoft Windows 95/NT 4.0 Explorer or Microsoft Windows NT File Manager into the navigator tree to open that file in Oracle Workflow Builder.

   Note: When you use Browse to find and open a file, the current directory that you open the file from becomes the new default directory from which you open files in the future. This default directory persists until you use Browse again to locate another file.
4. To open a Database connection: Enter the username and password for the database. Enter the name of the database alias or connect string and choose OK.
5. If you wish to retrieve a process definition that was effective at a particular point in time, you can specify a date and time in the Effective field and have Oracle Workflow Builder retrieve that data from the database. The format that you should use to specify the

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



6. If multiple item types exist in the data store, the Show Item Types window appears. Select from the Hidden list, the item type(s) you want to view, and choose << to move it into the Visible list. Choose OK to load these item types into the navigator tree.
7. If at any time you want to view and modify item types that are hidden in the current data store, you can double–click on the Hidden Item Types branch in the navigator tree to display the Show Item Types window and select the item types you want to show. You can also choose Show/Hide Item Types from the File menu to display the Show Item Types window.

   Note: You can copy item types from one store to another in any order even if the item types reference each other. However, you may get validation errors due to foreign key references. Pay attention to these errors as they may indicate that you need to also copy other item types into the new store to resolve the foreign key references. The final process definition in the new store will be valid as long as all referenced item types are copied to the new destination store.
8. When you finish working, choose Save from the File menu to preserve your changes and make them effective immediately. See: To Save Your Work: page 3 – 15.

See Also

To Start Oracle Workflow Builder from the MS–DOS Prompt: page 3 – 17

▶ **To Save Your Work**

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

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

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, as well as any previously customized objects. If you uncheck this check box, Oracle Workflow Builder runs in 'upgrade' mode and will only save edits to protected objects that you have access to change and will not overwrite objects that have been previously customized. These two modes match the upgrade and upload behavior of the Workflow Definitions Loader program. As the default, the check box is unchecked. See: To Set the Access Level for an Object: page 4 – 17 and Using the Workflow Definitions Loader: page 2 – 76.

2. If you want to save your work to a different data store (database or flat file), or if you want to save it to a database with an effective date other than the current system date, then choose Save As... from the File menu. Use the Save As window to specify the file or database you want to save your process definition to, and the date when you want your process definition to take effect in the database. You can leave the Effective field blank to save and make the changes effective immediately. See: Version/Effective Date: page 8 – 9.

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

3. Note that when you save your work, Oracle Workflow automatically validates the process definition for any invalid or missing information and displays what it finds in a Workflow Error verification window. You can either correct the information before saving your work, or go ahead and save your work as is, and fix the problems later. Use the Copy button to copy the information from the Workflow Error window to the clipboard for later reference. If you save your work without correcting the problems,

the Workflow Error window will reappear when you reopen your process definition.

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

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 or above, 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:

```
wfbldr20 wfdemo.wft -E WFDEMO:NOTIFYAPPROVER
```

See Also

Using the Workflow Definitions Loader: page 2 – 76

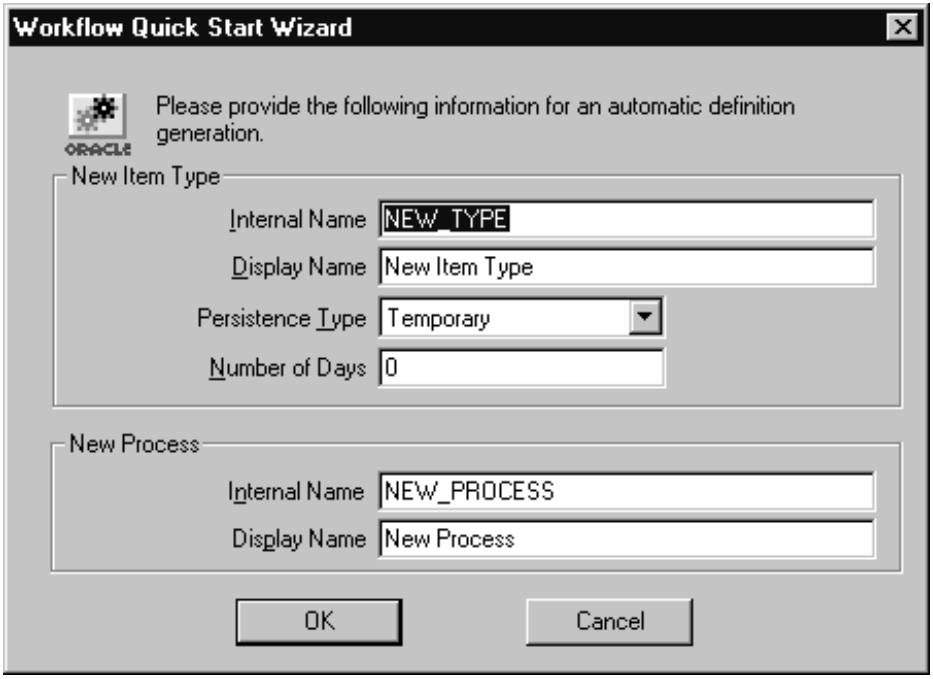
Creating a Shortcut to a Workflow Process: page 5 – 17

Quick Start Wizard Overview

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

▶ **To Use the Quick Start Wizard**

1. Select Quick Start Wizard from the File menu.



2. The Workflow Quick Start Wizard window prompts you for the following mandatory information:

   - New Item Type
     - Internal Name—Specify an all uppercase internal name with a maximum of eight characters. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying an item type.

Attention: To update the internal name of an item type once it is defined, you must use a special SQL script. See: Wfchitt.sql: page 14 – 6.

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

- Display Name—Enter a translatable Display Name for the item type.
- Persistence Type—Specify Temporary or Permanent persistence for the status audit trail of the item type.
- Days—If Persistence Type is Temporary, specify the number of days from the time an item type instance completes

before its status audit trail can be purged. See: Persistence Type: page 4 – 4.

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

Attention: To update the internal name of an activity once it is defined, you must use a special SQL script. See: Wfchact.sql: page 14 – 5.

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

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

3. The Quick Start Wizard does the following:
   - Creates a new data store called "Untitled–*n*" in the Navigator window.
   - Uses the information you entered in the Workflow Quick Start Wizard window to create a new item type and process activity in the data store.
   - Loads the Standard item type into the new data store so that you can include standard activities in the process you create.
   - Opens the Process window for the new process activity you defined. The Process window displays a Start and an End activity.
4. You can now customize your process definition in one of two ways:
   - Take a bottom–up design approach by first creating activities and all their supporting objects before trying to draw a workflow diagram. See: To Create a Process Definition From Bottom–Up: page 3 – 8.
   - Take a top–down design approach by creating activities that contain minimum information so you can draw the workflow diagram first. You can go back later to fill in the details of each activity and its supporting objects. See: To Create a Process Definition from Top–Down: page 3 – 10.

Item Type Definition Web Page

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

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

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

▶ To Query an Item Type

1. Enter the following URL in your web browser:

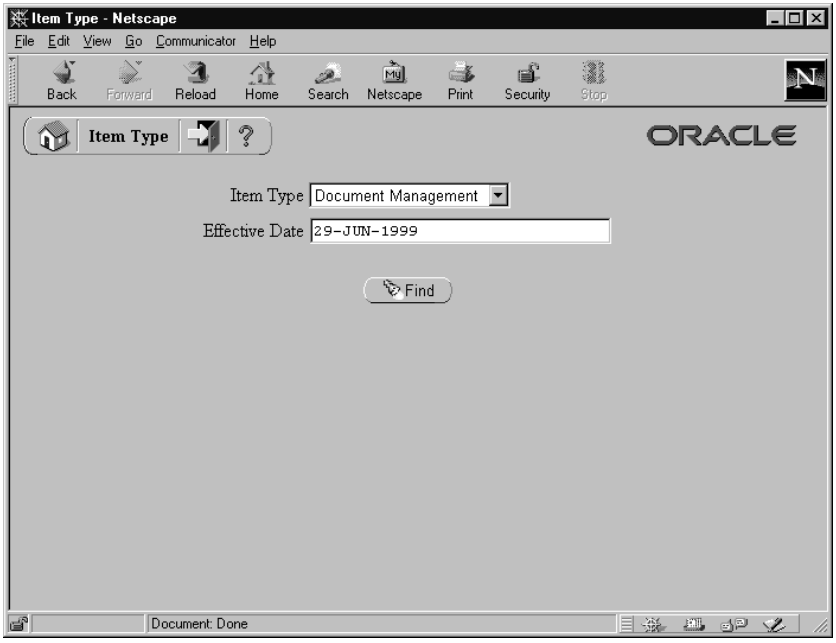
   `<webagent>/wf_item_definition.find_item_type`

   Replace the bracketed italicized text as follows:

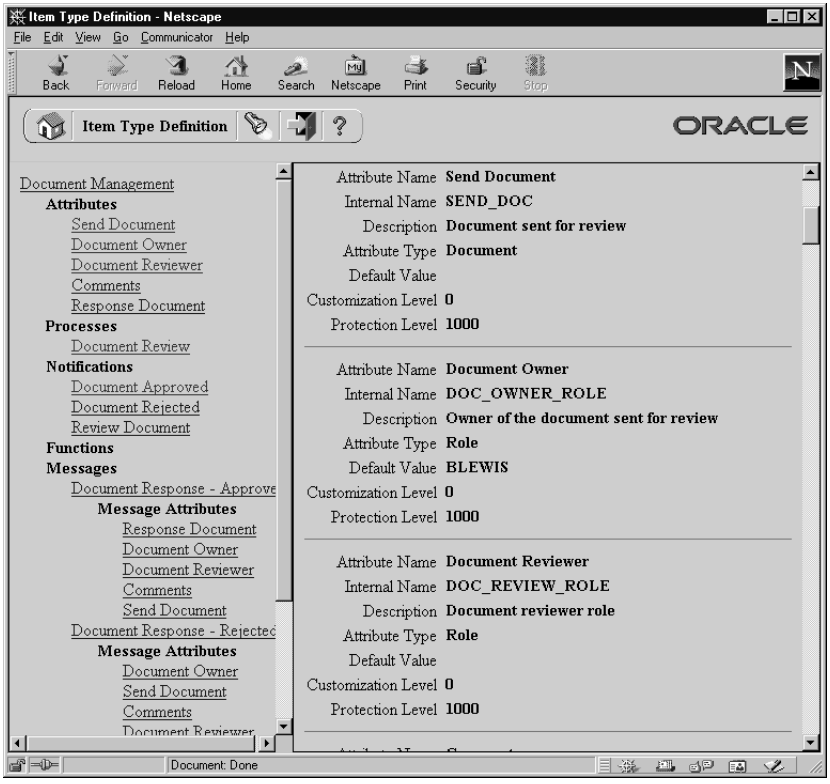
   <webagent> represents the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 12.

   Note: You can also access the Find Item Type web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 9 – 2.

   Note: If you are accessing this URL for the first time in your web browser session, Oracle WebServer prompts you for a valid username and password to log on, as this URL is protected.



2. Use the Item Type poplist field to select an item type.
3. Specify the effective date and time of the item type definition you want to display using the format specified in the Date Format field of your User Preferences web page. See: Setting User Preferences: page 9 – 4.
4. Choose Find to display the item type in the Item Type Definition web page.



▶ To Review an Item Type Definition

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

CHAPTER

4

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.

Workflow Process Components

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

- Item Types
- Lookup Types
- Messages
- Activities
- Attributes
- Roles

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, 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 4 – 6.

Item Type Attributes

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

Applications as well as function activities can reference and set item type attributes using the Oracle Workflow Engine APIs. You can define and maintain as many item type attributes as necessary for an item type. You should define as an item type attribute, any information that will be required by an activity in your process, or any information that

will need to be sent in a notification message. See: To Define a Message Attribute: page .

Attribute Types

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

- Text—The attribute value is a string of text.
- 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—The attribute value is an Oracle Applications internal form function name and its optional form function parameters. This attribute type is not relevant for the standalone version of Oracle Workflow.

  If you include a form–type attribute in a notification message as a message attribute, the notification, when viewed from the Notification Details web page, displays an attached form icon that lets users drill down to the referenced form. See: Overview of Menus and Function Security, *Oracle Applications Developer's Guide*.
- URL—The attribute value is a Universal Resource Locator (URL) to a network location. If you reference a URL attribute in a notification message as a message attribute, the notification, when viewed from the Notification Details web page or as an HTML–formatted E–mail, displays an anchor to the URL specified by the URL attribute. The user can complete an activity or see additional information related to the activity by accessing that URL.
- Document—The attribute value is an attached document. You can specify the following types of documents in the default value field:
  - PL/SQL document—a document representing data from the database, generated from a PL/SQL procedure.
  - DM document—a document managed by an external document management system.

- Role—The attribute value is the internal name of a role. If a message attribute of type role is included in a notification message, the attribute automatically resolves to the role's display name, eliminating the need for you to maintain separate attributes for the role's internal and display names. Also when you view the notification from a web browser, the role display name is a hypertext link to the email address for that role. To set a default value for the attribute, you must initially load roles from the database. See: Roles: page 5 – 19.
- Attribute—The attribute value is the internal name of another existing item type attribute that you want to maintain references to in a process.

Persistence Type

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

If you set an item type's Persistence to Temporary, you must also specify the number of days of persistence. The status audit trail for each instance of a Temporary item type is maintained for at least '*n*' days of persistence after its completion date. After the '*n*' days of persistence, you can then use any of the WF_PURGE APIs to purge the item type's runtime status information. See: WF_PURGE: page 8 – 68.

> **Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications, you may also use the Purge Obsolete Workflow Runtime Data concurrent program to purge obsolete item type runtime status information. The executable name for this concurrent program is "Oracle Workflow Purge Obsolete Data" and its short name is FNDWFPR. See: Purge Obsolete Workflow Runtime Data: page 8 – 76.

Attention: Since Persistence Type is a feature that is introduced in Oracle Workflow Release 2.5, any item type defined in a release prior to 2.5 is automatically upgraded to have a Persistence Type of Temporary and 0 Days.

Item Type Selector Function

If your item type has or will have more than one runnable process activity associated with it, define a PL/SQL function that determines

which process activity to run in a particular situation. For example, you may have two different requisition approval process activities associated with the same item type. The process that Oracle Workflow executes may vary depending on how and where the requisition originates. Your selector function would determine which process would be appropriate in any given situation.

You can also extend the Selector function to be a general callback function so that item type context information can be reset as needed if the SQL session is interrupted during the execution of a process. This is particularly important in the Oracle Applications scenario when you view a notification from the Notification Details web page and attempt to launch another form that is associated with the notification. Oracle Workflow calls the selector/callback function for your item type in 'TEST_CTX' mode to test the Oracle Applications context before turning the form launch over to the Oracle Application Object Library function security system. In 'TEST_CTX' mode, the selector/callback function can perform whatever logic necessary to determine whether it is appropriate to launch the form. See: Standard API for an Item Type Selector or Callback Function: page 7 – 7.

External Document Integration

Documents have an enormous impact in the operations of an organization. With the explosion of digital media and the worldwide web, electronic documents of a wide variety of formats, including non–printed media, are forcing organizations to address the management of these documents. The value of information in these documents can be maintained only if the documents can be managed and shared. Document management systems offer organizations the ability to share, manage, and reuse important digital documents in a single shared resource.

By integrating a document management system with a workflow system, a managed document can be directed to appropriate individuals at appropriate times, optimizing the impact of the document's information. Oracle Workflow provides seamless open integration with Oracle's third party document management (DM) integration partners. In a workflow process, you can attach documents stored in a document management system, which we call DM documents or documents generated by a PL/SQL procedure, which we call PL/SQL documents. You attach a document to a workflow process by referencing the document in a predefined item attribute or message attribute of type Document. See: Attribute Types: page 4 – 3, To Define an Item Type or Activity Attribute: page 4 – 8 and To Define a Message Attribute: page 4 – 29.

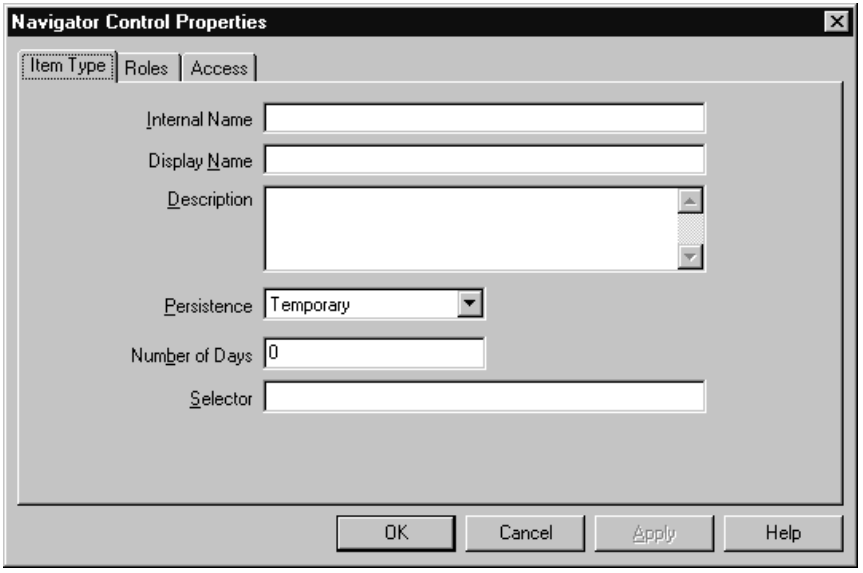
For PL/SQL documents, the item or message attribute's value would be the name of the PL/SQL package and procedure used to generate the document. The PL/SQL procedure must follow an Oracle Workflow standard interface. The document generated by the PL/SQL procedure is simply displayed within the text of a notification. See: Standard API for a "PL/SQL" Document: page 7 – 11.

For DM documents, the item or message attribute's value would be the prefix DM, followed by the node ID of the document management system, and the document ID, as assigned by the document management system. The node ID is assigned when you initially identify your document management system in the Document Nodes web page. See: Defining Document Management Repositories: page 2 – 31.

See Also

Document Management Integration in Notifications: page 10 – 31

▶ 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: To update the internal name for an item type once it is defined, you must use a special SQL script. See: Wfchitt.sql: page 14 – 6.

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

3. Enter a translatable Display Name that is longer and more descriptive. You can also supply a description for the item type.

4. Specify a persistence type of Temporary or Permanent. If you set the persistence type to Temporary, then specify the number of days from the time the item instance completes before its status audit trail can be purged. See: Persistence Type: page 4 – 4.

5. If your item type has or will have more than one workflow process associated with it, you may specify a selector function using the syntax <*package_name*>.<*procedure_name*>. The selector function is a PL/SQL stored procedure that automatically identifies the specific process definition the Workflow Engine should execute when a workflow is initiated for this item type. You can also extend the selector function to be a general callback function that resets context information each time the Workflow Engine establishes a new database session to execute activities. See: Standard API for an Item Type Selector or Callback Function: page 7 – 7.

6. Choose Apply to save your changes.

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

8. Select the Access tab page to set the access and customization levels for this item type. See: Allowing Access to an Object: page 4 – 16.

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

10. A secondary branch appears in the navigator tree that represents the item type you just created. You can review or edit the properties of this item type at any time by double–clicking on the

item type in the navigator tree or by selecting the item type and choosing Properties from the Edit menu.

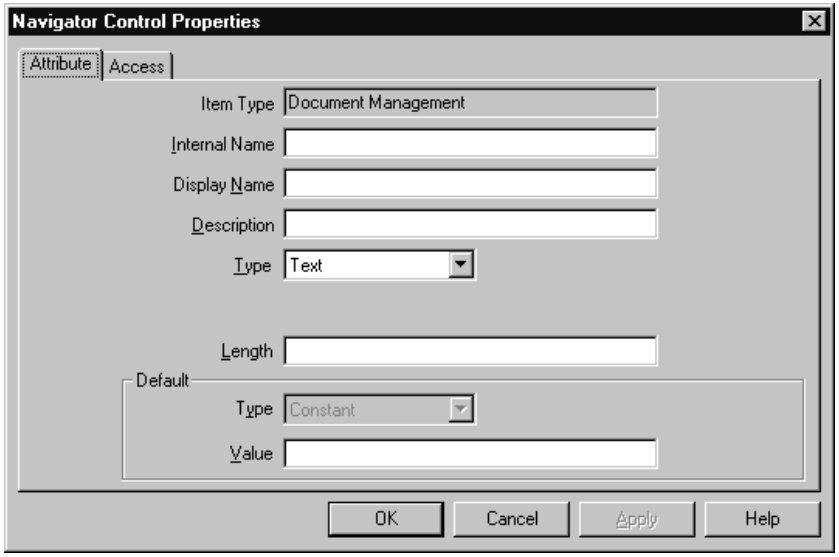
11. Define as many item type attributes as necessary to use as global variables in your process. You use these item type attributes to pass values to and from your function and notification activities. See: To Define an Item Type or Activity Attribute: page 4 – 8.

See Also

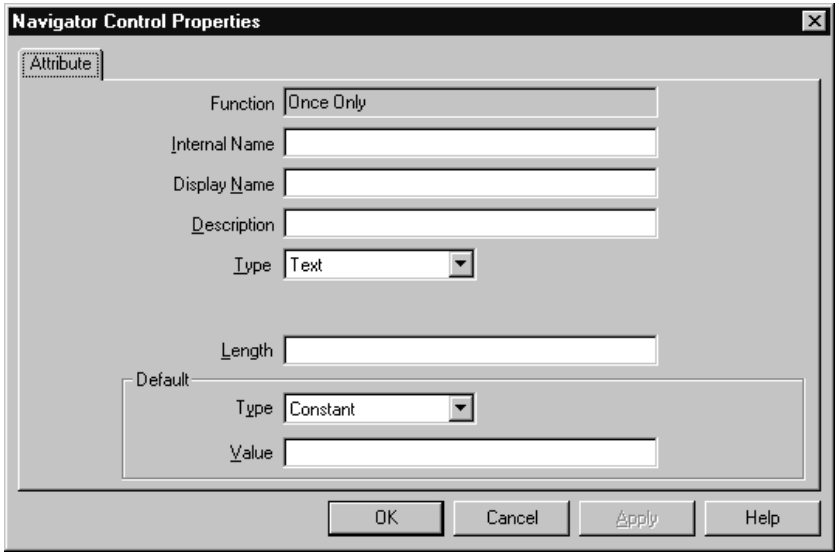
Using the Edit Button in a Property Page: page 3 – 8

▶ To Define an Item Type or Activity Attribute

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



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 leading/trailing 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.

Attention: To update the internal name for an attribute once it is defined, you must use a special SQL script. See: Wfchita.sql: page 14 – 6 and Wfchacta.sql: page 14 – 6.

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

3. Enter a Display Name. This is the name that appears in the navigator tree.
4. Enter an optional description.
5. Select the data type of the attribute. Form, URL, and document data types are not relevant if you are defining an activity attribute.
6. Depending on the data type of your attribute, provide the following default value information:
   - **Text**—Specify the maximum length of the text attribute and an optional default text string.
   - **Number**—Optionally provide a format mask for your number and a default value.

- **Date**—Optionally supply a format mask for the date and a default value.
- **Lookup**—Choose a predefined Lookup Type from which to draw values. Choose a lookup code from that lookup type for the default value.
- **URL**—Specify an optional Universal Resource Locator (URL) to a network location in the Default Value field and specify the frame target for the URL. See: To Define a URL Attribute: page 4 – 11.

  Note: The Frame Target field is applicable only for message attributes of type URL. It is not used for item type attributes or activity attributes.
- **Form**—This attribute is relevant only for the version of Oracle Workflow embedded in Oracle Applications.

  Specify an optional developer form function name and optional argument strings (form function parameters) in the Default Value field. See: Overview of Menus and Function Security, *Oracle Applications Developer's Guide* and To Define a Form Attribute: page 4 – 12.
- **Document**—Enter an optional string that identifies the document in the default value field. See: To Define a Document Attribute: page 4 – 13.

  Note: The Frame Target field is applicable only for message attributes of type DM Document. It is not used for item type attributes or activity attributes.
- **Role**—Specify a role name. See: Roles: page 5 – 19.
- **Attribute**—Specify the name of an item type attribute that you want to maintain references to in a process by choosing from the list of existing item type attributes.

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

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

associated with. The item type attribute you select must also be of the same data type as the activity attribute.

Note: An activity attribute type of 'Text' is compatible with any item attribute type, 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 4 – 16.
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 – 8

▶ To Define a URL Attribute

1. Specify a Universal Resource Locator (URL) to a network location in the Default Value field of the Attribute property page. The URL can be a constant or a value returned from another item attribute.
2. You can include argument strings in your URL that are text strings or that are token substituted with other item type attributes. See: To Token Substitute an Attribute: page 4 – 36.

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

`–&item_attr–`

For example, the following string represents a URL with two arguments called `arg1` and `arg2` that are token substituted with the runtime value of item type attributes *`itemattr1`* and *`itemattr2`*, respectively:

`http:\\www.oracle.com?arg1=–&itemattr1–&arg2=–&itemattr2–`

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

3. If your URL attribute contains an argument string, you must adhere to the following restrictions:
   - You cannot token substitute that argument string with another item attribute of type Document.
   - You can token substitute that argument string with another Form attribute or URL attribute, however, the argument string for the other attribute is not further token substituted.
4. If you need to pass a date and time as an argument to a URL, you should use TO_CHAR to format the string as YYYY/MM/DD+HH24:MI:SS. Similarly, you need to do the correlating format translation in the function that the URL calls, using TO_DATE. This formatting is required because in multibyte databases, the month portion of the DD–MON–YYYY format could potentially translate to a value that is not acceptable across a URL.
5. Choose OK when you are done.

▶ To Define a Form Attribute

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

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

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

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

See: To Token Substitute an Attribute: page 4 – 36.

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

3. If your form attribute contains an argument string, you must adhere to the following restrictions:
   - You cannot token substitute the value of *argN* with another item attribute of type Document.
   - You can token substitute the value of that argument string with another Form attribute or URL attribute, however, the argument string for the other attribute is not further token substituted.
4. Choose OK when you are done.

▶ To Define a Document Attribute

1. Enter a string that identifies the document in the default value field of the Attribute property page.
2. You can identify one of two types of document for a document attribute: a PL/SQL document or an external document stored in a Document Management repository.
3. A PL/SQL document represents data from the database, generated from a PL/SQL procedure. Specify the default value of a PL/SQL document as

   plsql:*<procedure>/<document_identifier>*.

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

   > **Note:** The PL/SQL procedure must follow a standard API format. See:Standard API for a "PL/SQL" Document: page 7 – 11.

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

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

4. If you wish to generate the document identifier for a PL/SQL document dynamically, you can token substitute the document identifier with other item type attributes. See: To Token Substitute an Attribute: page 4 – 36.

   For example:

   `plsql:po_wf.show_req/&item_attr1:&item_attr2`

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

5. To specify a document managed by an external document management system, enter the following in the default value field:

   `DM:<node_ID>:<document_ID>:[version]`

   Replace *<node_ID>* with the node ID of the document management system, as listed in the Document Management Nodes web page. Replace *<document_ID>* with the ID assigned to the document by the document management system. You may also optionally specify the version number of the document, if multiple versions of the document exist. See: Defining Document Management Repositories: page 2 – 31.

6. Choose OK when you are done.

▶ To Copy an Item Type

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

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

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

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

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

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.

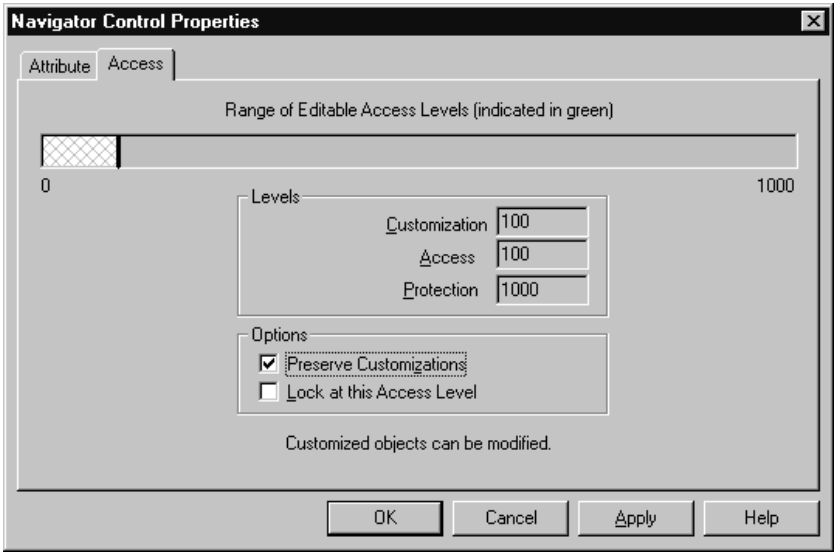
   When you are done, choose OK.

   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 – 8

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 – 70.

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

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

and leaving objects that have been previously customized untouched.

These two modes match the upgrade and upload behavior of the Workflow Definitions Loader program. See: To Set the Access Level for an Object: page 4 – 17 and Using the Workflow Definitions Loader: page 2 – 76.

▶ To Set the Access Level for an Object

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

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

Selected Checkbox	Resulting Levels	Levels Allowed to Modify the Object
NONE—Object can be updated at any time, by anyone.	Customization = 0 Access = 100 Protection = 1000	Object may be updated by any access level (0–1000).
Preserve Customizations—Disallow customized objects from being overwritten during a workflow definition upgrade.	Customization = 100 Access = 100 Protection = 1000	Object may only be updated by access levels from 100–1000. If, the "Allow modifications of customized objects" checkbox is checked, customized objects can also be updated by access levels 0–99, as represented by grey crosshatches in the indicator bar.

Table 4 – 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 grey crosshatches in the indicator bar.

Table 4 – 1 (Page 2 of 2)

3. Choose the Apply button to save your changes.

   Note: An object appears with a small red lock over its icon in the navigator tree to indicate that it is a read–only 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.

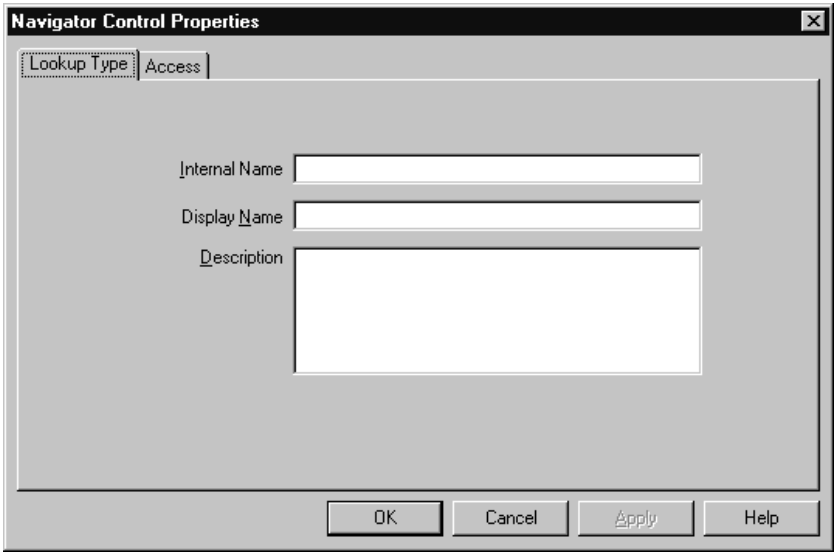
Lookup Types

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 4 – 19.

▶ 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 leading/trailing spaces and a translatable Display Name. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying a lookup type.

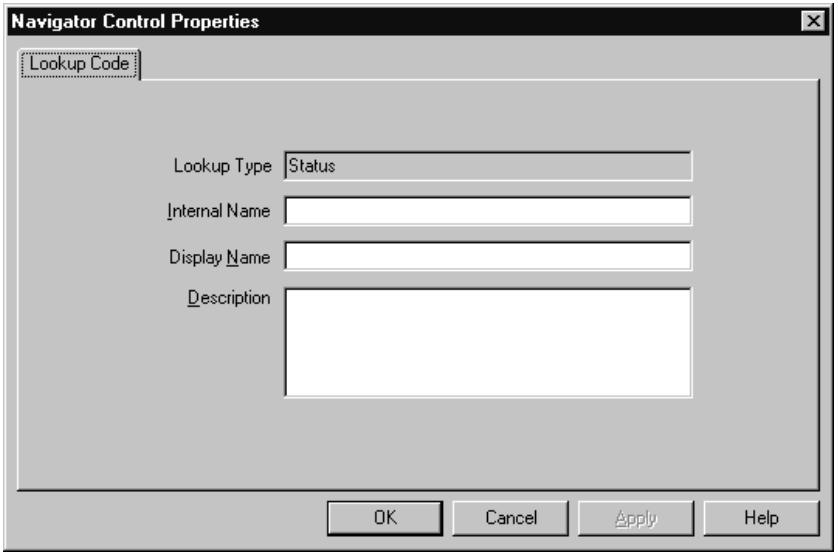
   Attention: You cannot update the internal name for a lookup type once it is defined.

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

   You can supply an optional description for this lookup type.
3. Select the Access tab page to set the access levels allowed to modify this lookup type. See: Allowing Access to an Object: page 4 – 16.
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 4 – 20.

▶ 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 leading/trailing spaces and a Display Name for the lookup code. You can also enter an optional description. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying a lookup code.

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

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

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

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

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

When you define a message, you can specify that the message prompts a recipient for a special response value that the Workflow Engine then uses to determine how to branch to the next eligible activity in the process. You can also create a message with context sensitive content by including message attributes in the message subject and body that reference item type attributes. You can also attach message attributes that represent entire documents or URL content to a notification message. In addition, you can create message attributes that generate a response section that is unique to the message.

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

Message Result

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

To create a message that prompts for this special response, complete the Result tab in the message's property page. The information you enter creates a special 'Respond' message attribute for the message that has an internal name of RESULT. The RESULT message attribute has a data type of Lookup and must be set to the same lookup type as the notification activity's Result Type. This ensures that the performer of the notification can choose from a list of possible response values that matches the list of possible results that the notification activity is expecting. See: Send and Respond Message Attributes: page 4 – 23.

Send and Respond Message Attributes

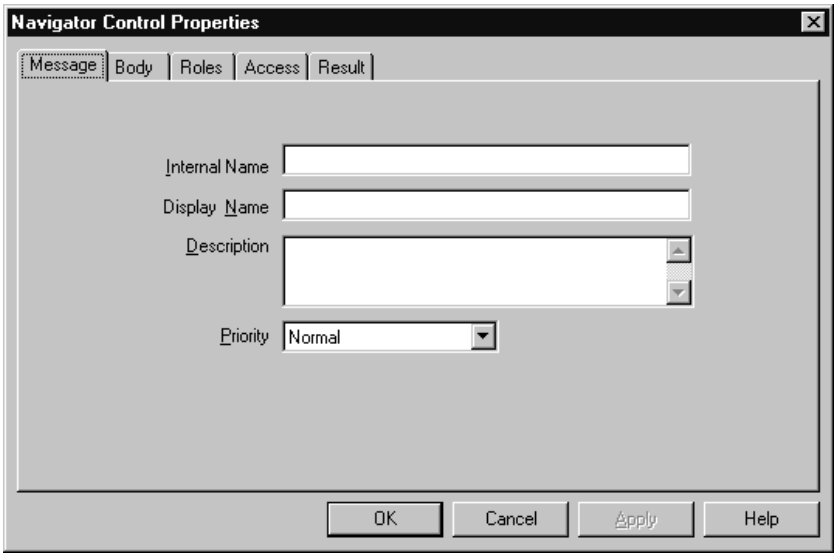
Once you create a message, you can define as many message attributes as necessary for that message. Message attributes are listed beneath a message in the navigator tree.

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

Each message attribute has a specific data type. The value of a 'Send' message attribute can be a constant or can be a value returned by an item type attribute of that same data type. To embed a message attribute in a message's subject or body for token substitution, specify the internal name of the message attribute using the format &MESGATTR within the subject or body text.

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

▶ 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 leading/trailing spaces and provide a display name. You may also enter an optional description. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying a message.

   ☞ **Attention:** To update the internal name for a message once it is defined, you must use a special SQL script. See: Wfchmsg.sql: page 14 – 7.

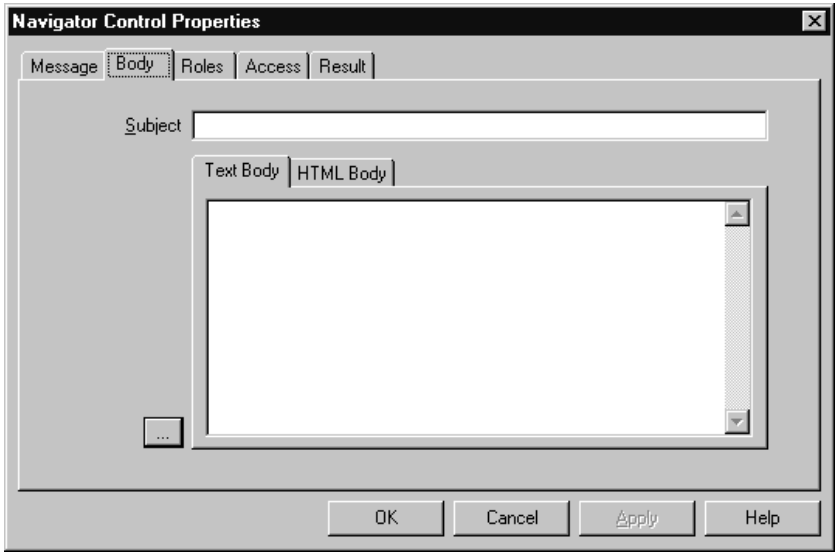
   Caution: Do not include colons ":" or leading/trailing spaces in your internal name.
3. Choose High, Normal, or Low for the default priority of the message. The priority level simply informs the recipient of the urgency of the message. It does not affect the processing or delivery of the message.

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

determined at runtime. See: To Define Nodes in a Process: page 5 – 5.

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

4. Choose Apply to save your changes.



5. Select the Body tab to display the Body property page of the message.

6. The subject gets its default value from the display name that you entered in the Message tab. You can choose to keep the default subject or enter a new subject for the message. The subject can include message attributes that get token replaced with runtime values when the message is delivered. To include a message attribute in the subject, use an "&" followed by the message attribute's internal name. See: Send and Respond Message Attributes: page 4 – 23 and To Define a Message Attribute: page 4 – 29.

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

7. Enter a plain text message body in the Text Body field. You can select the ellipsis button (...) to expand the view of the Subject and Text Body fields in another window.

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

   Attention: Make sure you enter a plain text message body in the Text Body field. If Text Body is null, you get an empty notification when you view your message from a plain text E–mail reader.

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

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

   Attention: Oracle Workflow Builder does not verify the HTML formatting of the message body.

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

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

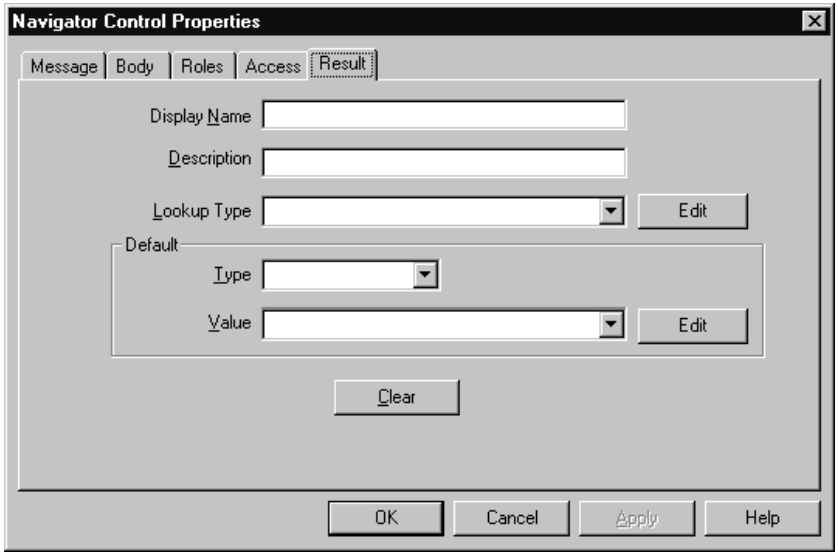
   Attention: Oracle Workflow does not fully support references to icon and image files in the HTML message body. Although your web server may be able to resolve the location of these files for proper display in the Notification Details web page, the Notification Mailer and third party E–mail applications are not able to identify the location of these files when users view the HTML version of their notifications in E–mail.

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

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

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

10. Choose Apply to save your changes.
11. Select the Roles tab page to specify the roles that have access to this message. (This functionality will be supported in a future release.)
12. Select the Access tab page to set the access levels allowed to modify this message. See: Allowing Access to an Object: page 4 – 16.
13. If you want the notification message to prompt the performer for a response value and you want Oracle Workflow to interpret that response value as the result of the notification activity, select the Result tab page and complete the information requested. Oracle Workflow uses the information you specify in the Result tab page to create a special 'Respond' message attribute called RESULT. See: Message Result: page 4 – 22



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

Note: In releases of Oracle Workflow prior to Release 2.5, if you wanted your message to prompt a performer for a response and have that response be interpreted as the result of the notification activity, you had to create a special 'Respond' message attribute whose internal name was RESULT. This special RESULT attribute appeared as a message attribute located beneath its parent message in the navigator tree. In the current release of Oracle Workflow Builder, if you load a process definition created from an earlier release, the special RESULT attribute automatically appears in the Result tab of the parent message's property page.

Note: To create any other type of message attribute, see: To Define a Message Attribute: page 4 – 29.

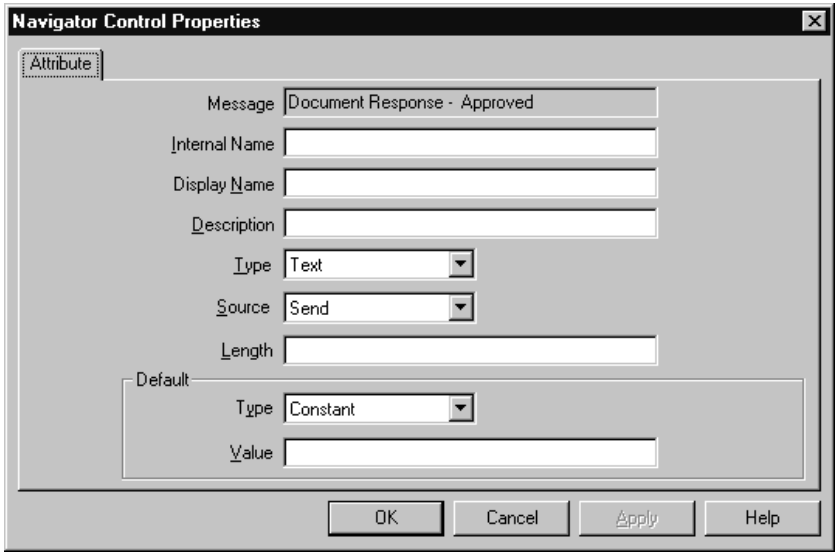
14. Choose Apply to save your changes, OK to save your changes and close the property page or Cancel to cancel your changes and close the property page.
15. The message you just defined now appears beneath the Message branch in the navigator tree. You can review or edit the properties of this message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

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

    Edit the property page that appears, making sure the message attribute has the proper Source. The Default Value region is automatically set to Item Attribute and references the originating item attribute.

18. You can also create message attributes that are not based on existing item type attributes. See: To Define a Message Attribute: page 4 – 29.

▶ To Define a Message Attribute



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

   An Attribute property page appears.

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

   Attention: To update the internal name for a message attribute once it is defined, you must use a special SQL script. See: Wfchmsga.sql: page 14 – 8.

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

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

4. Enter a Display Name. This is the name that appears in the navigator tree. If this is a 'Respond' message attribute, then this display name is also used as the response prompt.

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

5. Enter an optional description. If this is a 'Respond' message attribute, use this field to elaborate on response instructions.
6. Select the data type of the attribute.
7. Depending on the Type of your attribute, provide the following default information:
   - **Text**—Specify the maximum length of the text attribute.
   - **Number**—Optionally provide a format mask for your number and a default number value.
   - **Date**—Optionally supply a format mask for the date and a default date value.
   - **Lookup**—Choose the name of a predefined Lookup Type from which to draw values. Choose a lookup code for the default value.
   - **URL**—Specify a Universal Resource Locator (URL) to a network location in the Default Value field. See: To Define a URL Attribute: page 4 – 11.

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

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

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

Specify a developer form function name and any optional argument string (form function parameters) in the Default Value field. See: Overview of Menus and Function Security, *Oracle Applications Developer's Guide* and To Define a Form Attribute: page 4 – 12.

Attention: 'Send' and 'Respond' message attributes of type Form appear only when your Notifications web pages are launched from Oracle Applications. The attached form icon is enabled if a notification message includes a 'Send' message attribute of type Form. The notification recipient can click on the attached form icon to drill down to the form function defined by the message attribute.

Attention: If a message includes a 'Respond' message attribute of type Form, the attached form icon that appears in the notification's Response section simply drills down directly to the designated form function. This form function must be coded with a call to the Workflow Engine *CompleteActivity( )* API to inform the Workflow Engine that the notification response is complete. See: Workflow Engine APIs: page 8 – 14.

- **Document**—Enter a string that identifies the document in the Default Value field. See: To Define a Document Attribute: page 4 – 13.
- **Role**—Specify a role name. If a message attribute of type role is included in a notification message, the attribute automatically resolves to the role's display name, eliminating the need for you to maintain separate attributes for the role's internal and display names. Also when you view the notification from a web browser, the role display name is a hypertext link to the email address for that role. To set a default value for the attribute, you must initially load roles from the database. See: Roles: page 5 – 19.

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.

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

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

8. If your message attribute type is URL or DM Document, specify a Frame Target. When you reference this message attribute in a

message, the URL or URL for the DM document frame opens according to what you specify as the frame target. The frame target can be:

- New Window—the URL loads in a new, unnamed browser window.
- Same Frame—the URL loads in the same frame as the element that references the URL attribute.
- Parent Frameset—the URL loads into the immediate FRAMESET parent of the current frame. This value is equivalent to Same Frame if the current frame has no parent.
- Full Window—the URL loads into the full, original window, thus cancelling all other frames. This value is equivalent to Same Frame if the current frame has no parent.

9. If your message attribute is a Send attribute and is of type URL or Document, you can check Attach Content to attach the content of the attribute to the notification message. When you view your notification from the Notification web page interface, you see a document icon following the notification message body that displays the contents of the attached message attribute when you click on it. If you view your notification from E–mail, the presentation of the attachment will vary depending on what your E–mail notification preference setting is. See: Reviewing Notifications via Electronic Mail: page 10 – 2.

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

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

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

11. 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.

12. Any message attribute you define appears beneath the respective message you defined it for in the navigator tree. You can review or edit the properties of an attribute at any time by double–clicking on the attribute in the navigator tree or by selecting the attribute and choosing Properties from the Edit menu. Respond message attribute icons in the Navigator tree have a red question mark overlay to help you distinguish them from Send message attribute icons.

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

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.

See Also

Example 'Respond' Message Attributes: page 4 – 33

Using the Edit Button in a Property Page: page 3 – 8

Reviewing Notifications via Electronic Mail: page 10 – 2

Example 'Respond' Message Attributes

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

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 4 – 2 (Page 1 of 1)

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

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

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

Do you approve?

Action: " "

Approve

Reject

Review Comments: " "

If there is no required date, leave this blank.

Required Date: " "

This is the maximum approved amount.

Maximum Amount: " "

Table 4 – 3 (Page 1 of 1)

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

```
Enter the <Display Name> on line <Sequence>. <Description>
<Type_Hint>
```

<Display Name> is replaced with the Display Name of the message attribute. *<Sequence>* is replaced with the relative sequence number of the 'Respond' message attribute as it appears in the Navigator tree among all 'Respond' message attributes (that is, the presence of 'Send' message attributes is ignored when determining the sequence). *<Description>* is replaced with the Description of the message attribute. In addition, *<Type_Hint>* is replaced with one of the following statements, if the message attribute matches one of these data types:

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

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

Enter the Action on line 1. Do you approve? Value must be one of the following: Approve Reject Enter the Review Comments on line 2. Value must be 2000 bytes or less. Enter the Required Date on line 3. If there is no required date, leave this blank. Value must be a date in the form "DD–MON–YYYY". Enter the Maximum Amount on line 4. This is the maximum approved amount. Value must be a number.

Table 4 – 4 (Page 1 of 1)

▶ To Token Substitute an Attribute

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

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

  Attention: If you find that you need to nest message attributes more than two layers deep to display the necessary message body content, you should investigate creating a PL/SQL document–type message attribute. See: External Document Integration: page 4 – 5.

▶ To Copy a Message

1. Select the message to copy in the navigator tree.

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

   Note: You can also use the Copy and Paste options in the Edit menu.
4. Enter a new internal name and display name.
5. Make any additional modifications to the properties of the message.
6. When you are done, choose OK.

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

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 or some external program to perform an automated function. A process activity is a modelled workflow process, which can be included as an activity in another process to represent a sub–process.

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

Oracle Workflow provides an item type called Standard that includes generic activities you can use in any process you define. For example, some of the activities perform standard functions such as comparing two values. See: Standard Activities: page 6 – 2.

Oracle Workflow also provides an item type called System:Error that includes a standard error process and activities you can use to create a custom error process. You can assign an error process to a process activity. If an error occurs, the error process informs Oracle Workflow how to handle the error. See: Default Error Process: page 6 – 20.

Notification Activity

When the workflow engine reaches a notification activity, it issues a Send( ) API call to the Notification System to send the message to an assigned performer. You define the message that the notification sends. The message can be an informative note or it can prompt the performer for a response. When a performer responds to a notification activity, the Notification System processes the response and informs the workflow engine that the notification activity is complete so that it can continue processing the next eligible activity. See: To Create a Notification Activity: page 4 – 41.

You specify the performer of a notification activity when you include the notification activity as a node in the process. You can either designate the performer to be a specific role or an item type attribute that dynamically returns the name of a role. See: To Define Nodes: page 5 – 7 and Roles: page 5 – 19.

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

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

Attention: You should expand roles to send out a broadcast–type message that you want all users of that role to see.

If you do not expand the role for a notification activity, Oracle Workflow sends one copy of the notification message to the assigned performer role and that notification is visible in the notification queue of all the users in that role. If one user in that role responds or closes that notification, the notification is removed from the notification queue of all other users in that role.

- Specify a post–notification function that the Workflow Engine executes in response to an update of the notification's state after the notification is delivered. The Workflow Engine runs the post–notification function in RESPOND, FORWARD, TRANSFER, or TIMEOUT mode depending on whether the notification recipient responds to, forwards, or transfers the notification, or whether the notification times out, respectively.

  For example, if you wish to restrict the roles that a notification can be forwarded to, you can specify a post–notification function that the Workflow Engine executes in FORWARD mode when the notification recipient attempts to forward the notification.

The post–notification function would audit the role and either allow the forward to occur or reject it with an error. See: Post–notification Functions: page 8 – 10 and Notification Model: page 8 – 141.

To create a post–notification function, you should use the same PL/SQL API required for function activities. See: Standard API for PL/SQL Procedures Called by Function Activities: page 7 – 2.

By both checking Expand Roles and specifying a post–notification function, you can create your own custom vote tallying activity. See: Voting Activity: page 4 – 50.

Function Activity

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

If you pass a parameter for the stored procedure, you can expose that parameter as an activity attribute. The activity attribute's value can be set when you define that activity as a node in your process. Note that these activity attributes are available only to the current activity and are not global like item type attributes. See: To Define Activity Attribute Values: page 5 – 11.

As an external program, a function activity is able to enqueue payload information into an Oracle8 outbound queue for some external agent to dequeue and consume. The external agent can similarly enqueue updated attributes and a completion result into an inbound queue that the Workflow Engine consumes and processes. See: To Create a Function Activity: page 4 – 43.

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 4 – 46.

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

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 Workflow Engine to perform the activity, you can enter an estimated cost and update it later as you accumulate more information about its performance. Generally, you should assign complex, long running activities a high cost.

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

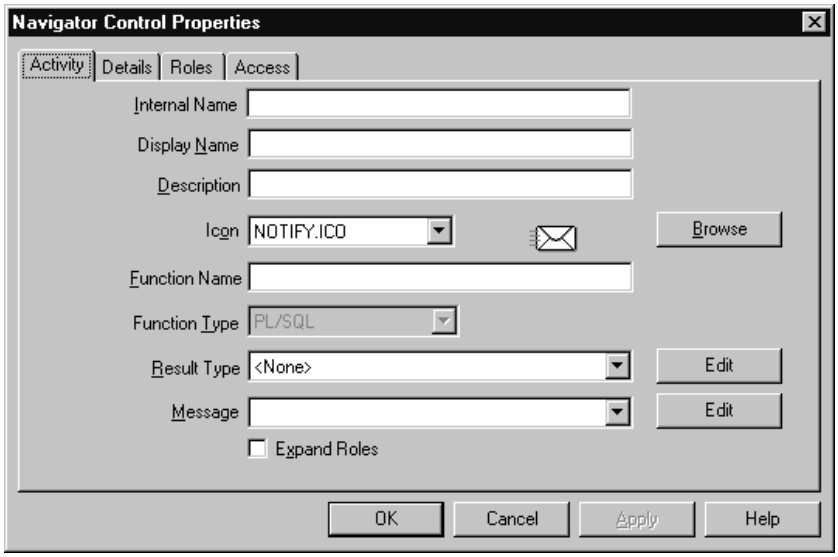
Attention: Although the cost is entered and displayed in seconds in Oracle Workflow Builder, it is actually converted and stored in the database as hundredths of a second.

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 a cost higher than a designated threshold to a background process. The engine then continues processing the next pending eligible activity that may occur in another parallel branch of the process.

The default threshold for the Workflow Engine is 50 hundredths of a second. Activities with a cost higher than this are deferred to background engines. A background engine can be customized to execute only certain types of activities. You can set the workflow engine threshold through SQL*Plus. See: Setting Up Background Workflow Engines: page 2 – 34 and To Set Engine Thresholds: page 2 – 36.

▶ To Create a Notification Activity



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

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

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

   Attention: To update the internal name for an activity once it is defined, you must use a special SQL script. See: Wfchact.sql: page 14 – 5.

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

3. Indicate the result type (a predefined Lookup Type) for this activity. Result types list the possible results returned by this activity. Your workflow diagram may branch depending on the value returned by your completed activity. See: To Create Lookup Types: page 4 – 19.

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

4. Select the name of the message you want this notification to send. See: To Create a Message: page 4 – 24.

5. If you plan to assign this notification to a role consisting of multiple users and you want to send an individual copy of this notification to each user in the role, then check Expand Roles. If you uncheck Expand Roles, then only one copy of the notification is delivered to the role as a whole. See: Notification Activity: page 4 – 38.

6. You can optionally specify a PL/SQL stored procedure in the Function field. The procedure is known as a post–notification function and the Workflow Engine executes this post–notification function in RESPOND, FORWARD, TRANSFER or TIMEOUT mode depending on whether the recipient responds to, forwards, or transfers the notification or whether the notification times out. You can employ a post–notification function to couple processing logic to the notification activity. See: Standard API for PL/SQL Procedures Called by Function Activities: page 7 – 2 and Post–Notification Functions: page 8 – 10.

   If you check Expand Roles and you assign a message that has a special Result, to this notification activity, then use the Function field to specify the name of a custom PL/SQL stored procedure that tallies the responses you get back from each of the recipients of this notification. Specify the procedure using the format: <*package_name*>.<*procedure_name*>. See: Voting Activity: page 4 – 50.

7. Choose an icon that identifies your activity. You can use any icon, as long as the icon is stored in a .ico file, to symbolize the action of an activity. See: Adding Custom Icons to Oracle Workflow: page 2 – 69.

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

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

8. Choose Apply to save your changes.

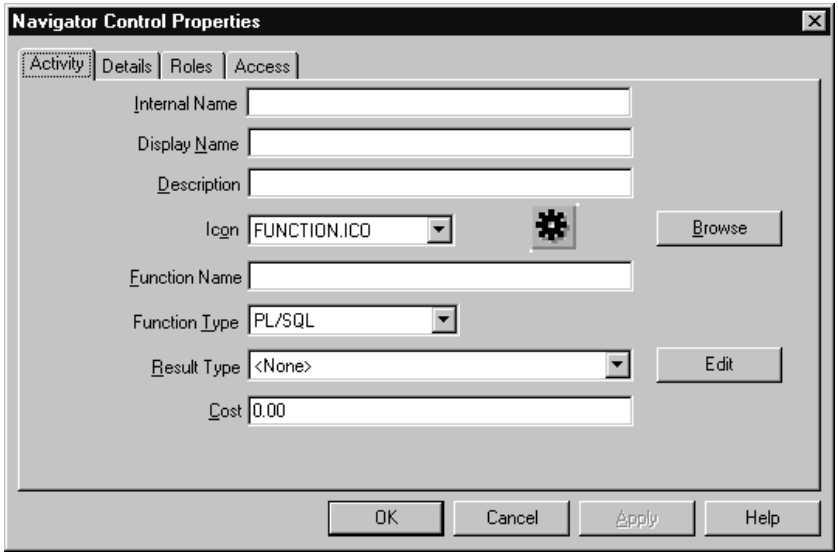
9. Select the Details tab to display and modify optional Details of the activity. See: To Define Optional Activity Details: page 4 – 48.

10. Select the Roles tab page to specify the roles that have access to this notification activity. (This functionality will be supported in a future release.)
11. Select the Access tab page to set the access levels allowed to modify this notification. See: Allowing Access to an Object: page 4 – 16.
12. Choose OK to save your changes and close the property pages.
13. The notification activity now appears beneath Notifications in the navigator tree. You can review or edit the properties of this activity at any time by double–clicking on the activity in the navigator tree or by selecting the activity and choosing Properties from the Edit menu or by pressing Enter on your keyboard.

See Also

Using the Edit Button in a Property Page: page 3 – 8

▶ To Create a Function Activity



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

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

   Attention: To update the internal name for an activity once it is defined, you must use a special SQL script. See: Wfchact.sql: page 14 – 5.

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

3. Enter the name of the function you want this activity to execute. In the Type field, specify whether the function is a PL/SQL function or an External function. If it is PL/SQL, specify the function as *<package_name>.<procedure_name>*. See: Standard API for PL/SQL Procedures Called by Function Activities: page 7 – 2.

   If you wish to define an external function activity, set the function Type to External. The Workflow Engine enqueues an entry in the "Outbound" queue and sets the correlation value of that entry to the value you specify in the Function field. See: Workflow Queue APIs: page 8 – 113.

   You must create your own queue handler to search for this type of record on the "Outbound" queue. The queue handler must execute the action associated with the record and seed the result of the action onto the "Inbound" queue. The background engine then takes care of messages on the inbound queue and restarts your original workflow process. See: Deferred Processing: page 8 – 6.

4. Indicate the result type (a predefined Lookup Type) for this activity. Result types list the possible results returned by this activity. Your workflow diagram may branch depending on the value returned by your completed activity. See: To Create Lookup Types: page 4 – 19.

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

5. Specify the cost of this function activity. See: Activity Cost: page 4 – 40.

6. Choose an icon that identifies your activity. You can use any icon, as long as the icon is stored in a .ico file, to symbolize the action of an activity. See: Adding Custom Icons to Oracle Workflow: page 2 – 69.

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

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

7. Choose Apply to save your changes.
8. Select the Details tab to display and modify the optional details of the activity. See: To Define Optional Activity Details: page 4 – 48.
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 4 – 16.
11. The function activity now appears beneath Functions in the navigator tree. You can review or edit the properties of this activity at any time by double–clicking on the activity in the navigator tree or by selecting the activity and choosing Properties from the Edit menu or by pressing Enter on your keyboard.
12. If your function requires input arguments, you can expose those arguments in Oracle Workflow Builder as attributes of the function activity. Function activity attributes behave as parameters whose values you can modify for each usage of the activity in a process. Function activity attributes are specific to a function activity and are not global to a process. See: To Define an Item Type or Activity Attribute: page 4 – 8.

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

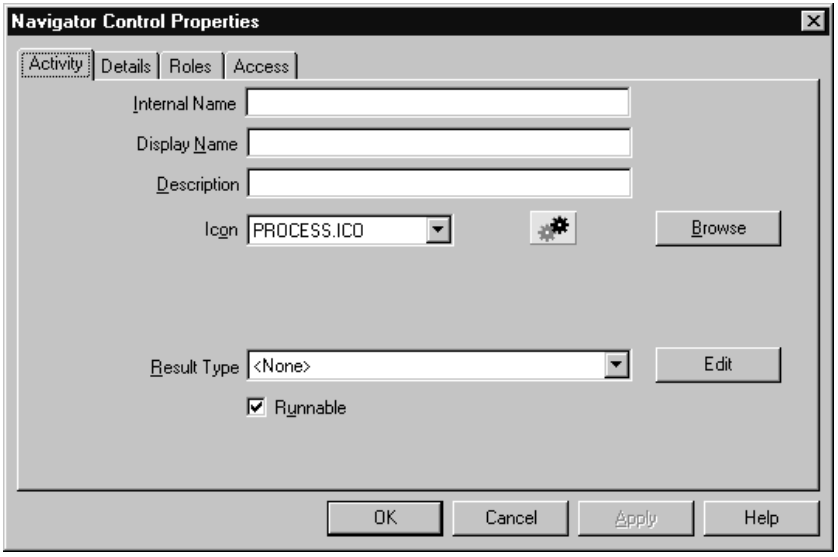
    When you include a function activity as a node in a process, you can assign a value to the function activity attribute that is specific to that node. See: To Define Activity Attribute Values: page 5 – 11.

See Also

Using the Edit Button in a Property Page: page 3 – 8

▶ To Create a Process Activity

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



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

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

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

Attention: To update the internal name of an activity once it is defined, you must use a special SQL script. See: Wfchact.sql: page 14 – 5.

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

3. Indicate the result type (a predefined Lookup Type) for this activity. Result types list the possible results returned by this process. See: To Create Lookup Types: page 4 – 19.

   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 – 69.

   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 8 – 16.

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

6. Choose Apply to save your changes.

7. Select the Details tab to display and modify the optional details of the activity. See: To Define Optional Activity Details: page 4 – 48.

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 4 – 16.

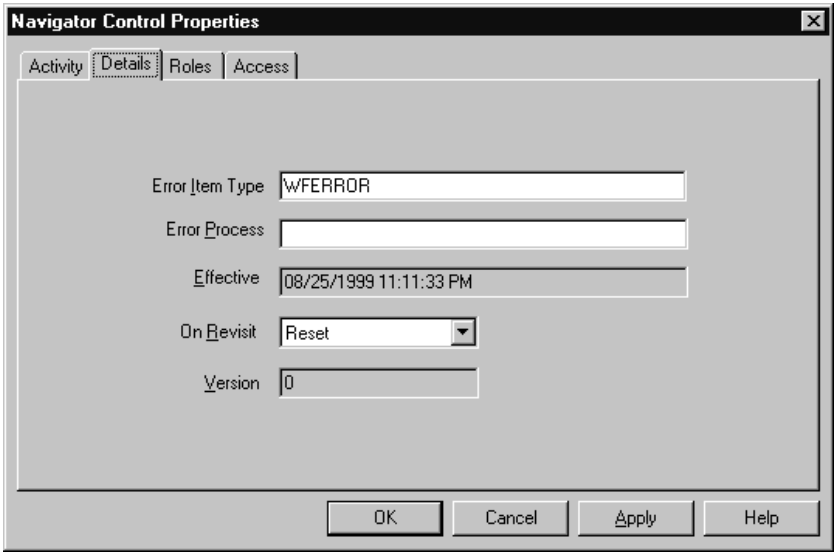
9. Choose Apply to save your changes, OK to save your changes and close the property page or Cancel to cancel your changes and close the property page.

10. The process activity now appears beneath Processes in the navigator tree. You can review or edit the properties of this activity at any time by selecting the activity and choosing Properties from the Edit menu or by pressing Enter on your keyboard.

See Also

Using the Edit Button in a Property Page: page 3 – 8

▶ To Define Optional Activity Details



1. Select the Details tab of the activity's property page.
2. If you are creating a process activity, you can specify an error process to execute in the event that an error occurs in the current process. Enter the internal name of the item type that owns the error process and then specify the internal name of the error process activity to execute. Note that the error process item type does not need to be open in your current Oracle Workflow Builder session for you to define it here. See: Default Error Process: page 6 – 20.
3. The effective date tells you when this version of the activity is available for the Workflow Engine to execute. If the Effective Date field is blank, the activity is effective immediately.

   You set the effective date when you save your changes using the Save As option in the File menu. All your activity modifications share the same effective date when you save.
4. Select a value for On Revisit to determine how the Workflow Engine handles this activity when it is transitioned to more than once. If this activity is the first activity that is revisited, as in a

loop, you should set On Revisit to specify how you want the Workflow Engine to process the loop. The first activity in a loop is also called the pivot activity. For all other activities in a loop, the value of On Revisit is irrelevant.

If On Revisit is set to Ignore, the Workflow Engine executes the activity only once, and ignores the activity for all subsequent revisits.

If On Revisit is set to Reset, the Workflow Engine resets the completed activities in the loop by traversing through the loop in reverse order from the pivot activity, executing each activity in CANCEL mode. You can include special logic in each function's CANCEL mode to undo prior operations. The Workflow Engine then traverses through the loop in forward order, reexecuting each activity, starting with the pivot activity, in RUN mode.

If On Revisit is set to Loop, the Workflow Engine simply reexecutes the pivot activity and all activities that follow in the loop, without resetting, as if they have never been executed before. See: Looping: page 8 – 7.

5. The version number identifies which revision of the activity you are examining. The engine ensures that it uses the most recent updates to an activity by using the latest effective version number of that activity.
6. Choose Apply to save your changes.

▶ **To Copy an Activity**

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

   Note: You can also use the Copy and Paste options in the Edit menu.
4. When you are done, choose OK.

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

Voting Activity

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

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

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

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

▶ Creating a Voting Activity

1. Create a voting lookup type that contains the responses you want to tally in your voting activity. See: To Create Lookup Types: page 4 – 19.
2. Create a voting message that prompts a recipient to respond with one of the values in the voting lookup type. Complete the Result tab for the message. Set the lookup type in the Result tab to the voting lookup type defined in Step 1. See: To Create a Message: page 4 – 24
3. Select the item type that you want to create a voting activity for in the navigator tree, then choose New Notification from the Edit menu.
4. Specify an Internal Name (all uppercase and no leading/trailing spaces) and a Display Name. Use the description to provide an explanation about this voting activity.

Attention: To update the internal name for an activity once it is defined, you must use a special SQL script. See: Wfchact.sql: page 14 – 5.

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

5. The Result Type field must contain the lookup type that lists the responses that you want the voting activity to tally. This is the voting lookup type defined in Step 1.
6. Choose an icon that identifies your voting activity.
7. In the Message field, select the name of the voting message you created in Step 2. The voting message prompts the recipient for a response. The response choices are one of the predefined values specified in your voting lookup type.
8. Check Expand Roles so that the Workflow Engine polls for responses from the multiple users in the role rather than just from the first user in the role that replies. See: Notification Activity: page 4 – 38.
9. In the Function field, specify a function that tallies the responses from users. You can use the PL/SQL procedure *WF_STANDARD.VOTEFORRESULTTYPE* that the Standard Vote Yes/No activity calls. *WF_STANDARD.VOTEFORRESULTTYPE* is a generic tallying function. The Result Type that you specify for the voting activity defines the possible responses for the function to tally. The activity attributes that you define for the voting activity determine how the function tallies the responses. See: Vote Yes/No Activity: page 6 – 11.

   Alternatively, you can specify your own custom tallying function, but you should make sure it conforms to the standard API for function activities. Specify the procedure using the format: <*package_name*>.<*procedure_name*>. See: Standard API for PL/SQL Procedures Called by Function Activities: page 7 – 2.
10. Choose Apply to save your changes.
11. Select the Details tab to display and modify the Details property page of the activity. See: To Define Optional Activity Details: page 4 – 48.
12. Select the Roles tab page to specify the roles that have access to this notification activity. (This functionality will be supported in a future release.)
13. Select the Access tab page to set the access levels allowed to modify this notification. See: Allowing Access to an Object: page 4 – 16.
14. If you use the *WF_STANDARD.VOTEFORRESULTTYPE* tallying function, create a custom activity attribute of type Number for each possible voting response. Remember that each possible voting response is a lookup code associated with the voting activity's result type. Hence, when you define your custom activity

attribute, the internal name of the activity attribute must match the internal name of the lookup code, that is, the response value.

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

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

15. If you use the *WF_STANDARD.VOTEFORRESULTTYPE* tallying function, then in addition to defining your set of custom activity attributes, you must also define an activity attribute called Voting Option, whose internal name must be VOTING_OPTION. You can also copy the Voting Option activity attribute from the Vote Yes/No standard activity.

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

    - "Wait for All Votes"—the Workflow Engine waits until all votes are cast before tallying the results as a percentage of all the users notified. If a timeout condition occurs, the Workflow Engine calculates the resulting votes as a percentage of the total votes cast before the timeout occurred.

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

Example Voting Methods

1. **Simple Majority**

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

Table 4 – 5 (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 4 – 6 (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 4 – 7 (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 4 – 8 (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	0

Table 4 – 9 (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 4 – 10 (Page 1 of 1)

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

Deleting Objects in Oracle Workflow Builder

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

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

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

CHAPTER

5

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 or create activities directly in the Process window. The properties for an activity node may be viewed or edited by double clicking on the node in the Process window with the select mouse button. You define transitions between activities by drawing arrows from one node to the next using the secondary mouse button.

Notification, function, and process activities make up the nodes of a process. If a process contains a process activity in its diagram, that process activity is known as a subprocess. There is no restriction on the depth of this hierarchy. To display the subprocess diagram in a Process window, double–click on the subprocess activity node in the parent Process window.

Transitions

Transitions appear as arrows in your diagram and represent the completion of one activity and the activation of another. For an activity that completes with a result type of <None>, any transition that you draw from it simply appears as an arrow to the next activity, indicating that as long as the originating activity completes, the process transitions to the next activity.

For an activity that has a defined result type, you must associate the transition arrow that you create with one of the activity's possible results. The result that the activity returns when it completes then determines what the next eligible activity is, as defined by the results–based transitions that originate from the completed activity. For example, "Notify Approver" with a result of 'REJECTED' transitions to "Reject Requisition." See: Requisition Process Activities: page 13 – 14.

You can also create a <Default>, <Any>, or <Timeout> transition for an activity that has a defined result type. The Workflow Engine follows a <Default> transition if no other transition matching the completion result exists. The Workflow Engine follows an <Any> transition regardless of what completion result the activity returns. This allows you to include a generic activity in the process that the Workflow Engine executes in parallel with the result–specific activity. The Workflow Engine follows a <Timeout> transition if the notification

activity times out before completion. See: Setting Up Background Workflow Engines: page 2 – 34.

Activities can have multiple transitions for a single result to create parallel branches.

Timeout Transitions

Draw a <Timeout> transition from a notification activity to some other activity to force the process to perform the other activity if the notification activity does not complete by a specified period of time. See: To Define Nodes in a Process: page 5 – 7.

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's notification preference is to receive 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 the error process associated with the timed out activity or its parent process(es). See: To Define Optional Activity Details: page 4 – 48.

> **Note:** You must have a background engine set up to process timed out activities. See: Setting Up Background Workflow Engines: page 2 – 34.

Creating Multiple Transitions to a Single Activity

You can create multiple transitions to a single activity in a process diagram. Sometimes these multiple transitions indicate that there are multiple ways that the process can transition to this one node and you may want the node to execute just once.

In other cases, the multiple transitions may indicate that the activity may be transitioned to multiple times because it is the starting point of a loop. In these cases, you want the activity to be reexecuted each time it is revisited.

The On Revisit flag for an activity determines whether the activity reexecutes when it is revisited more than once. It is an important flag to set for the pivot activity of a loop. On Revisit is set initially in an activity's Details property page. However, for each usage of an activity in a process, you may change On Revisit for that node in the activity's Node property page. You can also use the standard Loop Counter

activity as the initial activity in a loop to control how many times a process can transition through a loop. See: Looping: page 8 – 7 and Loop Counter Activity: page 6 – 7.

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 6 – 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 corner of the activity node's icon in the Process window.

Initiating a Process

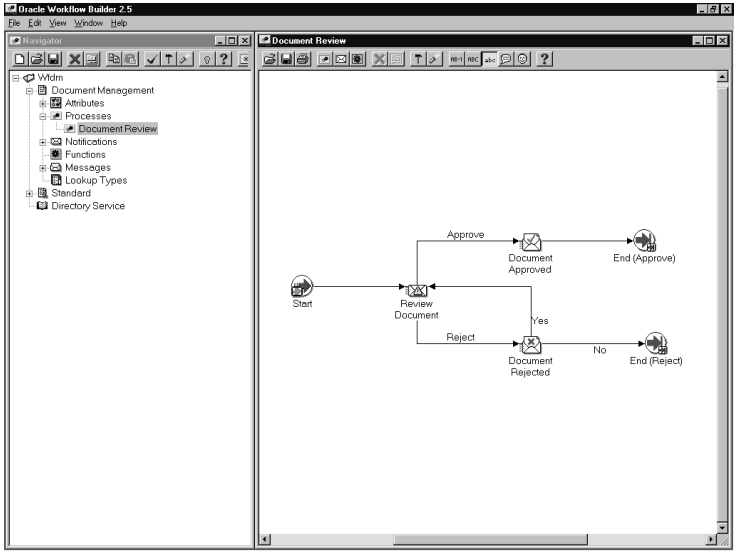
A workflow process begins when an application calls the Workflow Engine *CreateProcess( )* and *StartProcess( )* APIs. A subprocess is started when the Workflow Engine transitions to a process activity that represents the subprocess. See: Workflow Engine APIs: page 8 – 14.

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 5 – 5
- To Define Nodes: page 5 – 7
- To Define Activity Attribute Values: page 5 – 11
- To Create and Edit a Transition: page 5 – 12
- To Display a Process Overview: page 5 – 14
- To Print a Process: page 5 – 15
- To Copy a Process Diagram to the Clipboard: page 5 – 15
- To Validate a Process Definition: page 5 – 15

▶ To Add Nodes to a Workflow Process



1. To begin drawing a process diagram, you must first display the Process window for your process activity. To display a process window, you can do one of several things:
   - Double–click on a predefined process activity on the navigator tree.
   - Select a predefined process activity and press Ctrl + E.
   - Select a predefined process activity and choose Process Details from the Edit menu.
   - Use the Quick Start Wizard to create a new process activity. See: To Use the Quick Start Wizard: page 3 – 18.

A Process window opens with the name of your process in the window title.

2. Create a new node in a process by using one of the following methods:

   - Drag and drop 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.

     Note: If you want to drag an activity into a process, where the activity is in a different data store than the process you are dragging it to, then you must first copy the item type that the activity belongs to into the same data store as the process.

   - Choose the New Function, New Process, or New Notification toolbar button to create a new activity.

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

3. You can also create a new node using the right mouse button menu. You can create a new function, notification or process. An Activities property page appears for you to select the activity for this node. See: To Define Nodes in a Process: page 5 – 7.

4. In the Process window, you can display information about an activity by moving your mouse over the activity. The Label Name, Internal Name, Display Name, Comment and Performer, appears in a "tool–tip"–style display.

5. If you single click on an activity node in the Process window, Oracle Workflow Builder expands the navigator tree and highlights the master activity of the node you select.

6. 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.

7. 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 5 – 12.

   If the source activity has a result code associated with it, then a list of lookup values appears when you attempt to create a transition to the destination activity. Select a value to assign to the transition. You can also select the values <Default>, <Any>, or <Timeout> to define a transition to take if the activity returns a result that does

not match the result of any other transition, if the activity returns any result, or if the activity times out, respectively.

You can also drag and drop a lookup code from the navigator tree onto an existing transition in the Process window to change the result of that transition. The lookup code you drag and drop must belong to the same data store and same lookup type as the lookup code you replace.

8. 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.

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

9. 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

Process Window Toolbar: page A – 8

▶ To Define Nodes in a Process

1. Open the Process window for your process activity.
2. To create a new function, notification, or process node, first select the New Process, New Function or New Notification icon from the Process window toolbar. Next, click on the position within the Process window where you want to place this new node. The property page for the new node appears.

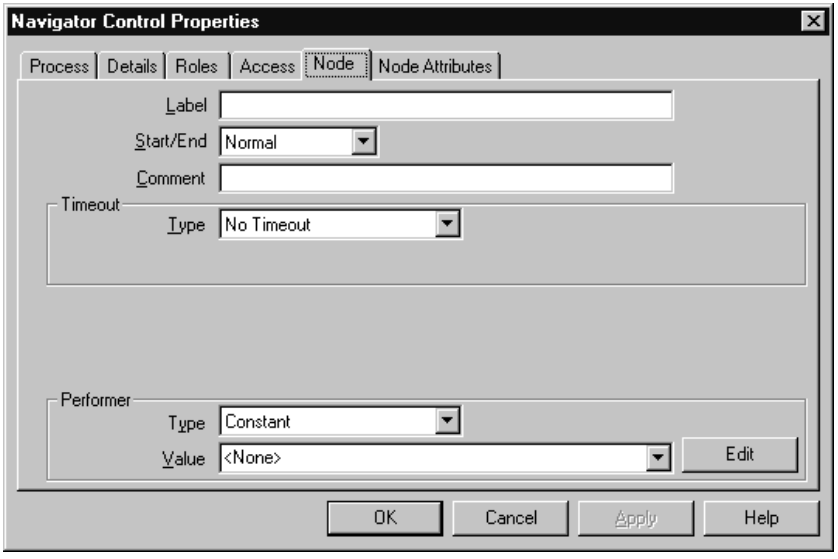
   Note: You can also create a new node by dragging and dropping a predefined activity from the navigator tree into the process window. This automatically populates the node's property page with predefined information. Double–click on the node and skip to Step 5 to further edit its property page.

3. In the Item Type field, select the item type that you want this activity node to be associated with.

4. Choose one of the following methods to define the remaining information for the node.

   - Select either the internal name or display name of a predefined activity. Oracle Workflow Builder then populates all the fields with predefined information from the master activity as shown in the Navigator window.
   - Alternatively, choose the New button to define a new activity. To complete the following tabs of the property page, refer to the sections listed:
     - Process—To Create a Process Activity: page 4 – 46
     - Function—To Create a Function Activity: page 4 – 43
     - Notification—To Create a Notification Activity: page 4 – 41
     - Details—To Define Optional Activity Details: page 4 – 48
     - Roles—The information in this tab is currently not supported.
     - Access—To Set the Access Level for an Object: page 2 – 70

     Caution: Any changes that you make to the any of the above tabs automatically propagate to the master activity and affect all other instances of that activity. Only changes that you make to the Node and Node Attributes tabs are local and specific to the current node activity.



5. Select the Node tab to specify information that is specific to this node. Specify a Label for the node. Since an activity can be used more than once in any given process, the Label field lets you give a unique name to the instance of this particular activity in the process. By default, the label name is the activity name, but if the activity is used more than once in the process, *–N* is appended to the activity name, where *N* represents the '*N*th' instance of the activity in use.

   ☞ **Attention:** When you call most Oracle Workflow APIs, you must pass the activity's label name and not its activity name. See: Workflow Engine APIs: page 8 – 14.

6. Indicate if the current node is a start or end activity in your process, by choosing 'START' or 'End', respectively. 'NORMAL' is the default if it is neither. You may have multiple START and END nodes in your process.

   A Start activity is marked (Start) and has a small green arrow in its activity icon, and an End activity is marked (End) and has a red arrow in its activity icon.

   ☞ **Attention:** The Start/End field is always set to Normal by default for all activity nodes. Even if you use the Standard Start or Standard End activity, you must manually edit the Start/End field to be either Start or End, respectively.

7. For an END node, you must also select a value for the final process result if the overall process activity has a result type associated with it. The list of values for the final process result is derived from the lookup type specified as the process activity's result type.

8. You can provide a comment to yourself about this node.

9. For a notification or (sub)process activity, specify whether the activity must be completed by some specified time. If the activity is not completed by a given time, you can redirect the parent process to transition to a different activity. See: Timeout Transitions: page 5 – 3.

   Choose 'No Timeout' if the activity does not have to be completed by a given time.

   Choose 'Relative Time' if you want the activity to be completed by some constant relative time. You can enter any combination of days, hours and minutes to specify when the activity times out. The value you enter is interpreted as a relative offset from the begin date of the activity, in the unit of MINUTES. A relative timeout value of zero means no timeout.

Choose 'Item Attribute' if you want the activity to be completed by some relative time that is computed dynamically at runtime. Note that you must first create an item attribute of type number to store the computed timeout value and reference that predefined item attribute here. See: Item Type Attributes: page 4 – 2 and To Define an Item Type or Activity Attribute: page 4 – 8.

Attention: The dynamic timeout value stored in this attribute is interpreted as a relative offset from the begin date of the activity, in the unit of MINUTES. A null timeout value or a value of zero means no timeout.

10. For a notification activity node, you can override the priority assigned to the notification's message. Choose 'Default' to keep the default priority of the message.

    Choose 'Constant' to override the default priority with the new specified priority level.

    Choose 'Item Attribute' to override the default priority with a new priority level that is dynamically determined at runtime. Note that you must first create an item attribute of type number to store the computed priority value and reference that predefined item attribute here. See: Item Type Attributes: page 4 – 2 and To Define an Item Type or Activity Attribute: page 4 – 8.

    Note: The computed priority value can be any number between 1–99. Oracle Workflow automatically converts the number to a priority level as follows: 1–33 = High, 34–66=Normal, and 67–99=Low.

11. For a notification activity node, specify the performer of the activity. The performer is the role to whom the notification is sent. You may either select a constant role name or an item type attribute that dynamically determines the role at runtime. Note that you must first create an item attribute of type role to store the computed role name and reference that predefined item attribute here. See: Item Type Attributes: page 4 – 2, To Define an Item Type or Activity Attribute: page 4 – 8, and Roles: page 5 – 19.

    Note: If you set the Performer Type to Constant and you are connected to the database and have loaded roles from the database, you can select a constant role name from the Performer poplist. If you are working in a .wft file data store without any open connection to the database, you can directly type in a valid role display name in the Performer field. When you upload the file to a database, the role will be resolved to the appropriate role data stored in the database based on the role display name you entered.

Note: When you assign a notification to a multi–user role, the Workflow Engine keeps track of the individual from that role that actually responds to the notification. See: Respond API: page 8 – 158.

Note: Although Oracle Workflow Builder allows you to specify a performer for any type of node activity, Oracle Workflow only considers the value of Performer for notification activity nodes.

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

    When you save and close your property page, the activity node appears in the position you specified in the Process window. If this is a new activity you created, a corresponding master activity is also created under the appropriate branch in the navigator tree.

13. If the node is a function or notification activity and the activity has activity attributes, you can assign values to those activity attributes by choosing the Node Attributes tab. See: To Define Activity Attribute Values: page 5 – 11.

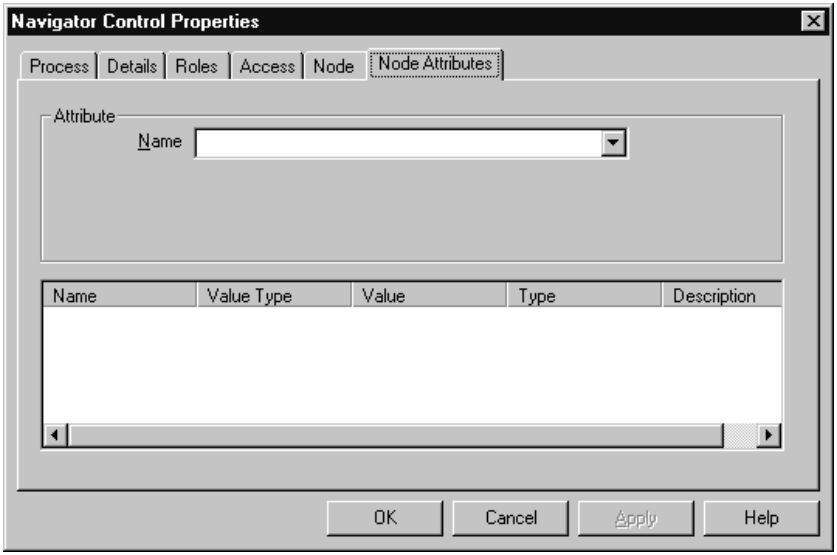
14. If the node is a process activity, then a small subprocess overlay icon appears over the upper right corner of process activity icon. The subprocess overlay icon identifies the node as a subprocess within the process diagram.

See Also

To Find an Object in the Navigator Tree: page 3 – 6

▶ To Define Activity Attribute Values

Activity attribute values are used by the PL/SQL stored procedure that the function or notification activity calls. See: To Define an Item Type or Activity Attribute: page 4 – 8.



1. Display the property pages of an activity node. Select the Node Attributes tab.
2. Select an attribute.
3. In the Value region, enter the value for this attribute. The value can be a constant or a value stored in an item type attribute.

   The value you enter must match the data type of the activity attribute, and 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.

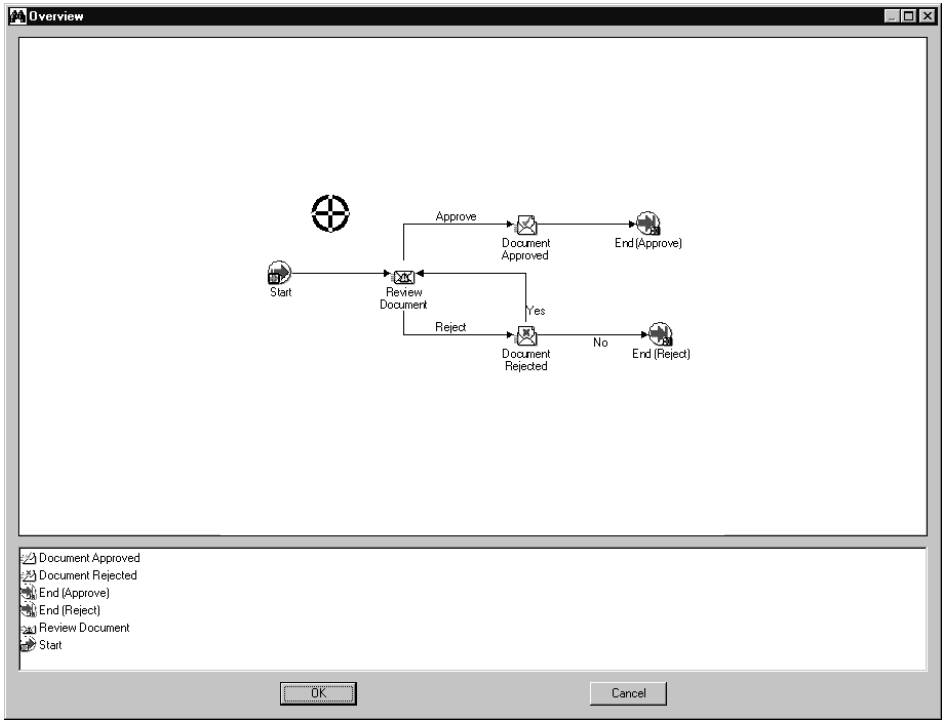
▶ To Create and Edit a Transition

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.

   Note: Overlapping transitions appear in a different color than single, non–overlapping transitions.
2. To edit a transition, select the transition.

3. To reposition a transition label, simply select the label with your mouse and drag it to its new position. The label snaps onto the transition.
4. You can bring up the following menu of editing options at any time by selecting a transition with your mouse and clicking on the right mouse button:
   - Delete Transition—deletes the selected transition.
   - Locked—toggles between locking and unlocking the transition from further edits. If a transition is locked, you cannot add or delete vertex points along the transition, but you can delete the transition.
   - Hidden Label—toggles between displaying and hiding the transition label.
   - Straighten—straightens the transition by removing the extra vertex points that can cause bends in the transition.
   - Results...—if the transition has a result assigned to it, use this option to change the result label on the transition. An additional menu appears that lists the possible result labels you can choose.
5. To bend a transition, create a vertex point by selecting the transition and dragging the transition as you hold down your left mouse button. You can reposition any vertex point to create a bend in the transition.
6. You can create a transition that loops back to its source activity node in one of two ways:
   - Hold down your right mouse button and drag your mouse from a source activity back to itself to create a self loop.
   - From a source activity node, create a transition to another arbitrary activity node. Add a vertex point to create a bend in the transition. Then select and drag the arrowhead of the transition back to the source activity node. Create additional vertex points as necessary to improve the visual display of the looping transition.
7. To remove a single vertex point from a transition, select the vertex and drag it over another vertex to combine the two points.

▶ To Display a Process Overview



1. Place your cursor in the Process window and choose Overview from the right mouse button menu.
2. An Overview dialog window of your process appears.

   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 that one page when printed. However, depending on the printer driver you use, you may get a Print dialog box that lets you scale your image down to one page for printing.

   > **Note:** If your process diagram uses a font that the printer cannot find, your printer driver may either substitute a similar font or not print any text.

▶ To Copy a Process Diagram to the Clipboard

1. Display and make the Process window containing the process you wish to copy active.
2. Choose Copy Design from the Edit menu or from the right mouse button menu.

   This copies the process to the clipboard in the form of a metafile and a bitmap diagram.
3. To paste the metafile–version or bitmap–version of the process diagram into another application window, you should consult the other application's documentation on how to paste metafiles or bitmaps.

   To edit a bitmap image, you must paste the image into an application that can edit bitmaps.

▶ To Validate a Process Definition

1. Choose Verify from the File menu to validate all process definitions for the currently selected data store.
2. The following list is an example of some of the validation that the Verify command performs:

- Checks that a process has at least one Start and one End activity.
- Verifies that a process does not contain itself as a process activity.
- Restricts the same subprocess from being used twice in a process.
- Validates that all possible activity results are modelled as outgoing transitions. If an activity completes with a result that is not associated with an outgoing transition, and a <Default> transition doesn't exist for that activity, the activity enters an 'ERROR' state.
- Validates that activity nodes marked as END nodes do not have any outgoing transitions.
- Validates that a notification activity's result type matches the lookup type defined for the message's 'RESULT' message attribute.
- Verifies that message attributes referenced in a message body for token substitution exist in the message definition.
- For processes that reference objects from another item type, verifies that the requisite item attributes associated with the referenced item type exists.

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 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 display and reference the roles stored in a database.

Referencing Roles in a Workflow Process

One example of how roles are referenced in a workflow process is when you include a notification activity in a process as a node. You must assign that node to a performer. The performer can be a designated role or an item type attribute that dynamically returns a role. To assign a performer to a role, you must initially load the roles from your Oracle Workflow database into your Oracle Workflow Builder session. See: Setting Up an Oracle Workflow Directory Service: page 2 – 17 and To Define Nodes in a Process: page 5 – 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.

Ad Hoc Users and Roles

Oracle Workflow allows you to create new ad hoc users and roles within a workflow process, to add to your directory service. To do so, you define a function activity that makes a server–side call to the appropriate WF_DIRECTORY API and include that function activity in your process diagram. See: Standard API for PL/SQL Procedures Called by Function Activities: page 7 – 2 and Workflow Directory Service APIs: page 8 – 77.

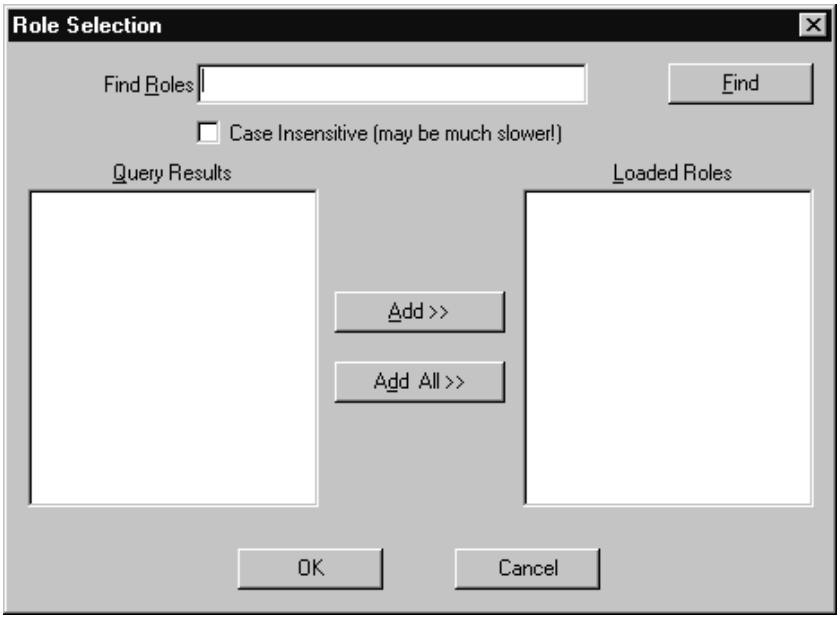
See Also

To Load Roles: page 5 – 20

To Display the Directory Service in Oracle Workflow Builder: page 5 – 21

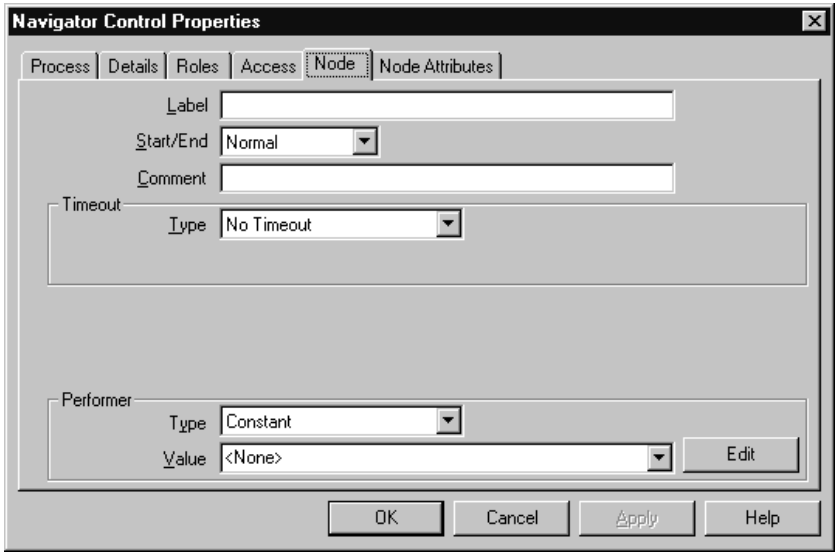
▶ To Load Roles

1. If you are not connected to an Oracle Workflow database, choose Open from the File menu to connect to the database and open your item type.
2. Choose from the File menu, Load Roles from Database. A Role Selection window appears. You can enter search criteria using SQL query syntax in the Find Roles field to find a subset of roles, or just choose Find without specifying any search criteria to identify all roles. The Role Selection window finds the roles you specify and displays them in the Query Results list box.



3. Select the roles you want to load from the Query Results list and choose Add to add them to the Loaded Roles list. Alternatively, just choose Add All to add all the roles in the Query Results list to the Loaded Roles list. Choose OK to load the selected roles into Oracle Workflow Builder and make them available to the workflow objects in your open item type.

   The workflow objects that need to reference role information contain specific fields in their property pages. These fields are poplist fields that display the list of roles you loaded from the database, as shown in the following Node property page example.

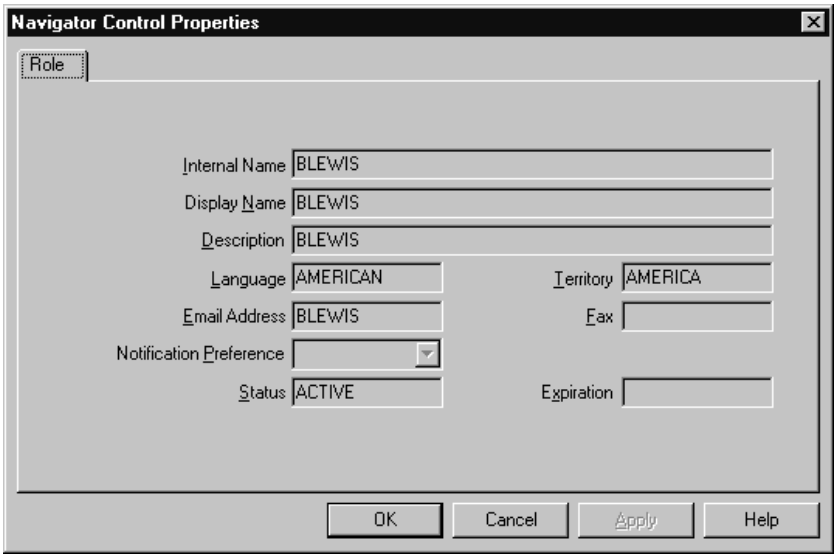


4. When you select a role from one of these poplist fields, you can also choose the Edit button to the right of the field to display the property sheet of the selected role.
5. The Role property page that appears lists read–only information about that role.

   Note: When you reopen a saved process definition in Oracle Workflow Builder, any role information that the process references automatically gets loaded even if you open the process definition from a file and are not connected to the database.

▶ **To Display the Directory Service in Oracle Workflow Builder**

1. Once you load your roles from the database in Oracle Workflow Builder, you can display your directory service information in the navigator tree. See: To Load Roles: page 5 – 20.
2. Expand the Directory Service branch in the navigator tree. All the roles that you loaded from the database appear.
3. Double–click on a role to display read–only information about that role as shown below. Note that the Directory Service branch does not currently allow you to view the participant users of a role.


Navigator Control Properties
Role
Internal Name
BLEWIS
Display Name
BLEWIS
Description
BLEWIS
Language
AMERICAN
Territory
AMERICA
Email Address
BLEWIS
Fax
Notification Preference
Status
ACTIVE
Expiration
OK
Cancel
Apply
Help

CHAPTER

6

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_HOME*>\Wf\data\<*language*>\ 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 *WF_STANDARD.ANDJOIN*.

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

Comparison Activities

The comparison activities provide a standard way to compare two numbers, dates, or text strings.

Compare Date	Use to compare the value of an item type attribute of type Date with a constant date.
Compare Number	Use to compare the value of an item type attribute of type Number with a constant number.
Compare Text	Use to compare the value of two item type attributes of type Text.

All the Comparison activities call a PL/SQL procedure named *WF_STANDARD.COMPARE.*

Activity Attributes

Each comparison activity has two activity attributes:

- Test Value—a constant number, date, or text string which to compare to a reference value.
- Reference Value—an item type attribute of type Number, Date, or Text.

The comparison activities use the Comparison lookup type for a result code. Possible values are ”Greater Than,” ”Less Than,” ”Equal,” or ”Null,” if the item type attribute is null. You can guide your workflow process based on how the value of an item type attribute compares to a given value that you set. See: To Define Activity Attribute Values: page 5 – 11.

Compare Execution Time Activity

The Compare Execution Time activity provides a standard way to compare the elapsed execution time of a process with a constant test time.

The Compare Execution Time activity calls a PL/SQL procedure named *WF_STANDARD.COMPAREEXECUTIONTIME.*

Activity Attributes

The Compare Execution Time activity has two activity attributes:

- Test Execution Time—the time, in seconds with which to compare the elapsed execution time.
- Parent Type—takes as its value, the lookup codes, "Root" or "Parent". A value of "Root" compares the test time with the elapsed execution time of the current root process. A value of "Parent" compares the test time with the elapsed execution time of just the immediate parent process, which can be a subprocess.

The activity uses the Comparison lookup type for a result code. Possible values are "Greater Than," "Less Than," "Equal," or "Null," if the test time is null. See: To Define Activity Attribute Values: page 5 – 11.

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 5 – 11.

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 8 – 49

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

Defer Thread Activity

The Defer Thread activity defers the subsequent process thread to the background queue without requiring you to change the cost of each activity in that thread to a value above the Workflow Engine threshold. This activity always interrupts the process thread by causing a disconnect to occur in the current database session, even if the thread is already deferred.

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

Launch Process Activity

The Launch Process activity lets you launch another workflow process from the current process. This activity calls the PL/SQL procedure named *WF_STANDARD.LAUNCHPROCESS*.

Activity Attributes

The Launch Process activity has six activity attributes:

- Item Type—the item type of the process to launch. Specify the item type's internal name. This activity attribute requires a value.
- Item Key—an item key for the process to launch. If you do not specify a value, the item key defaults to *<current_item_type>:<current_item_key>–<n>*, where *current_item_type* and *current_item_key* identify the current process instance, and *n* is the number of processes launched by the current process instance, starting at 1.
- Process name—the internal name of the process to launch. If a process name is not specified, the activity will check the item type selector function of the process to launch for a process name.
- User Key—a user defined key for the process to launch.
- Owner—a role designated as the owner of the process to launch.
- Defer immediate—choose between YES or NO to determine whether the process to launch should be immediately deferred to the background engine. The default is NO, so once the process is launched, it continues to execute until completion or until one of

its activities is deferred. See: To Define Activity Attribute Values: page 5 – 11.

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.

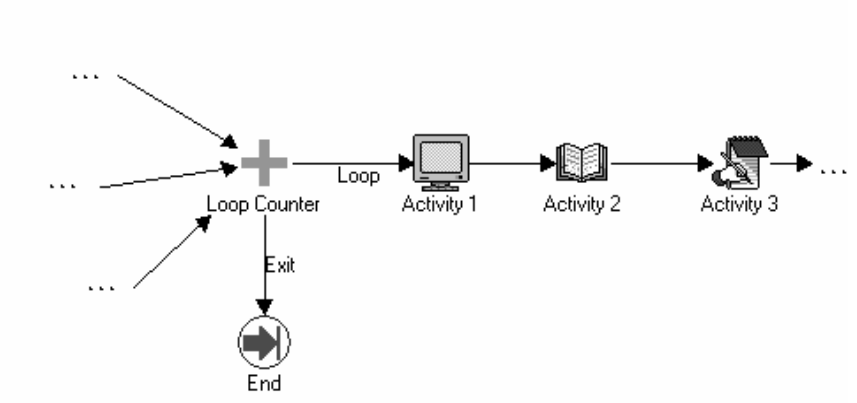
This Loop Counter activity calls the PL/SQL procedure named *WF_STANDARD.LOOPCOUNTER.*

Activity Attribute

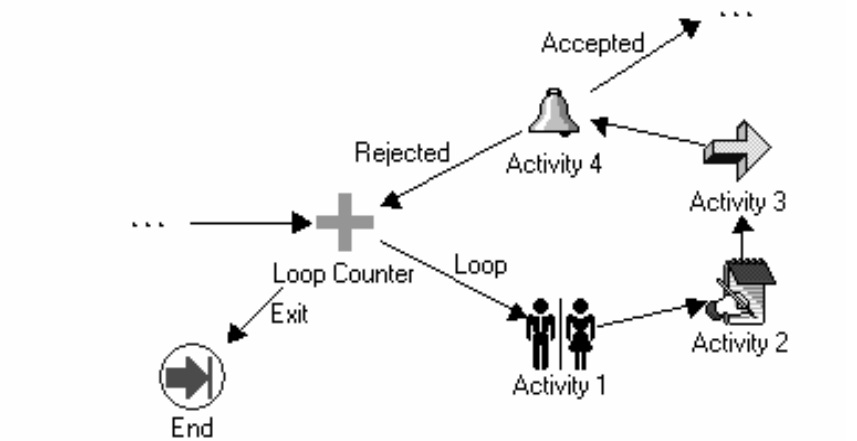
The Loop Counter activity has an activity attribute called Loop Limit. If the number of times that the Workflow Engine transitions to the Loop Counter activity is less than the value specified in Loop Limit, the Loop Counter activity will complete with a result of Loop and the engine will take the 'Loop' 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 5 – 11.



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 5 – 11.

Notify Activity

The Notify function activity lets you send a notification, where the message being sent is determined dynamically at runtime by a prior function activity. To use the Notify activity, you must model a prerequisite function activity into the process that selects one of several predefined messages for the Notify activity to send.

Attention: Since the Notify activity is locked against modifications at access level 0, you cannot change the result type from its value of <None>. Therefore, the message that the function activity dynamically selects must not have a result type, that is, it can only be an informative message that does not illicit a response.

Attention: If you want the Notify activity to send a message that requires a response, then you must copy and create your own version of the Notify activity. Since any one of several messages (with response attributes) can be sent by your version of the Notify activity, you must model into your process all the possible notification results that can be returned.

Note: If you to define an activity that always sends the same message, you should define a notification activity and not use this Notify function activity.

The Notify activity calls a PL/SQL procedure named *WF_STANDARD.NOTIFY.*

Activity Attributes

The Notify activity has two activity attributes:

- Message Name—the name of the predefined message to send. The prerequisite function activity that determines which message to send should store the name of that message in an item attribute. The Message Name activity attribute should reference that item attribute to determine the name of the message to send.
- Performer—the name of the role to which to send the notification message. If you load the roles from your database, you can select a constant role as the performer. Alternatively, you can set the performer to an item attribute that returns the name of a role at runtime.
- Expand Roles—takes as its value, the lookup codes ”Yes” or ”No”. Set Expand Roles to Yes if you wish to send an individual copy of the notification message to every user in the role. See: To Define Activity Attribute Values: page 5 – 11.

Vote Yes/No Activity

The Vote Yes/No activity lets you send a notification to a group of users in a role and tally the Yes/No responses from those users. The results of the tally determine the activity that the process transitions to next.

The Vote Yes/No activity, classified as a notification activity, first sends a notification message to a group of users and then performs a PL/SQL post–notification function to tally the users' responses (votes).

Activity Attributes

The Vote Yes/No activity has three activity attributes:

- Percent Yes—The percentage of Yes votes cast in order for the activity to complete with a result of Yes.
- Percent No—The percentage of No votes cast in order for the activity to complete with a result of No
- Voting Option—specify how the votes are tallied by selecting one of three values:
  - "Wait for All Votes"—the Workflow Engine waits until all votes are cast before tallying the results as a percentage of all the users notified. If a timeout condition occurs, the Workflow Engine calculates the resulting votes as a percentage of the total votes cast before the timeout occurred.
  - "Tally on Every Vote"—the Workflow Engine keeps a running tally of the cumulative responses as a percentage of all the users notified. If a timeout condition occurs, then the responses are tallied as a percentage of the total number of votes cast. Note that this option is meaningless if any of the custom response activity attributes have a blank value.
  - "Require All Votes"—the Workflow Engine evaluates the responses as a percentage of all users notified only after all votes are cast. If a timeout condition occurs, the Workflow Engine progresses along the standard timeout transition, or if none is available, raises an error, and does not tally any votes. See: To Define Activity Attribute Values: page 5 – 11.

See Also

Voting Activity: page 4 – 50

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 are in effect creating a separate detail process with its own unique item type and item key. You define the master/detail relationship between the two processes by making a call to the Workflow Engine *SetItemParent* API after you call the *CreateProcess* API and before you call the *StartProcess* API when you create the detail process. See: SetItemParent: page 8 – 55.

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

The Wait for Flow activity contains two activity attributes:

- Continuation Flow—specify whether this activity is waiting for a corresponding ”Master” or ”Detail” process to complete.
- Continuation Activity—specify the label of the activity node that must complete in the corresponding process before the current process continues. The default value is CONTINUEFLOW. See: To Define Activity Attribute Values: page 5 – 11.

Continue Flow Activity

Use this activity to mark the position in the corresponding detail or master process where, upon completion, you want the halted process to continue. This activity calls a PL/SQL procedure named *WF_STANDARD.CONTINUEFLOW.*

Activity Attributes

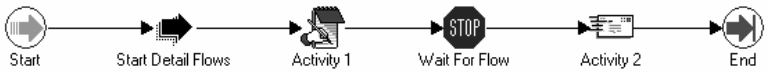
The Continue Flow activity contains two activity attributes:

- Waiting Flow—specify whether the halted process that is waiting for this activity to complete is a "Master" or "Detail" flow.
- Waiting Activity—specify the label of the activity node in the halted process that is waiting for this activity to complete. See: To Define Activity Attribute Values: page 5 – 11.

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 Activity

The Get Monitor URL activity generates the URL for the Workflow Monitor diagram window and stores it in an item attribute that you specify. This activity calls the PL/SQL procedure named *WF_STANDARD.GETURL*.

Activity Attributes

The Get Monitor URL activity has two activity attributes:

- Item Attribute—choose the name of the item attribute that you want to use to store the URL for the Workflow Monitor window.
- Administration Mode—determine how the URL displays the Workflow Monitor window. If you set Administration Mode to ”Yes”, the URL displays the Workflow Monitor in ’ADMIN’ mode, otherwise it displays the Workflow Monitor in ’USER’ mode. See: To Define Activity Attribute Values: page 5 – 11.

Concurrent Manager Standard Activities

Oracle Applications provides some generic activities you can use to control your process if you are using the version of Oracle Workflow embedded in Oracle Applications. These activities are associated with the Concurrent Manager Functions item type but can be used within any process you define:

- Execute Concurrent Program Activity
- Submit Concurrent Program Activity
- Wait for Concurrent Program Activity

The Concurrent Manager Functions item type is automatically installed on your Oracle Applications workflow server. You can also access this item type from the file fndwfaol.wft located in the $FND_TOP/*admin/import* subdirectory.

Execute Concurrent Program Activity

The Execute Concurrent Program activity is available only in the version of Oracle Workflow embedded in Oracle Applications. It submits an Oracle Applications concurrent program from your workflow process and waits for it to complete, at which point it updates the status of the activity and returns execution of the workflow process to the background engine. The concurrent program can complete with any of the following results, as defined by the Concurrent Program Status lookup type: NORMAL, ERROR, WARNING, CANCELLED, or TERMINATED. You should make sure all of these results are modelled into your process diagram.

Attention: To use the Execute Concurrent Program activity, you must ensure that the background engine is set up to run.

Attention: Generally, the context for your process' item type is always set if your session is initiated from an Oracle Applications form. However, if an interrupt occurs in your session, for example, due to a notification or blocking activity, you must ensure that the context is set by calling `FND_GLOBAL.APPS_INITIALIZE(user_id, resp_id, resp_appl_id)` in SET_CTX mode in your Selector/Callback function. See: Standard API for an Item Type Selector or Callback Function: page 7 – 7 and FNDSQF Routine APIs, *Oracle Applications Developer's Guide.*

The Execute Concurrent Program activity calls the standard Oracle Application Object Library API *FND_WF_STANDARD.EXECUTECONCPROGRAM.*

Activity Attributes

The Execute Concurrent Program activity has the following activity attributes:

- Application Short Name—Short name of the application to which the concurrent program is registered.
- Program Short Name—Short name of the concurrent program to run.
- Number of Arguments—Number of arguments required for the concurrent program.
- Item Attribute Name—Optional name of the item attribute to store the concurrent program request ID.
- Argument1, Argument2,...Argument100—Value of each concurrent program argument, ordered to match the correct syntax of the concurrent program. Up to 100 arguments are allowed, but you should only specify as many argument values as you define in the Number of Arguments activity attribute. See: To Define Activity Attribute Values: page 5 – 11.

Submit Concurrent Program Activity

The Submit Concurrent Program activity is available only in the version of Oracle Workflow embedded in Oracle Applications. It submits an Oracle Applications concurrent program from your workflow process, but does not wait for it to execute or complete. Once this activity submits a concurrent request, the Workflow Engine continues with the next activity in the process.

Attention: Generally, the context for your process' item type is always set if your session is initiated from an Oracle Applications form. However, if an interrupt occurs in your session, for example, due to a notification or blocking activity, you must ensure that the context is set by calling `FND_GLOBAL.APPS_INITIALIZE(user_id, resp_id, resp_appl_id)` in SET_CTX mode in your Selector/Callback function. See: Standard API for an Item Type Selector or Callback Function: page 7 – 7.

The Submit Concurrent Program activity calls the standard Oracle Application Object Library API *FND_WF_STANDARD.SUBMITCONCPROGRAM.*

Activity Attributes

The Submit Concurrent Program activity has the following activity attributes:

- Application Short Name—Short name of the application to which the concurrent program is registered.
- Program Short Name—Short name of the concurrent program to run.
- Number of Arguments—Number of arguments required for the concurrent program.
- Item Attribute Name—Name of the item attribute to store the concurrent program request ID.
- Argument1, Argument2,...Argument100—Value of each concurrent program argument, ordered to match the correct syntax of the concurrent program. Up to 100 arguments are allowed, but you should only specify as many argument values as you define in the Number of Arguments activity attribute. See: To Define Activity Attribute Values: page 5 – 11.

Wait for Concurrent Program Activity

The Wait for Concurrent Program activity is available only in the version of Oracle Workflow embedded in Oracle Applications. If you submit a concurrent program from your workflow process, you can use the Wait for Concurrent Program activity as a means of blocking the process from further execution until the concurrent program completes. When the concurrent program completes, this activity clears the block by updating the status of the activity and returning execution of the workflow process to the background engine. The concurrent program can complete with any of the following results, as defined by the Concurrent Program Status lookup type: NORMAL, ERROR, WARNING, CANCELLED, or TERMINATED. You should make sure all of these results are modelled into your process diagram.

Attention: To use the Wait for Concurrent Program activity, you must ensure that the background engine is set up to run.

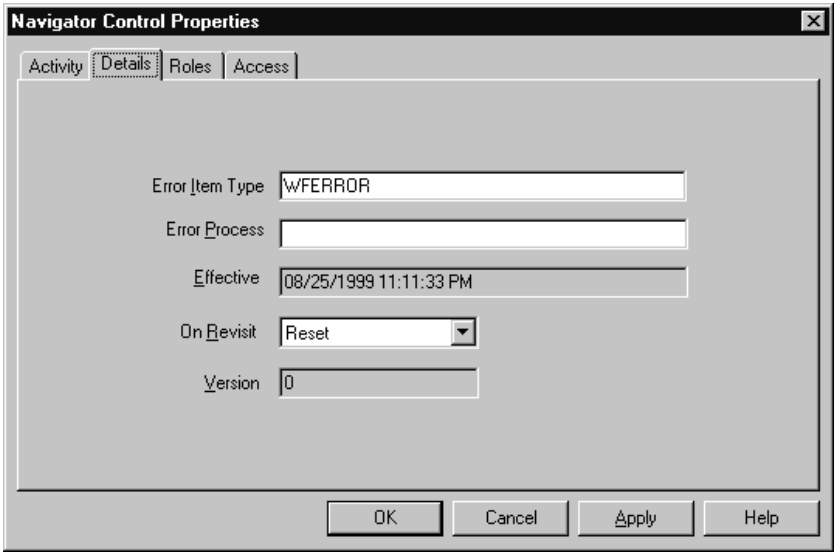
The Wait for Concurrent Program activity calls the standard Oracle Application Object Library API *FND_WF_STANDARD.WAITFORCONCPROGRAM.*

Activity Attributes

The Wait for Concurrent Program activity has one activity attribute called Request ID, which should be set to the concurrent program request ID that you are waiting for to complete. See: To Define Activity Attribute Values: page 5 – 11.

Default Error Process

At design time, Oracle Workflow permits you to specify an error handling process to execute if an error is detected in your current process. You indicate the error handling process in your process or function activity's Details property page. You specify the internal names of both the item type that owns the error handling process and the error handling process.



Oracle Workflow provides a special item type called System: Error, which contains two error processes that you can use for generic error handling in any of your processes. Note however, that you cannot customize the error processes in the System: Error item type. If you want to incorporate functionality that is not available in these error processes, you should create your own custom error handling process in your own item type.

> **Note:** Rather than relying on an error process to handle errors due to specific business rule incompatibilities, you should try to model those situations into your workflow process definition. For example, if a function activity can potentially encounter an error because a business prerequisite is not met, you might model your process to send a notification to an appropriate role to correct that situation if it occurs, so that the workflow process can progress forward. If you do not model this situation into your workflow process, and instead rely on the error to activate an error process, the entire workflow

process will have an 'Error' status and will halt until a workflow administrator handles the error.

System: Error Item Type and Item Attributes

To view the details of the System: Error 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 <*drive*>:\<*ORACLE_HOME*>\wf\Data\<*Language*> subdirectory.

The System: Error item type consists of the following item attributes:

- Error Activity ID
- Error Activity Label
- Error Assigned User
- Error Item Type
- Error Item Key
- Error Item User Key
- Error Message
- Error Name
- Error Notification ID
- Error Result Code
- Error Stack
- Error Monitor URL
- Timeout Value

These item attributes are referenced by the function and notification activities that make up the error processes called Default Error Process and Retry–only.

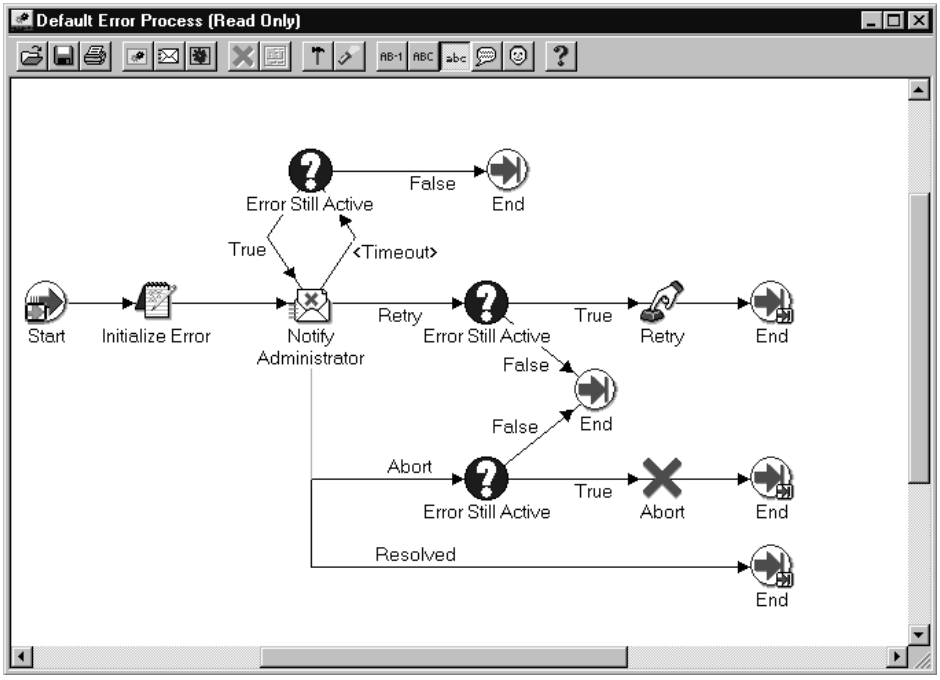
Attention: If you create a custom error handling process in your own item type, Oracle Workflow automatically sets the above item attributes when it calls your error handling process. If these item attributes do not already exist in your process, Oracle Workflow creates them. However, if you want to reference these item attributes in your error handling process, such as in a message, you must first create them as item attributes in your process' item type using Oracle Workflow Builder.

Default Error Process

DEFAULT_ERROR is the internal name of the Default Error Process. The purpose of this error handling process is to:

- send an administrator a notification when an error occurs in a process
- provide information to the administrator about the error
- allow the administrator to abort the process, retry the errored activity, or resolve the problem that caused the error to occur.

Although you cannot customize the Default Error Process, the process is flexible enough for you to customize its behavior. By defining two item type attributes called WF_ADMINISTRATOR and ERROR_TIMEOUT in your item type that calls the Default Error Process, you define who the error process sends the notification to and whether the error notification times out, respectively.



'Initialize Error' Function Activity

The Initialize Error activity calls a PL/SQL procedure named *WF_STANDARD.INITIALIZEERRORS*. This procedure determines if the item type of the errored process has an item type attribute defined

with an internal name of WF_ADMINISTRATOR. If it does, it sets the performer of the subsequent notification activity, Notify Administrator, to the role stored in WF_ADMINISTRATOR. If it does not, the subsequent notification activity remains set to the default performer, System Administrator.

By checking for an item attribute called WF_ADMINISTRATOR in your errored process' item type, the Initialize Error activity lets you specify who you want a notification to be sent to in the case of an error in your specific process without modifying the error process.

For example, suppose you have a requisition approval workflow and you want the purchasing administrator, not the system administrator, to resolve any problems that arise from this workflow. You can define an item attribute called WF_ADMINISTRATOR in the item type that owns your requisition approval workflow and set WF_ADMINISTRATOR to the purchasing administrator's role, which may be PO_ADMIN.

'Notify Administrator' Notification Activity

The Notify Administrator activity sends the Default Retry Error message to a performer (the System Administrator or whatever role is stored in your item type's WF_ADMINISTRATOR item attribute). The message indicates that an error has occurred in the specified process and that a response is needed. The response options and their resulting actions are:

- Abort the process—executes the Error Still Active activity to verify if the error is still present and if it is, calls the Abort function activity and ends the default error process.
- Retry the process—executes the Error Still Active activity to verify if the error is still present and if it is, calls the Retry function activity and ends the default error process.
- Resolved the process—ends the default error process because you addressed the errored process directly through some external means or using the embedded URL link to the Workflow Monitor.

  Note: The notification message's embedded monitor URL displays the process in error in the Workflow Monitor with full administrator privileges. You can perform actions such as retrying, skipping or rolling back part of your process to resolve the error.

The subject and body of the Default Retry Error message is as follows:

```
Subject: Error in Workflow &ERROR_ITEM_TYPE/&ERROR_ITEM_KEY
Body: An Error occurred in the following Workflow.

   Item Type = &ERROR_ITEM_TYPE
   Item Key = &ERROR_ITEM_KEY
   User Key =&USER_KEY

   Error Name = &ERROR_NAME
   Error Message = &ERROR_MESSAGE
   Error Stack = &ERROR_STACK

   Activity Id = &ERROR_ACTIVITY_ID
   Activity Label = &ERROR_ACTIVITY_LABEL
   Result Code = &ERROR_RESULT_CODE
   Notification Id = &ERROR_NOTIFICATION_ID
   Assigned User = &ERROR_ASSIGNED_USER

   &MONITOR
```

The Notify Administrator notification activity has a dynamic timeout value assigned to it. It checks the item type of the errored process for an item type attribute whose internal name is ERROR_TIMEOUT. ERROR_TIMEOUT must be an attribute of type NUMBER. The Workflow Engine interprets the value of this attribute as a relative offset from the begin date of the activity, in the unit of MINUTES to determine the timeout value of Notify Administrator. If ERROR_TIMEOUT contains a null value, a value of zero, or is not defined at all, then Notify Administrator has no timeout.

'Error Still Active' Function Activity

The Workflow Engine initiates the Error Still Active function activity if the Notify Administrator activity times out or returns Abort or Retry as a result.

The Error Still Active activity calls a PL/SQL procedure called *WF_STANDARD.CHECKERRORACTIVE*. The purpose of the Error Still Active activity is to determine whether the errored process is still in error before continuing with the error handling. If it is, Error Still Active returns TRUE and the Workflow Engine takes the appropriate transition to either send another notification or abort or retry the errored process. If the errored process is no longer in error, this activity returns False and the error handling process ends, as modelled in the process diagram.

'Retry' Function Activity

The Retry function activity executes the PL/SQL procedure *WF_STANDARD.RESETERROR* to clear the activity that was in error and run it again. This procedure calls the *WF_ENGINE.HandleError* API to rerun the activity.

'Abort' Function Activity

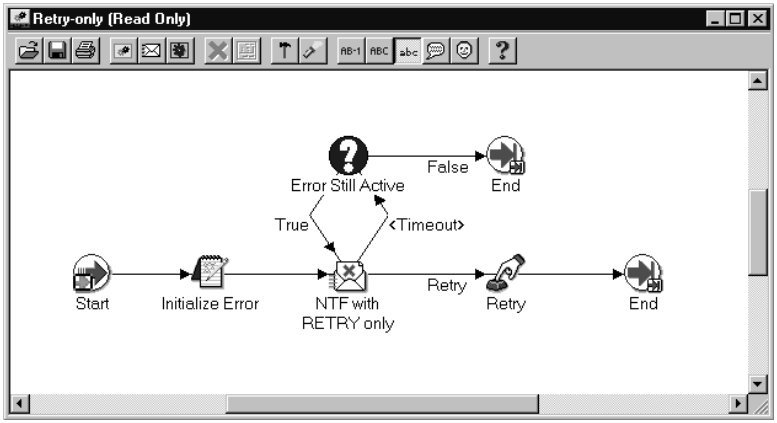
The Abort function activity executes the PL/SQL procedure *WF_STANDARD.ABORTPROCESS*, which in turn calls the *WF_ENGINE.AbortProcess* API to abort the process that encountered the error.

See Also

Workflow Core APIs: page 8 – 58

Retry–only Process

RETRY_ONLY is the internal name of the Retry–only error process. The purpose of this error handling process is to alert an administrator when an error occurs in a process and prompt the administrator to retry the process in error.



'Initialize Error' Function Activity

The Initialize Error activity calls a PL/SQL procedure named *WF_STANDARD.INITIALIZEERRORS*. This procedure determines if the item type of the errored process has an item type attribute defined with an internal name of WF_ADMINISTRATOR. If it does, it sets the performer of the subsequent notification activity, NTF with RETRY Only, to the role stored in WF_ADMINISTRATOR. If it does not, the subsequent notification activity remains set to the default performer, System Administrator.

By checking for an item attribute called WF_ADMINISTRATOR in your errored process' item type, the Initialize Error activity lets you specify who you want a notification to be sent to in the case of an error in your specific process without modifying the error process.

For example, suppose you have a requisition approval workflow and you want the purchasing administrator, not the system administrator, to resolve any problems that arise from this workflow. You can define an item attribute called WF_ADMINISTRATOR in the item type that owns your requisition approval workflow and set WF_ADMINISTRATOR to the purchasing administrator's role, which may be PO_ADMIN.

'NTF with RETRY Only' Notification Activity

The NTF with RETRY Only activity sends the Retry As Only Option message to a performer (the System Administrator or whatever role is stored in your item type's WF_ADMINISTRATOR item attribute). The message indicates that an error has occurred in the specified process and prompts the administrator to retry the activity that errored. The error process then transitions to the Retry function activity and ends the Retry–only error process.

> **Note:** The notification message's embedded URL link displays the process in error in the Workflow Monitor with full administrator privileges. You can perform actions such as retrying, skipping or rolling back part of your process within the Workflow Monitor to resolve the error.

The subject and body of the Retry As Only Option message is as follows:

```
Subject: Error in Workflow &ERROR_ITEM_TYPE/&ERROR_ITEM_KEY
Body: An Error occurred in the following Workflow.

   Item Type = &ERROR_ITEM_TYPE
   Item Key = &ERROR_ITEM_KEY
```

```
User Key =&USER_KEY

Error Name = &ERROR_NAME
Error Message = &ERROR_MESSAGE
Error Stack = &ERROR_STACK

Activity Id = &ERROR_ACTIVITY_ID
Activity Label = &ERROR_ACTIVITY_LABEL
Result Code = &ERROR_RESULT_CODE
Notification Id = &ERROR_NOTIFICATION_ID
Assigned User = &ERROR_ASSIGNED_USER

&MONITOR
```

The NTF with RETRY Only notification activity has a dynamic timeout value assigned to it. It checks the item type of the process in error for an item attribute that has an internal name called ERROR_TIMEOUT. ERROR_TIMEOUT must be an attribute of type NUMBER. The Workflow Engine interprets the timeout value of this attribute as a relative offset from the begin date of the activity, in the unit of MINUTES. If ERROR_TIMEOUT contains a null value, a value of zero, or is not defined at all, then NTF with RETRY Only has no timeout.

'Error Still Active' Function Activity

The Workflow Engine initiates the Error Still Active function activity if the NTF with RETRY Only activity times out.

The Error Still Active activity calls a PL/SQL procedure called *WF_STANDARD.CHECKERRORACTIVE*. The purpose of the Error Still Active activity is to determine whether the errored process is still in error before continuing with the error handling. If it is, Error Still Active returns TRUE and the Workflow Engine transitions back to the NTF with RETRY Only notification activity to send another notification to the administrator. If the errored process is no longer in error, this activity returns False and the error handling process ends, as modelled in the process diagram.

'Retry' Function Activity

The Retry function activity executes the PL/SQL procedure *WF_STANDARD.RESETERROR* to clear the activity that was in error and run it again. This procedure calls the *WF_ENGINE.HandleError* API to rerun the activity.

See Also

Workflow Core APIs: page 8 – 58

CHAPTER

7

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 or notification activities in an Oracle Workflow process should follow this standard API format so that the Workflow Engine can properly execute the activity.

Attention: The Workflow Engine traps errors produced by function activities by setting a savepoint before each function activity. If an activity produces an unhandled exception, the engine performs a rollback to the savepoint, and sets the activity to the ERROR status. For this reason, you should never commit within the PL/SQL procedure of a function activity. The Workflow Engine never issues a commit as it is the responsibility of the calling application to commit.

For environments such as database triggers or distributed transactions that do not allow savepoints, the Workflow Engine automatically traps "Savepoint not allowed" errors and defers the execution of the activity to the background engine.

The example in this section is numbered with the notation 1⇒ for easy referencing. The numbers 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,
                                                 resultout    out varchar2  ) is
2⇒  <local declarations>
3⇒  begin
    if ( funcmode = 'RUN' ) then
      <your RUN executable statements>
      resultout := 'COMPLETE:<result>';
      return;
    end if;
4⇒  if ( funcmode = 'CANCEL' ) then
      <your CANCEL executable statements>
      resultout := 'COMPLETE';
      return;
    end if;
5⇒  if ( funcmode = 'RESPOND' ) then
```

```
      <your RESPOND executable statements>
      resultout := 'COMPLETE';
      return;
    end if;
6⇒  if ( funcmode = 'FORWARD' ) then
      <your FORWARD executable statements>
      resultout := 'COMPLETE';
      return;
    end if;
7⇒  if ( funcmode = 'TRANSFER' ) then
      <your TRANSFER executable statements>
      resultout := 'COMPLETE';
      return;
    end if;
8⇒  if ( funcmode = 'TIMEOUT' ) then
      <your TIMEOUT executable statements>
      resultout := 'COMPLETE';
      return;
    end if;
9⇒  if ( funcmode = '<other funcmode>' ) then
      resultout := ' ';
      return;
    end if;
10⇒ exception
      when others then
      WF_CORE.CONTEXT ('<package name>', '<procedure name>', <itemtype>, <itemkey>,
          to_char(<actid>), <funcmode>);
      raise;
11⇒ 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 activity. If the activity is a function activity, the mode is either `'RUN'` or `'CANCEL'`. If the activity is a notification activity, with a post–notification function, then the mode can be 'RESPOND', 'FORWARD', 'TRANSFER', or 'TIMEOUT'. Other execution modes may be added in the future.

resultout If a result type is specified in the Activities properties page for the activity in the Oracle Workflow Builder, this parameter represents the expected result that is returned when the procedure completes. The possible results are:

COMPLETE:<*result_code*>—activity completes with the indicated result code. The result code must match one of the result codes specified in the result type of the function activity.

WAITING—activity is pending, waiting on another activity to complete before it completes. An example is the Standard 'AND' activity.

DEFERRED:<*date*>—activity is deferred to a background engine for execution until a given date. <*date*> must be of the format:

```
to_char(<date_string>, wf_engine.date_format)
```

NOTIFIED:<*notification_id*>:<*assigned_user*>—an external entity is notified that an action must be performed. A notification ID and an assigned user can optionally be returned with this result. Note that the external entity must call *CompleteActivity*( ) to inform the Workflow engine when the action completes.

ERROR:<*error_code*>—activity encounters an error and returns the indicated error code.

2⇒ This section declares any local arguments that are used within the procedure.

3⇒ The procedure body begins in this section with an `IF` statement. This section contains one or more executable statements that run if the value of `funcmode` is `'RUN'`. One of the executable statements can return a result for the procedure. For example, a result can be 'COMPLETE:APPROVED'.

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 8 – 7 and Loop Counter Activity: page 6 – 7.

- 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 8 – 28.

5⇒ This section is needed only for post–notification functions. Use this section to include execution statements that run if the value of funcmode is `'RESPOND'`, that is, when a RESPOND operation is performed. For example, include execution statements that validate the response of the notification. See: Post–notification functions: page 8 – 10.

6⇒ This section is needed only for post–notification functions. Use this section to include execution statements that run if the value of funcmode is `'FORWARD'`, that is, when a notification's state changes to 'FORWARD'. For example, include execution statements that validate the role to which the notification is being forwarded.

7⇒ This section is needed only for post–notification functions. Use this section to include execution statements that run if the value of funcmode is `'TRANSFER'`, that is, when a notification's state changes to

'TRANSFER'. For example, include execution statements that validate the role to which the notification is being transferred.

> **Note:** For 'RESPOND', 'FORWARD', and 'TRANSFER' funcmodes, the resultout parameter is ignored, except if the returned value looks something like 'ERROR%'. Therefore, if you do not want the Respond, Forward or Transfer operation to occur after having executed your post–notification function, you can do one of two things:
>
> – Return 'ERROR:<*errcode*>' in the resultout parameter to convert it to a generic exception with the errcode mentioned in the message.
>
> – Raise an exception directly in your procedure with a more informative error message. See: Post–notification Functions: page 8 – 10 and Notification Model: 8 – 141

8⇒ This section should only be used for post–notification functions associated with Voting activities. Use this section to include execution statements that run if a Voting activity times out. For example, if the voting activity times out before all recipients respond, you can include logic that completes the activity and determines how to interpret the responses based on the current response pool. For non–Voting activities that time out, you should model any subsequent behavior in your process diagram using a <Timeout> transition to another activity.

9⇒ This section handles execution modes other than `'RUN'`, `'CANCEL'`, `'RESPOND'`, `'FORWARD'`, `'TRANSFER'`, or `'TIMEOUT'`. Other execution modes may be added in the future. Since your activity does not need to implement any of these other possible modes, it should simply return null.

10⇒ 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 8 – 65.

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 needs to execute, before a new database session begins.
- TEST_CTX—to determine if the current item type context information is correct before executing a function. For example, the selector/callback function in TEST_CTX mode lets you check if a form can be launched with the current context information just before the Notification Details web page launches a reference form. If the context is incorrect, the form cannot be launched and a message is displayed to that effect. See: To View the Details of a Notification: page 10 – 18.

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,
                                                resultout      in out varchar2) is
2⇒  <local declarations>
3⇒  begin
    if ( command = 'RUN' ) then
      <your RUN executable statements>
      resultout := '<Name of process to run>';
      return;
    end if;
4⇒  if ( command = 'SET_CTX' ) then
      <your executable statements for establishing context information>
      return;
    end if;
5⇒  if ( command = 'TEST_CTX' ) then
      <your executable statements for testing the validity of the current context information>
      resultout := '<TRUE or FALSE> ';
      return;
    end if;
6⇒  if ( command = '<other command>' ) then
      resultout := ' ';
      return;
    end if;
7⇒  exception
      when others then
      WF_CORE.CONTEXT ('<package name>', '<procedure name>', <itemtype>, <itemkey>,
           to_char(<actid>), <command>);
      raise;
8⇒  end   <procedure name>;
```

1⇒ When the Workflow Engine calls the selector/callback function, it passes four parameters to the procedure and may expect a result when the procedure completes. The parameters are defined here:

itemtype The internal name for the item type. Item types are defined in the Oracle Workflow Builder.

itemkey A string that represents a primary key generated by the workflow–enabled application for the item type. The string uniquely identifies the item within an item type.

actid The ID number of the activity that this procedure is called from. Note that this parameter is always null if the procedure is called with the `'RUN'` command to execute the selector functionality.

command The command that determines how to execute the selector/callback function. Either `'RUN'`, `'SET_CTX'`, or `'TEST_CTX'`. Other commands may be added in the future.

resultout A result may be returned depending on the command that is used to call the selector/callback function.

If the function is called with `'RUN'`, the name of the process to run must be returned through the resultout parameter. If the function is called with `'SET_CTX'`, then no return value is expected. If the function is called with `'TEST_CTX'`, then the code must return 'TRUE' if the context is correct 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.

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 Notification Details web page 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.

6⇒ This section handles execution modes other than `'RUN'`, `'SET_CTX' or 'TEST_CTX'` as others may be added in the future. Since your function does not need to implement any of these other possible commands, it should simply return null.

7⇒ This section calls WF_CORE.CONTEXT( ) if an exception occurs, so that you can include context information in the error stack to help you locate the source of an error. See: CONTEXT: page 8 – 65.

Standard API for a "PL/SQL" Document

You can integrate a document into a workflow process by defining an attribute of type document for an item type, message, or activity. One type of document that Oracle Workflow supports is a "PL/SQL" document. The document–type attribute that you create tells Oracle Workflow how to construct a dynamic call to a PL/SQL procedure that generates the document. You can embed a PL/SQL document–type message attribute in a message body to display the document in a notification.

The PL/SQL procedure that generates the document must have the following standard API:

```
procedure <procedure name>  (  document_id     in varchar2,
                               display_type    in varchar2,
                               document        in out varchar2,
                               document_type   in out varchar2)
```

The arguments for the procedure are as follows:

document_id A string that uniquely identifies a document. This is the same string as the value that you specify in the default value field of the Attribute property page for a "PL/SQL" document (plsql:*<procedure>*/*<document_identifier>*). `<procedure>` should be replaced with the PL/SQL package and procedure name in the form of package.procedure. The phrase `<document_identifier>` should be replaced with the PL/SQL argument string that you want to pass directly to the procedure. The argument string should identify the document. For example: `plsql:po_wf.show_req/2034`. If you wish to generate the PL/SQL argument string value dynamically, create another item attribute, and reference that item attribute as "&*item_attribute*" in place of the PL/SQL argument string. Then before any activity that references this other item attribute gets executed, call the WF_ENGINE.SetItemAttribute API to dynamically set the PL/SQL argument string value. For example: `plsql:po_wf.show_req/&POREQ_NUMBER`.

display_type One of three values that represents the content type used for the notification presentation, also referred to as the requested type:

text/plain—the document is embedded inside a plain text representation of the notification as viewed from an E–mail message. The entire 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 – 56.

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. If you include top level HTML tags accidentally, Oracle Workflow removes the tags for you when the document attribute is referenced in a message body. Note that the procedure can alternatively generate a plain text document, as the notification system can automatically surround plain text with the appropriate HTML tags to preserve formatting.

' '—the document is presented as a separate attachment to the notification. Any content type may be returned.

document The outbound text buffer where up to 32K of document text is returned.

document_type The outbound text buffer where the document content type is returned. Also referred to as the returned type. If no type is supplied, then 'text/plain' is assumed.

CHAPTER

8

Oracle Workflow APIs

This chapter describes the APIs for Oracle Workflow. The APIs consist 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 8 – 14
- WF_CORE: page 8 – 58
- WF_PURGE: page 8 – 68
- WF_DIRECTORY: page 8 – 77
- WF_PREF: page 8 – 100
- WF_MONITOR: page 8 – 101
- Oracle Workflow Views: page 8 – 108
- WF_QUEUE: page 8 – 113
- FND_DOCUMENT_MANAGEMENT: page 8 – 134
- WF_NOTIFICATIONS: page 8 – 141

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. Since the engine is embedded inside Oracle8, if the Workflow server goes down for any reason, Oracle8 is able to manage the recovery and transactional integrity of any workflow transactions that were running at the time of the failure.

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 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: page 8 – 16
- StartProcess: page 8 – 22
- CompleteActivity: page 8 – 49
- AssignActivity: page 8 – 52
- HandleError: page 8 – 53
- SuspendProcess: page 8 – 26
- ResumeProcess: page 8 – 27
- AbortProcess: page 8 – 28

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

- Active—activity is running.

- Complete—activity completed normally.
- Waiting—activity is waiting to run.
- Notified—notification activity is delivered and open.
- Deferred—activity is deferred.
- Error—activity completed with error.
- Suspended—activity is suspended.

Attention: The Workflow Engine traps errors produced by function activities by setting a savepoint before each function activity. If an activity produces an unhandled exception, the engine performs a rollback to the savepoint, and sets the activity to the ERROR status. For this reason, you should never commit within the PL/SQL procedure of a function activity. The Workflow Engine never issues a commit as it is the responsibility of the calling application to commit.

For environments such as database triggers or distributed transactions that do not allow savepoints, the Workflow Engine automatically traps "Savepoint not allowed" errors and defers the execution of the activity to the background engine.

Oracle Workflow Java Interface

The Oracle Workflow Java interface provides a means for any Java program to integrate with Oracle Workflow. The Oracle Workflow Engine and Notification APIs are accessible through public server PL/SQL packages and published views. The Oracle Workflow Java interface exposes those APIs as Java methods that can be called by any Java program to communicate with Oracle Workflow. The Java methods directly reference the WF_ENGINE and WF_NOTIFICATION PL/SQL package procedures and views and communicate with the Oracle Workflow database through JDBC.

The methods are defined within the EngineAPI class and the NotificationAPI class, in the Java package 'oracle.apps.fnd.wf.engine'. If a Workflow Engine or Notification API has a corresponding Java method, its Java method syntax is displayed immediately after its PL/SQL syntax in the documentation.

Oracle Workflow Context

Each Java method requires an input of a WFContext object that consists of database connectivity information which you instantiate and

resource context information that the WFContext class instantiates. To call a Workflow Engine Java API in your Java program, you must first instantiate a database variable of class WFDB with your database username, password and alias, then instantiate wCtx with the database variable.

Sample Java Program

Oracle Workflow provides an example Java program that illustrates how to call most of the Workflow Engine Java APIs. The Java program is named WFTest. It calls the various Java APIs to launch the WFDEMO process, set/get attributes and suspend, resume and abort the process. To initiate this program, make sure you define CLASSPATH and LD_LIBRARY_PATH for the Oracle JDBC implementation and a supported version of Oracle before running the WFTest java program. For example, on UNIX, use the following commands:

```
setenv CLASSPATH
<WF_installed_directory>/java/oracle/apps/fnd/wf/jar/
wfapi.jar:${ORACLE_HOME}/jdbc/lib/classes111.zip


setenv LD_LIBRARY_PATH ${ORACLE_HOME}/lib:${LD_LIBRARY_PATH}
```

To initiate the WFTEST program, run Java against 'oracle.apps.fnd.wf.WFTest'. For example, on UNIX, enter the following statement on the command line:

```
$java oracle.apps.fnd.wf.WFTest
```

See Also

Workflow Engine APIs: page 8 – 14

Additional Workflow Engine Features

In addition to managing a process, the Workflow Engine also supports the following features:

- Completion Processing: page 8 – 6
- Deferred Processing: page 8 – 6
- Error Processing: page 8 – 7

- Looping: page 8 – 7
- Version/Effective Date: page 8 – 9
- Item Type Attributes: page 8 – 9
- Post–notification functions: page 8 – 10
- Forced Synchronous Processes: page 8 – 11

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 when you review your process status through the Workflow Monitor or the Oracle Applications Status form.

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.

Each activity has a user–defined processing cost. You can set this cost to be small if the activity merely sets an item attribute, or you may set it to be very high if the activity performs a resource–intensive operation. If the result of a completed activity triggers the execution of a costly function, you might want to defer the execution of that costly function to a background engine.

The Workflow Engine integrates with Oracle8 Advanced Queues to carry out deferred processing. If a function activity has a cost that exceeds the main threshold cost, the Workflow Engine marks that

activity with a status of 'DEFERRED' in the workflow status tables and enqueues the deferred activity to a special queue for deferred activities. A special queue processor called the background engine checks and processes the activities in the 'deferred' queue. The order in which the deferred activities are processed are based on the first in, first out ordering of an activity' enqueue time. At least one background engine must be set up to run at all times. Some sites may have multiple background engines operating at different thresholds or item type specifications, to avoid tying up all background processing with long–running 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 6 – 20.

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 4 – 48.

The Workflow Engine traps errors produced by function activities by setting a savepoint before each function activity. If an activity produces an unhandled exception, the engine performs a rollback to the savepoint, and sets the activity to the ERROR status.

> **Note:** For this reason, you should never commit within the PL/SQL procedure of a function activity. The Workflow Engine never issues a commit as it is the responsibility of the calling application to commit.

The Workflow Engine then attempts to locate an error process to run by starting with the activity which caused the error, and then checking each parent process activity until an associated error process is located.

Looping

Looping occurs when the completion of an activity causes a transition to another activity that has already been completed. The first activity

that gets detected as a revisited activity is also called a loop point or pivot activity. The Workflow Engine can handle a revisited activity in one of three ways:

- Ignore the activity, and stop further processing of the thread, so in effect, the activity can only run once.
- Reset the loop to the loop point before reexecuting by first running logic to undo the activities within the loop.
- Reexecute the loop point and all activities within the loop without running any logic.

Every activity has an On Revisit poplist field in its Oracle Workflow Builder Details property page. The On Revisit poplist lets you specify the behavior of the Workflow Engine when it revisits the activity in a workflow process. You can set the field to Ignore, Reset, or Loop.

Setting On Revisit to Ignore 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 you to implement a "logical OR"–type of activity which is transitioned to multiple times, but completes after the first transition only.

Setting On Revisit to Reset for an activity is useful when you want to reexecute activities in a loop, but you want to first reset the status of the activities in the loop. Reset causes the Workflow Engine to do the following:

- Build a list of all activities visited following the pivot activity.
- Traverse the list of activities, cancelling each activity and resetting its status.

Cancelling an activity is similar to executing the activity, except that the activity is executed in "CANCEL" mode rather than "RUN" mode. You can include compensatory logic in "CANCEL" mode that reverses any operation performed earlier in "RUN" mode. Note that in the case of a notification activity that has On Revisit set to Reset, if the engine encounters that notification activity again, it cancels the existing notification that was sent earlier.

Setting On Revisit to Loop for an activity is useful when you want to simply reexecute activities in a loop without resetting the status of the activities in the loop. Loop causes the Workflow Engine to reexecute the activity in "RUN" mode without executing any "CANCEL" mode logic for the activity.

Version / Effective Date

Certain workflow objects in a process definition are marked with a version number so that more than one version of the object can be in use at any one time. These objects are:

- Activities—notifications, functions, and processes

  Note: Although function activities support versioning, the underlying PL/SQL code does not, unless implemented by your developer. You should avoid adding references to new activity attributes or returning result lookup codes not modelled by existing activities in your PL/SQL code.

- Activity attributes
- Process activity nodes
- Activity attribute values
- Activity transitions

If you edit and save any of the above objects in Oracle Workflow Builder to the database, Oracle Workflow automatically creates a new version of that object or the owning object by incrementing the version number by one. If you save edits to any of the above objects to an existing file, then the original objects are overwritten. If you have a process instance that is still running and you upgrade the underlying workflow definition in your Workflow server, the process instance continues to run using the version of the workflow object definitions with which it was originally initiated.

An effective date controls which version of a definition the engine uses when executing a process. When you edit a process, you can save it with an immediate or future effective date. Any new process instance that is initiated always uses the version that is specified to be effective at that point in time. See: Opening and Saving Item Types: page 3 – 12.

Note that Oracle Workflow does not maintain versions for other workflow objects. Any modifications that you save to the following objects overwrites the existing definition of the object:

- Item attributes
- Messages
- Lookup types

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.

Post–Notification Functions

You can associate a post–notification function with a notification activity. The Workflow Engine executes the post–notification function in response to an update of the notification's state after the notification is delivered. For example, you can specify a post–notification function that executes when the notification recipient forwards or transfers the notification. The post–notification function could perform backend logic to either validate the legitimacy of the forward/transfer or execute some other supporting logic.

The post–notification function should be a PL/SQL procedure written to the same API standards required for function activities. See: Standard API for PL/SQL Procedures Called by Function Activities: page 7 – 2.

When you specify a post–notification function, the Workflow Engine first sets the context information to use with the function via the following two global engine variables:

- WF_ENGINE.context_nid = *notification_ID*.
- WF_ENGINE.context_text = *new recipient_role*, if the post–notification function gets called in FORWARD or TRANSFER mode. This variable is the new role to which the notification gets forwarded/transferred.

  Note: WF_ENGINE.context_text = *responder*, if the post–notification function gets called in RESPOND mode. Note that the value of *responder* varies depending on the notification interface the recipient uses to respond. If the recipient responds using the Notification web page, *responder* is set to the role name of the responder. If the recipient responds via E–mail, *responder* is set to "email:*responder_email_address*".

You may reference these global engine variables in your PL/SQL function.

Then when the notification's state changes, a notification callback function executes the post–notification function in the mode that matches the notification's state: RESPOND, FORWARD, or TRANSFER. When the post–notification function completes, the Workflow Engine erases the two global engine variables.

As a final step, if the post–notification function is run in TRANSFER mode and Expand Roles is not checked for the notification activity, the

Workflow Engine sets the assigned user for the notification to the new role name specified.

Attention: If the post–notification function returns ERROR:<*errcode*> as a result or raises an exception, the Workflow Engine aborts the respond, forward, or transfer operation. For example, if the post–notification function is executed in FORWARD mode and it raises an exception because the role being forwarded to is invalid, an error is displayed to the user and the Forward operation does not get executed. The notification recipient is then prompted again to take some type of action.

See Also

Notification Model: page 8 – 141

Forced Synchronous Processes

A workflow process is executed synchronously when it includes consecutive function activities in a single thread that are not deferred to the background engine. The Workflow Engine passes control back to the calling application when it completes those activities and reaches a notification, block, wait, deferred or end activity.

The Workflow Engine also supports a special class of synchronous processes called forced synchronous processes. A forced synchronous process completes in a single SQL session from start to finish and never inserts into or updates any database tables. As a result, the execution speed of a forced synchronous process is significantly faster than a typical synchronous process.

There may be cases when your application requires a forced synchronous process to generate a specific result quickly and recording an audit trail is not a concern. For example, in Oracle Applications, several products require Account Generator workflows to generate a meaningful flexfield code derived from a series of concatenated segments pulled from various tables. The Account Generator workflows are forced synchronous processes that compute and pass back completed flexfield codes to the calling applications instantaneously.

To create a forced synchronous process, you need to set the itemkey of your process to #SYNCH or wf_engine.eng_synch, which returns the #SYNCH constant when you call the necessary WF_ENGINE APIs. Since a forced synchronous process never writes to the database, using

a non–unique itemkey such as #SYNCH is not an issue. Your process definition, however, must adhere to the following set of restrictions:

- No notification activities are allowed.
- Limited blocking–type activities are allowed. A process can block and restart with a call to WF_ENGINE.CompleteActivity only if the blocking and restarting activities:
  - Occur in the same database session.
  - Contain no intervening calls to Oracle Workflow.
  - Contain no intervening commits.
- No Error Processes can be assigned to the process or the process' activities.
- Each function activity behaves as if On Revisit is set to Loop, and is run in non–cancelling mode, regardless of its actual On Revisit setting. Loops are allowed in the process.
- No Master/Detail coordination activities are allowed.
- No parallel flows are allowed in the process, as transitions from each activity must have a distinct result. This also means that no <Any> transitions are allowed since they cause parallel flows.
- None of the following Standard activities are allowed:
  - And
  - Block (restricted by the conditions stated in the Limited Blocking bullet point above.)
  - Defer Thread
  - Wait
  - Continue Flow/Wait for Flow
  - Role Resolution
  - Voting
  - Compare Execution Time
  - Notify
- No use of the background engine, that is, activities are never deferred.
- No data is ever written to the Oracle Workflow tables and as a result:
  - The process cannot be viewed from the Workflow Monitor.

  - No auditing is available for the process.

- Only the following WF_ENGINE API calls are allowed to be made, and in all cases, the itemkey supplied to these APIs must be specified as #SYNCH or wf_engine.eng_synch:
  - WF_ENGINE.CreateProcess
  - WF_ENGINE.StartProcess
  - WF_ENGINE.GetItemAttribute
  - WF_ENGINE.SetItemAttribute
  - WF_ENGINE.GetActivityAttribute
  - WF_ENGINE.CompleteActivity (for the limited usage of blocking–type activities)
- WF_ENGINE API calls for any item besides the item for the current synchronous item are not allowed.

Attention: If you encounter an error from an forced synchronous process, you should rerun the process with a unique item key in asynchronous mode and check the error stack using the Workflow Monitor or the script wfstat.sql. If the synchronous process completes successfully, the error you encountered in the forced synchronous process is probably due to a violation of one of the above listed restrictions. See: Wfstat.sql: page 14 – 13.

Workflow Engine APIs

The Workflow Engine 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.

Many of these Workflow Engine APIs also have corresponding Java methods that you can call from any Java program to integrate with Oracle Workflow. The following list indicates whether the Workflow Engine APIs are available as PL/SQL functions/procedures, as Java methods, or both.

Attention: Java is case–sensitive and all Java method names begin with a lower case letter to follow Java naming conventions.

- CreateProcess: page 8 – 16—PL/SQL and Java
- SetItemUserKey: page 8 – 18—PL/SQL
- GetItemUserKey: page 8 – 19—PL/SQL
- GetActivityLabel: page 8 – 20—PL/SQL
- SetItemOwner: page 8 – 21—PL/SQL
- StartProcess: page 8 – 22—PL/SQL and Java
- LaunchProcess: page 8 – 24—PL/SQL and Java
- SuspendProcess: page 8 – 26—PL/SQL and Java
- ResumeProcess: page 8 – 27—PL/SQL and Java
- AbortProcess: page 8 – 28—PL/SQL and Java
- CreateForkProcess: page 8 – 30—PL/SQL
- StartForkProcess: page 8 – 32—PL/SQL
- Background: page 8 – 33—PL/SQL
- AddItemAttr: page 8 – 35—PL/SQL and Java
- SetItemAttribute: page 8 – 36—PL/SQL and Java
- SetItemAttrDocument: page 8 – 38—PL/SQL and Java
- getItemTypes: page 8 – 40—Java
- GetItemAttribute: page 8 – 41—PL/SQL
- GetItemAttrDocument: page 8 – 42—PL/SQL
- getItemAttributes: page 8 – 43—Java

- GetItemAttrInfo: page 8 – 44—PL/SQL
- GetActivityAttrInfo: page 8 – 45—PL/SQL
- GetActivityAttribute: page 8 – 46—PL/SQL
- BeginActivity: page 8 – 47—PL/SQL
- CompleteActivity: page 8 – 49—PL/SQL
- AssignActivity: page 8 – 52—PL/SQL
- HandleError: page 8 – 53—PL/SQL
- SetItemParent: page 8 – 55—PL/SQL
- ItemStatus: page 8 – 56—PL/SQL and Java
- getProcessStatus: page 8 – 57—Java

See Also

Standard API for PL/SQL Procedures Called by a Function Activities: page 7 – 2

CreateProcess

PL/SQL Syntax

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

Java Syntax

```
public static boolean createProcess
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String process)
```

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)

wCtx Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.

itemtype A valid item type. Item types are defined in the Workflow Builder.

itemkey A string derived usually 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.

Note: You can pass #SYNC as the itemkey to create a forced synchronous process. See: Forced Synchronous Processes: page 8 – 11.

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. If you do not specify a selector function and this argument is null, an error will be raised.

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.

Attention: The Workflow Engine always issues a savepoint before executing each activity in a process so that it can rollback to the previous activity in case an error occurs. For environments such as database triggers or distributed transactions that do not allow savepoints, the Workflow Engine automatically traps "Savepoint not allowed" errors and defers the execution of the activity. If you initiate a workflow process from a database trigger, the Workflow Engine immediately defers the initial start activities to a background engine, so that they are no longer executing from a database trigger.

SetItemUserKey

PL/SQL 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 user interface components of Oracle Workflow.

Arguments (input)

itemtype	A valid item type.
itemkey	A string generated usually 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 8 – 16.
userkey	The user key to assign to the item identified by the specified item type and item key.

GetItemUserKey

PL/SQL Syntax

```
function GetItemUserKey
    (itemtype in varchar2,
     itemkey in varchar2)
     return varchar2;
```

Description

Returns the user–friendly key assigned to an item in a process, identified by an item type and item key. The user key is a user–friendly identifier to locate items in the Workflow Monitor and other user interface components of Oracle Workflow.

Arguments (input)

itemtype A valid item type.

itemkey A string generated usually 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 8 – 16.

GetActivityLabel

PL/SQL 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

PL/SQL 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. Typically, the role that initiates a transaction is assigned as the process owner, so that any participant in that role can find and view the status of that process instance in the Workflow Monitor.

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

PL/SQL Syntax

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

Java Syntax

```
public static boolean startProcess
    (WFContext wCtx,
     String itemType,
     String itemKey)
```

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)

wCtx Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.

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: CreateProcess: page 8 – 16.

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 0.5).

or

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

```
begin
       save_threshold := WF_ENGINE.threshold;
       WF_ENGINE.threshold := -1;
       WF_ENGINE.CreateProcess(...);
       WF_ENGINE.StartProcess(...);
--Always reset threshold or all activities in this
--session will be deferred.
       WF_ENGINE.threshold := save_threshold;
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.

Note: You can pass #SYNC as the itemkey to create a forced synchronous process. See: Forced Synchronous Processes: page 8 – 11.

LaunchProcess

PL/SQL Syntax

```
procedure LaunchProcess
    (itemtype in varchar2,
     itemkey in varchar2,
     process in varchar2 default '',
     userkey in varchar2 default '',
     owner in varchar2 default '');
```

Java Syntax

```
public static boolean LaunchProcess
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String process,
     String userKey,
     String owner)
```

Description

Launches a specified process by creating the new runtime process and beginning its execution. This is a wrapper that combines CreateProcess and StartProcess.

Arguments (input)

wCtx Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.

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 new process and must be passed to all subsequent API calls for that process.

Note: You can pass #SYNC as the itemkey to create a forced synchronous process. See: Forced Synchronous Processes: page 8 – 11.

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.

userkey The user key to assign to the item identified by the specified item type and item key. If userkey is null, then no userkey is assigned to the item instance.

owner A valid role designated as the owner of the item. If owner is null, then no owner is assigned to the process and only the workflow administrator role can monitor the process.

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.

Attention: The Workflow Engine always issues a savepoint before executing each activity in a process so that it can rollback to the previous activity in case an error occurs. For environments such as database triggers or distributed transactions that do not allow savepoints, the Workflow Engine automatically traps "Savepoint not allowed" errors and defers the execution of the activity. If you initiate a workflow process from a database trigger, the Workflow Engine immediately defers the initial start activities to a background engine, so that they are no longer executing from a database trigger.

SuspendProcess

PL/SQL Syntax

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

Java Syntax

```
public static boolean suspendProcess
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String process)
```

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)

wCtx Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.

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 8 – 16.

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, *<parent_process_internal_name>:<label_name>*. If this argument is null, the top level process for the item is suspended. This argument defaults to null.

ResumeProcess

PL/SQL Syntax

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

Java Syntax

```
public static boolean resumeProcess
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String process)
```

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)

wCtx Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.

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 8 – 16.

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, *<parent_process_internal_name>:<label_name>*. If this argument is null, the top level process for the item is resumed. This argument defaults to null.

AbortProcess

PL/SQL Syntax

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

Java Syntax

```
public static boolean abortProcess
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String process,
     String result)
```

Description

Aborts process execution and cancels outstanding notifications. The process status is considered COMPLETE, with a result specified by the `result` argument. Also, any outstanding notifications or subprocesses are set to a status of COMPLETE with a result of force, regardless of the `result` argument.

Arguments (input)

wCtx Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.

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 8 – 16.

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, *<parent_process_internal_name>:<label_name>*. If this argument is null, the top level process for the item is aborted. This argument defaults to null.

result A status assigned to the aborted process. The result must be one of the values defined in the

process Result Type, or one of the following standard engine values:

eng_exception

eng_timeout

eng_force

eng_mail

eng_null

This argument defaults to ″eng_force″.

CreateForkProcess

PL/SQL Syntax

```
procedure CreateForkProcess
    (copy_itemtype in varchar2,
     copy_itemkey in varchar2,
     new_itemkey in varchar2,
     same_version in boolean default TRUE);
```

Description

Forks a runtime process by creating a new process that is a copy of the original. After calling *CreateForkProcess()*, you can call APIs such as *SetItemOwner()*, *SetItemUserKey()*, or the *SetItemAttribute* APIs to reset any item properties or modify any item attributes that you want for the new process. Then you must call *StartForkProcess()* to start the new process.

Use *CreateForkProcess()* when you need to change item specific attributes during the course of a process. For example, if an order cannot be met due to insufficient inventory stock, you can use *CreateForkProcess()* to fork a new transaction for the backorder quantity. Note that any approval notification will be copied. The result is as if two items were created for this transaction.

Arguments (input)

copy_itemtype	A valid item type for the original process to be copied. The new process will have the same item type.
copy_itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The copy item type and key together identify the original process to be copied.
new_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 new item key together identify the new process.
same_version	Specify TRUE or FALSE to indicate whether the new runtime process uses the same version as the original or the latest version. If you specify TRUE, *CreateForkProcess()* copies the item attributes and status of the original process to the new process. If you specify FALSE, *CreateForkProcess()* copies the item attributes of the original process to the new process but does not not copy the status. Defaults to TRUE.

Caution: Do not call *CreateForkProcess()* and *StartForkProcess()* from within a parallel branch in a process. These APIs do not copy any branches parallel to their own branch that are not active.

Note: If an audit trail is required for forked items, your application must record the split and track any required audit information.

StartForkProcess

PL/SQL Syntax

```
procedure StartForkProcess
    (itemtype in varchar2,
     itemkey in varchar2);
```

Description

Begins execution of the new forked process that you specify. Before you call *StartForkProcess*( ), you must first call *CreateForkProcess*( ) to create the new process. You can modify the item attributes of the new process before calling *StartForkProcess()*.

If the new process uses the same version as the original, *StartForkProcess()* copies the status and history of each activity in the forked process, activity by activity. If the new process uses the latest version, then *StartForkProcess()* executes *StartProcess()*.

If you call StartForkProcess() from within a process, any function activity in the process that had a status of 'Active' is updated to have a status of 'Notified.' You must call *CompleteActivity()* afterwards to continue the process.

StartForkProcess() automatically refreshes any notification attributes that are based on item attributes. Any open notifications in the original process are copied and sent again in the new process. Closed notifications are copied but not resent; their status remains remains 'Complete.'

Any Wait activities in the new process are activated at the same time as the original activities. For example, if a 24 hour Wait activity in the original process is due to be eligible in two hours, the new Wait activity is also eligible in two hours.

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.

Caution: Do not call *CreateForkProcess()* and *StartForkProcess()* from within a parallel branch in a process. These APIs do not copy any branches parallel to their own branch that are not active.

Background

PL/SQL 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 are 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 14 – 5.

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 – 35

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 Oracle 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 Oracle database account username and password where you want to run the background engine.

AddItemAttr

PL/SQL Syntax

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

Java Syntax

```
public static boolean addItemAttr
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String aName)
```

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)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
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 8 – 16.
aname	The internal name of the item type attribute.

SetItemAttribute

PL/SQL 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);
```

Java Syntax

```
public static boolean setItemAttrText
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String aName,
     String aValue)

public static boolean setItemAttrNumber
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String aName,
     BigDecimal aValue)

public static boolean setItemAttrDate
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String aName,
     String aValue)
```

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)

wCtx Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.

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 8 – 16.

Note: You can pass #SYNC as the itemkey to create a forced synchronous process. See: Forced Synchronous Processes: page 8 – 11.

aname The internal name of the item type attribute.

avalue The value for the item type attribute.

SetItemAttrDocument

PL/SQL Syntax

```
procedure SetItemAttrDocument
    (itemtype in varchar2,
     itemkey in varchar2,
     aname in varchar2,
     documentid in varchar2);
```

Java Syntax

```
public static boolean setItemAttrDocument
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String aName,
     String documentId)
```

Description

Sets the value of an item attribute of type document, to a document identifier.

Arguments (input)

wCtx Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.

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 8 – 16.

Note: You can pass #SYNC as the itemkey to create a forced synchronous process. See: Forced Synchronous Processes: page 8 – 11.

aname The internal name of the item type attribute.

documentid The value for the item type attribute as a fully concatenated string of the following values:

DM:<*node_id*>:<*doc_id*>:<*version*>

<*node_id*> is the node ID assigned to the document management system node as defined in the Document Management Nodes web page. See: To Define a Document Node: page 2 – 31.

<doc_id> is the document ID of the document, as assigned by the document management system where the document resides.

<version> is the version of the document. If a version is not specified, the latest version is assumed.

getItemTypes

Java Syntax

```
public static WFTwoDDataSource getItemTypes
    (WFContext wCtx)
```

Description Returns a list of all the item types defined in the Oracle Workflow database as a two dimensional data object.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.

GetItemAttribute

PL/SQL 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: CreateProcess: page 8 – 16.

Note: Pass #SYNC as the itemkey to create a forced synchronous process. See: Forced Synchronous Processes: page 8 – 11.

aname The internal name of an item type attribute.

GetItemAttrDocument

PL/SQL Syntax

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

Description

Returns the document identifier for a document–type item attribute. The document identifier is a concatenated string of the following values:

DM:<*nodeid*>:<*documentid*>:<*version*>

<*nodeid*> is the node ID assigned to the document management system node as defined in the Document Management Nodes web page. See: To Define a Document Node: page 2 – 31.

<*documentid*> is the document ID of the document, as assigned by the document management system where the document resides.

<*version*> is the version of the document. If a version is not specified, the latest version is assumed.

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 8 – 16.

Note: Pass #SYNC as the itemkey to create a forced synchronous process. See: Forced Synchronous Processes: page 8 – 11.

aname The internal name of the item type attribute.

getItemAttributes

Java Syntax

```
public static WFTwoDDataSource getItemAttributes
    (WFContext wCtx,
     String itemType,
     String itemKey)
```

Description

Returns a list of all the item attribute and their values for the specified item type instance as a two dimensional data object.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
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 8 – 16.

GetItemAttrInfo

PL/SQL 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. Currently, subtype information is not available for item type attributes

Arguments (input)

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

GetActivityAttrInfo

PL/SQL 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 8 – 16.

actid The activity ID for a particular usage of an activity in a process definition. Also referred to as the activity ID of the node

aname The internal name of an activity attribute.

GetActivityAttribute

PL/SQL 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: CreateProcess: page 8 – 16.

Note: Pass #SYNC as the itemkey to create a forced synchronous process. See: Forced Synchronous Processes: page 8 – 11.

actid The activity ID for a particular usage of an activity in a process definition. Also referred to as the activity ID of the node

aname The internal name of an activity attribute.

BeginActivity

PL/SQL 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 8 – 49.

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 activity node to perform on the process. Provide the activity node's label name. If the activity node label name does not uniquely identify the activity node you can precede the label name with the internal name of its parent process. For example, *<parent_process_internal_name>:<label_name>.*

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);
end if;
```

CompleteActivity

PL/SQL 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 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 'Notified'. An optional activity completion result can also be passed. The result can determine what transition the process takes next.
- **To start and 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.

  Note: You can call CompleteActivity instead of StartActivity when you want to start a process with an activity node that is mid–stream in a process thread and not at the beginning of a process thread. The activity node you specify as the beginning of the process must be set to Start in the Node tab of its property page or else an error will be raised.

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 activity node that is completed. Provide the activity node's label name. If the activity node label name does not uniquely identify the subprocess you can precede the label name with the internal name of its parent process. For example, *<parent_process_internal_name>:<label_name>*.

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 8 – 28.

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

PL/SQL 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 by calling *WF_Notification.Transfer*.

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 activity node. If the activity node label name does not uniquely identify the activity node 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

PL/SQL Syntax

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

Description

This procedure is generally 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 activity node label is not unique within the process you may precede the activity node label name with the internal name of its parent process. For example, *<parent_process_internal_name>:<label_name>*.

If the On_Revisit flag is set to Reset, this procedure clears the activity specified and all activities following it that have already been transitioned to by reexecuting each activity in 'Cancel' mode. See: Looping: page 8 – 7. 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 activity node that encountered the error or that you want to undo. Provide the label name of the activity node. If the activity node label name does not uniquely identify the subprocess you can precede the label name with the internal name of

its parent process. For example, *<parent_process_internal_name>:<label_name>.*

command One of two commands that determine how to handle the process activity:

SKIP—do not reexecute the activity, but mark the activity as complete with the supplied result and continue execution of the process from that activity.

RETRY—reexecute the activity and continue execution of the process from that activity.

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_ENGINEAbortProcess*( ).
- 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

PL/SQL 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.

ItemStatus

PL/SQL Syntax

```
procedure ItemStatus
    (itemtype in varchar2,
     itemkey in varchar2,
     status out varchar2,
     result out varchar2);
```

Java Syntax

```
public static WFTwoDDataSource itemStatus
    (WFContext wCtx,
     String itemType,
     String itemKey)
```

Description

Returns the status and result for the root process of the specified item instance. Possible values returned for the status are: ACTIVE, COMPLETE, ERROR, or SUSPENDED. If the root process does not exist, then the item key does not exist and will thus cause the procedure to raise an exception.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
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 instance.

getProcessStatus

Java Syntax

```
public static WFTwoDDataSource getProcessStatus
    (WFContext wCtx,
     String itemType,
     String itemKey,
     BigDecimal process)
```

Description

Returns the process status for the given item type instance as a two dimensional data object.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
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 8 – 16.
process	A process instance ID for the item type. If the instance ID is unknown, you can simply provide any negative number and the method will return the process status for the root process.

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 6 – 20.

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.

- CLEAR: page 8 – 59
- GET_ERROR: page 8 – 60
- TOKEN: page 8 – 61
- RAISE: page 8 – 62
- CONTEXT: page 8 – 65
- TRANSLATE: page 8 – 67

See Also

Standard API for an Oracle Workflow PL/SQL Stored Procedure: page 7 – 2

CLEAR

Syntax `procedure CLEAR;`

Description Clears the error buffers.

See Also

GET_ERROR: page 8 – 60

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 8 – 59

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 8 – 62

CONTEXT: page 8 – 65

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 8 – 61

CONTEXT: page 8 – 65

▶ **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] -u <username/password@database>
  <lang> <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>
   ```

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), funmode);
    raise;
end MyFunction;
```

See Also

TOKEN: page 8 – 61

RAISE: page 8 – 62

TRANSLATE

Syntax

```
function TRANSLATE
    (tkn_name IN VARCHAR2)
     return VARCHAR2;
```

Description Translates the string value of a token by returning the value for the token as defined in WF_RESOURCES for your language setting.

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 for completed items and processes, and to purge information for obsolete activity versions that are no longer in use. You may want to periodically purge this obsolete data from your system to increase performance.

A PL/SQL variable called "persistence_type"in the WF_PURGE package defines how all the WF_PURGE APIs behave, with the exception of TotalPerm( ). When the variable is set to TEMP, the WF_Purge APIs only purge data associated with Temporary item types, whose persistence, in days, has expired. Persistence_type is set to TEMP by default and should not be changed. If you need to purge runtime data for item types with Permanent persistence, you should use the procedure TotalPerm( ). See: Persistence Type: page 4 – 4.

Attention: You cannot run any WF_PURGE API for a future end date. By entering a future end date, you may inadvertently violate the persistence period for Temporary item types. The WF_PURGE APIs will display an error message if you enter a future end date.

The three 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.

The complete list of purge APIs are as follows:

- Items: page 8 – 70
- Activities: page 8 – 71
- Notifications: page 8 – 72
- Total: page 8 – 73
- TotalPERM: page 8 – 74

- AdHocDirectory: page 8 – 75

  Note: If you are using the version of Oracle Workflow embedded in Oracle Applications, you may also use the "Purge Obsolete Workflow Runtime Data" concurrent program to purge obsolete item type runtime status information. See: Purge Obsolete Workflow Runtime Data: page 8 – 76.

See Also

Standard API for an Oracle Workflow PL/SQL Stored Procedure: page 7 – 2

Items

Syntax

```
procedure Items
    (itemtype in varchar2 default null,
     itemkey in varchar2 default null,
      enddate in date default sysdate);
```

Description Deletes all items for the specified item type, and/or item key, and end date, including process status and notification information. Deletes from the tables WF_NOTIFICATIONS, WF_ITEM_ACTIVITY_STATUSES, WF_ITEM_ATTRIBUTE_VALUES and WF_ITEMS.

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.

Activities

Syntax

```
procedure Activities
    (itemtype in varchar2 default null,
     name in varchar2 default null,
     enddate in date default sysdate);
```

Description

Deletes old versions of eligible activities from the tables WF_ACTIVITY_ATTR_VALUES, WF_ACTIVITY_TRANSITIONS, WF_PROCESS_ACTIVITIES, WF_ACTIVITY_ATTRIBUTES_TL, WF_ACTIVITY_ATTRIBUTES, WF_ACTIVITIES_TL, and WF_ACTIVITIES 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 as either a process or activity.

> **Note:** You should call *Items*( ) before calling *Activities*( ) to avoid having obsolete item references prevent obsolete activities from being deleted.

Arguments (input)

itemtype	Item type associated with the activities you want to delete. Leave this argument null to delete activities for all item types.
name	Internal name of activity to delete. Leave this argument null to delete all activities for the specified item type.
enddate	Specified date to delete up to.

Notifications

Syntax

```
procedure Notifications
    (itemtype in varchar2 default null,
     enddate in date default sysdate);
```

Description

Deletes old eligible 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.

Total

Syntax

```
procedure Total
    (itemtype in varchar2 default null,
     itemkey in varchar2,
      enddate in date default sysdate);
```

Description

Deletes all eligible 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.

TotalPERM

Syntax

```
procedure TotalPERM
    (itemtype in varchar2 default null,
     itemkey in varchar2 default null,
      enddate in date default sysdate);
```

Description

Deletes all eligible obsolete runtime data that is of persistence type 'PERM' (Permanent) and that is associated with the specified item type and has 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.

AdHocDirectory

Syntax

```
procedure AdHocDirectory
    (end_date in date default sysdate);
```

Description Purges all users and roles in the WF_LOCAL_* tables whose expiration date is less than or equal to the specified end_date and that are not referenced in any notification.

Arguments (input)

end_date	Date to purge to.

Purge Obsolete Workflow Runtime Data Concurrent Program

▶ **To Purge Obsolete Workflow Runtime Data**

1. Navigate to the Submit Requests form in Oracle Applications to submit the Purge Obsolete Workflow Runtime Data 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. The executable name for this concurrent program is "Oracle Workflow Purge Obsolete Data" and its short name is FNDWFPR. See: Overview of Concurrent Programs and Requests, *Oracle Applications System Administrator's Guide*
2. Submit the Purge Obsolete Workflow Runtime Data 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:

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 program purges all items in the specified itemtype.
age	Minimum age of data to purge, in days, if x_persistence_type is set to 'TEMP'. The default is 0.
x_persistence_type	Persistence type to be purged, either 'TEMP' for Temporary or 'PERM' for Permanent. The default is 'TEMP'.

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.

Workflow Directory Service APIs

The following APIs can be called by an application program or a workflow function in the runtime phase to retrieve information about existing users and roles, as well as create and manage new ad hoc users and roles in the directory service. These APIs are defined in a PL/SQL package called WF_DIRECTORY.

- GetRoleUsers: page 8 – 78
- GetUserRoles: page 8 – 79
- GetRoleInfo: page 8 – 80
- GetRoleInfo2: page 8 – 81
- IsPerformer: page 8 – 82
- CurrentUser: page 8 – 83
- UserActive: page 8 – 84
- GetUserName: page 8 – 85
- GetRoleName: page 8 – 86
- GetRoleDisplayName: page 8 – 87
- SetAdHocUserStatus: page 8 – 88
- SetAdHocRoleStatus: page 8 – 89
- CreateAdHocUser: page 8 – 90
- CreateAdHocRole: page 8 – 92
- AddUsersToAdHocRole: page 8 – 94
- SetAdHocUserExpiration: page 8 – 95
- SetAdHocRoleExpiration: page 8 – 96
- SetAdHocUserAttr: page 8 – 97
- SetAdHocRoleAttr: page 8 – 98
- RemoveUsersFromAdHocRole: page 8 – 99

See Also

Standard API for an Oracle Workflow PL/SQL Stored Procedure: page 7 – 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', 'MAILATTH', 'SUMMARY')
- Language
- Territory

Arguments (input)

role	A valid role name.

GetRoleInfo2

Syntax

```
procedure GetRoleInfo2
    (Role in varchar2,
    Role_Info_Tbl out wf_directory.wf_local_roles_tbl_type);
```

Description

Returns the following information about a role in a SQL table:

- Display name
- Description
- Notification Preference ('QUERY', 'MAILTEXT', 'MAILHTML', 'SUMMARY')
- Language
- Territory
- Email address
- FAX
- Status
- Expiration Date

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.

GetRoleDisplayName

Syntax

```
function GetRoleDisplayName
      (p_role_name in varchar2)
       return varchar2;
       pragma restrict_references(GetRoleDisplayName, WNDS,
       WNPS);
```

Description Returns a Workflow role's display name given the role's internal name.

Arguments (input)

p_role_name The internal name of the role.

SetAdHocUserStatus

Syntax

```
procedure SetAdHocUserStatus
     (user_name in varchar2,
      status in varchar2 default 'ACTIVE');
```

Description Sets the status of an ad hoc user as 'ACTIVE' or 'INACTIVE'.

Arguments (input)

user_name	The internal name of the user.
status	A status of 'ACTIVE' or 'INACTIVE' to set for the user. If null, the status is 'ACTIVE'.

SetAdHocRoleStatus

Syntax

```
procedure SetAdHocRoleStatus
     (role_name in varchar2,
      status in varchar2 default 'ACTIVE');
```

Description Sets the status of an ad hoc role as 'ACTIVE' or 'INACTIVE'.

Arguments (input)

role_name	The internal name of the role.
status	A status of 'ACTIVE' or 'INACTIVE' to set for the role. If null, the status is 'ACTIVE'.

CreateAdHocUser

Syntax

```
procedure CreateAdHocUser
      (name in out varchar2,
       display_name in out varchar2,
       language in varchar2 default null,
       territory in varchar2 default null,
       description in varchar2 default null,
       notification_preference in varchar2 default
       'MAILHTML',
       email_address in varchar2 default null,
       fax in varchar2 default null,
       status in varchar2 default 'ACTIVE',
       expiration_date in date default sysdate);
```

Description

Creates a user at runtime by creating a value in the WF_LOCAL_USERS table. This is referred to as an ad hoc user.

Arguments (input)

name

An internal name for the user. The name must less than 30 characters long and is all uppercase. This procedure checks that the name provided does not already exist in WF_USERS and returns an error if the name already exists. If you do not provide an internal name, the system generates an internal name for you where the name contains a prefix of '~WF_ADHOC–' followed by a sequence number.

display_name

The display name of the user. This procedure checks that the display name provided does not already exist in WF_USERS and returns an error if the display name already exists. If you do not provide an display name, the system generates one for you where the display name contains a prefix of '~WF_ADHOC–' followed by a sequence number.

language

The value of the Oracle8 NLS_LANGUAGE initialization parameter that specifies the default language–dependent behavior of the user's notification session. If null, the procedure resolves this to the language setting of your current session.

territory

The value of the Oracle8 NLS_TERRITORY initialization parameter that specifies the default territory–dependant date and numeric formatting used in the user's notification session. If null, the

	procedure resolves this to the territory setting of your current session.
description	An optional description for the user.
notification_ preference	Indicate how this user prefers to receive notifications: 'MAILTEXT', 'MAILHTML', 'MAILATTH', 'QUERY' or 'SUMMARY'. If null, the procedure sets the notification preference to 'MAILHTML'.
email_address	A optional electronic mail address for this user.
fax	An optional Fax number for the user.
status	The availability of the user to participate in a workflow process. The possible statuses are 'ACTIVE', 'EXTLEAVE', 'INACTIVE', and 'TMPLEAVE'. If null, the procedure sets the status to 'ACTIVE'.
expiration_date	The date at which the user is no longer valid in the directory service. If null, then no expiration date is set.

See Also

Setting Up an Oracle Workflow Directory Service: page 2 – 17

CreateAdHocRole

Syntax

```
procedure CreateAdHocRole
     (role_name in out varchar2,
      role_display_name in out varchar2,
      language in varchar2 default null,
      territory in varchar2 default null,
      role_description in varchar2 default null,
      notification_preference in varchar2 default
      'MAILHTML',
      role_users in varchar2 default null,
      email_address in varchar2 default null,
      fax in varchar2 default null,
      status in varchar2 default 'ACTIVE',
      expiration_date in date default sysdate);
```

Description

Creates a role at runtime by creating a value in the WF_LOCAL_ROLES table. This is referred to as an ad hoc role.

Arguments (input)

role_name
An internal name for the role. The name must less than 30 characters long and is all uppercase. This procedure checks that the name provided does not already exist in WF_ROLES and returns an error if the name already exists. If you do not provide an internal name, the system generates an internal name for you where the name contains a prefix of '~WF_ADHOC–' followed by a sequence number.

role_display_name
The display name of the role. This procedure checks that the display name provided does not already exist in WF_ROLES and returns an error if the display name already exists. If you do not provide an display name, the system generates one for you where the display name contains a prefix of '~WF_ADHOC–' followed by a sequence number.

language
The value of the Oracle8 NLS_LANGUAGE initialization parameter that specifies the default language–dependent behavior of the user's notification session. If null, the procedure resolves this to the language setting of your current session.

territory
The value of the Oracle8 NLS_TERRITORY initialization parameter that specifies the default territory–dependant date and numeric formatting used in the user's notification session. If null, the

	procedure resolves this to the territory setting of your current session.
role_description	An optional description for the role.
notification_ preference	Indicate how this role receives notifications: 'MAILTEXT', 'MAILHTML', 'MAILATTH', 'QUERY' or 'SUMMARY'. If null, the procedure sets the notification preference to 'MAILHTML'.
role_users	Indicate the names of the users that belong to this role, using commas or spaces to delimit the list.
email_address	A optional electronic mail address for this role or a mail distribution list defined by your electronic mail system.
fax	An optional Fax number for the role.
status	The availability of the role to participate in a workflow process. The possible statuses are ACTIVE, EXTLEAVE, INACTIVE, and TMPLEAVE. If null, the procedure sets the status to 'ACTIVE'.
expiration_date	The date at which the role is no longer valid in the directory service. If null, then no expiration date is set.

See Also

Setting Up an Oracle Workflow Directory Service: page 2 – 17

AddUsersToAdHocRole

Syntax

```
procedure AddUsersToAdHocRole
     (role_name in varchar2,
      role_users in varchar2);
```

Description Adds users to a existing ad hoc role.

Arguments (input)

role_name	The internal name of the ad hoc role.
role_users	List of users delimited by spaces or commas. Users can be ad hoc users or users defined in an application, but they must already be defined in the Oracle Workflow directory service.

SetAdHocUserExpiration

Syntax

```
procedure SetAdHocUserExpiration
     (user_name in varchar2,
      expiration_date in date default sysdate);
```

Description Updates the expiration date for an ad hoc user.

Arguments (input)

user_name	The internal name of the ad hoc user.
expiration_date	New expiration date. If null, the procedure defaults the expiration date to sysdate.

SetAdHocRoleExpiration

Syntax

```
procedure SetAdHocRoleExpiration
     (role_name in varchar2,
      expiration_date in date default sysdate);
```

Description Updates the expiration date for an ad hoc role.

Arguments (input)

role_name	The internal name of the ad hoc role.
expiration_date	New expiration date. If null, the procedure defaults the expiration date to sysdate.

SetAdHocUserAttr

Syntax

```
procedure SetAdHocUserAttr
     (user_name in varchar2,
      display_name in varchar2 default null,
      notification_preference in varchar2 default null,
      language in varchar2 default null,
      territory in varchar2 default null,
      email_address in varchar2 default null,
      fax in varchar2 default null);
```

Description Updates the attributes for an ad hoc user.

Arguments (input)

user_name	The internal name of the ad hoc user to update.
display_name	A new display name for the ad hoc user. If null, the display name is not updated.
notification_ preference	A new notification preference of 'MAILTEXT', 'MAILHTML', 'MAILATTH', 'QUERY' or 'SUMMARY'. If null, the notification preference is not updated.
language	A new value of the Oracle8 NLS_LANGUAGE initialization parameter for the ad hoc user. If null, the language setting is not updated.
territory	A new value of the Oracle8 NLS_TERRITORY initialization parameter for the ad hoc user. If null, the territory setting is not updated.
email_address	A new valid electronic mail address for the ad hoc user. If null, the electronic mail address is not updated.
fax	A new fax number for the ad hoc user. If null, the fax number is not updated.

SetAdHocRoleAttr

Syntax

```
procedure SetAdHocRoleAttr
     (role_name in varchar2,
      display_name in varchar2 default null,
      notification_preference in varchar2 default null,
      language in varchar2 default null,
      territory in varchar2 default null,
      email_address in varchar2 default null,
      fax in varchar2 default null);
```

Description Updates the attributes for an ad hoc role.

Arguments (input)

role_name	The internal name of the ad hoc role to update.
display_name	A new display name for the ad hoc role. If null, the display name is not updated.
notification_ preference	A new notification preference of 'MAILTEXT', 'MAILHTML', 'QUERY' or 'SUMMARY'. If null, the notification preference is not updated.
language	A new value of the Oracle8 NLS_LANGUAGE initialization parameter for the ad hoc role. If null, the language setting is not updated.
territory	A new value of the Oracle8 NLS_TERRITORY initialization parameter for the ad hoc role. If null, the territory setting is not updated.
email_address	A new valid electronic mail address for the ad hoc role. If null, the electronic mail address is not updated.
fax	A new fax number for the ad hoc role. If null, the fax number is not updated.

RemoveUsersFromAdHocRole

Syntax

```
procedure RemoveUsersFromAdHocRole
     (role_name in varchar2,
      role_users in varchar2 default null);
```

Description Removes users from an existing ad hoc role.

Arguments (input)

role_name	The internal name of the ad hoc role.
role_users	List of users to remove from the ad hoc role. The users are delimited by commas or spaces. If null, all users are removed from the role.

Workflow Preferences API

Call the following API to retrieve user preference information. The API is defined in the PL/SQL package called WF_PREF.

get_pref

Syntax

```
function get_pref
    (p_user_name in varchar2,
     p_preference_name in varchar2)
     return varchar2;
```

Description Retrieves the value of the specified preference for the specified user.

Arguments (input)

p_user_name	The internal name of the user. To retrieve the value for a global preference, specify the user as –WF_DEFAULT–.
p_preference_name	The name of the user preference whose value you wish to retrieve. Valid preference names are: LANGUAGE TERRITORY MAILTYPE DMHOME DATEFORMAT

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.

- GetDiagramURL: page 8 – 102
- GetEnvelopeURL: page 8 – 104
- GetAdvancedEnvelopeURL: page 8 – 106

Attention: The GetURL API from earlier versions of Oracle Workflow is 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 web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 12.

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

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:

http://*<server.com:portID>*/*<PLSQL_agent_path>*

When calling this function, your application must first retrieve the web agent string from the

WF_RESOURCES token WF_WEB_AGENT by calling *WF_CORE.TRANSLATE( )*. See: Setting Global User Preferences: page 2 – 12.

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');
```

See Also

TRANSLATE: page 8 – 67

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 web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 12.

`wf_monitor.envelope` represents the name of the PL/SQL package procedure that generates the Workflow Monitor Notifications List for the process instance.

Arguments (input)

x_agent 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:

http://*<server.com:portID>*/*<PLSQL_agent_path>*

When calling this function, your application must first retrieve the web agent string from the WF_RESOURCES token WF_WEB_AGENT by calling *WF_CORE.TRANSLATE( )*. See: Setting Global User Preferences: page 2 – 12.

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 8 – 67

GetAdvancedEnvelopeURL

Syntax

```
function GetAdvancedEnvelopeURL
    (x_agent in varchar2,
     x_item_type in varchar2,
     x_item_key in varchar2,
     x_admin_mode in varchar2 default 'NO',
     x_options in varchar2 default null)
     return varchar2;
```

Description

Can be called by an application to return a URL that displays the Workflow Monitor Activities List with an attached access key password. The URL displays the Activities List for a specific instance of a workflow process in the Workflow Monitor. The Activities List allows you to apply advanced filtering options in displaying the list of activities for a process instance.

The URL returned by the function *WF_MONITOR.GetAdvancedEnvelopeURL( )* looks as follows if the x_options argument is null:

Example

<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>*&*x_advanced=TRUE*

<webagent> represents the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 12.

wf_monitor.envelope represents the name of the PL/SQL package procedure that generates the Workflow Monitor Notifications List for the process instance.

Arguments (input)

x_agent

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:

http://*<server.com:portID>*/*<PLSQL_agent_path>*

When calling this function, your application must first retrieve the web agent string from the WF_RESOURCES token WF_WEB_AGENT by calling *WF_CORE.TRANSLATE( ).* See: Setting Global User Preferences: page 2 – 12.

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.
x_options	Specify 'All' if you wish to return a URL that displays the Activities List with all filtering options checked. If you leave this argument null, then a URL that displays the Activities List with no filtering options checked, is returned. This allows you to append any specific options if you wish. The default is null.

See Also

TRANSLATE: page 8 – 67

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: page 8 – 108
- WF_NOTIFICATION_ATTR_RESP_V: page 8 – 110
- WF_RUNNABLE_PROCESSES_V: page 8 – 111
- WF_ITEMS_V: page 8 – 112

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(4000)
ASSIGNED_USER	VARCHAR2(30)
ASSIGNED_USER_DISPLAY_NAME	VARCHAR2(4000)
NOTIFICATION_ID	NUMBER
OUTBOUND_QUEUE_ID	RAW(16)
ERROR_NAME	VARCHAR2(30)
ERROR_MESSAGE	VARCHAR2(2000)
ERROR_STACK	VARCHAR2(4000)

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 4 – 50.

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(4000)
ATTRIBUTE_NAME	NOT NULL	VARCHAR2(30)
ATTRIBUTE_DISPLAY_NAME	NOT NULL	VARCHAR2(80)
ATTRIBUTE_VALUE		VARCHAR2(2000)
ATTRIBUTE_DISPLAY_VALUE		VARCHAR2(4000)
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)
USER_KEY                                 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
```

Workflow Queue APIs

Oracle Workflow queue APIs can be called by an application program or a workflow function in the runtime phase to handle workflow Advanced Queues processing. In Oracle Workflow, an 'outbound' and an 'inbound' queue is established. A package of data on the queue is referred to as an event or a message. A message in this context is different from the messages associated with notification activities. Events are enqueued in the outbound queue for agents to consume and process. These agents may be any application that is external to the database. Similarly an agent may enqueue some message to the inbound queue for the Workflow Engine to consume and process. The outbound and inbound queues facilitate the integration of external activities into your workflow processes.

Note: Background engines use a separate 'deferred' queue.

All Oracle Workflow queue APIs are defined in a PL/SQL package called WF_QUEUE. You must execute these queue APIs from the same Oracle Workflow account since the APIs are account dependent.

Attention: In using these APIs, we assume that you have prior knowledge of Oracle8 Advanced Queues concepts and terminology. Refer to the *Oracle8 Server Application Developer's Guide* for more information on Advanced Queues.

Queue APIs

- EnqueueInbound: page 8 – 116
- DequeueOutbound: page 8 – 117
- DequeueEventDetail: page 8 – 120
- PurgeEvent: page 8 – 122
- PurgeItemtype: page 8 – 123
- ProcessInboundQueue: page 8 – 124
- GetMessageHandle: page 8 – 125
- Deferred_queue: page 8 – 126
- Inbound_queue: page 8 – 127
- Outbound_queue: page 8 – 128

Developer APIs for the Inbound Queue

The following APIs are for developers who wish to write to the inbound queue by creating messages in the internal stack rather than

using WF_QUEUE.EnqueueInbound( ). The internal stack is purely a storage area and you must eventually write each message that you create on the stack to the inbound queue.

> **Note:** For efficient performance, you should periodically write to the inbound queue to prevent the stack from growing too large.

- ClearMsgStack: page 8 – 129
- CreateMsg: page 8 – 130
- WriteMsg: page 8 – 131
- SetMsgAttr: page 8 – 132
- SetMsgResult: page 8 – 133

Payload Structure

All Oracle Workflow queues use the data type system.wf_payload_t to define the payload for any given message. The payload contains all the information that is required about the event. A description of system.wf_payload_t is as follows:

System.wf_payload_t

Column Name	Type	Description
ITEMTYPE	Varchar2(8)	The item type of the event.
ITEMKEY	Varchar2(240)	The item key of the event.
ACTID	Number	The function activity instance ID.
FUNCTION_NAME	Varchar2(200)	The name of the function to execute.
PARAM_LIST	Varchar2(4000)	A list of "value_name=value" pairs. In the inbound scenario, the pairs are passed as item attributes and item attribute values. In the outbound scenario, the pairs are passed as all the attributes and attribute values of the function (activity attributes).
RESULT	Varchar2(30)	An optional activity completion result. Possible values are determined by the function activity's Result Type or can be an engine standard result.

Table 8 – 1 (Page 1 of 1)

See Also

Standard API for an Oracle Workflow PL/SQL Stored Procedure: page 7 – 2

Advanced Queuing: *Oracle8 Server Application Developer's Guide*

EnqueueInbound

Syntax

```
procedure EnqueueInbound
    (itemtype in varchar2,
     itemkey in varchar2,
     actid in number,
     result in varchar2 default null,
     attrlist in varchar2 default null,
     correlation in varchar2 default null);
```

Description

Enqueues the result from an outbound event onto the inbound queue. An outbound event is defined by an outbound queue message that is consumed by some agent. Oracle Workflow marks the outbound event as complete with the given result when it processes the inbound queue.

Arguments (input)

itemtype The item type of the event.

itemkey The item key of the event. An item key is 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 instance.

actid The function activity instance ID that this event is associated with.

result An optional activity completion result. Possible values are determined by the function activity's Result Type.

attrlist A longlist of "value name=value" pairs that you want to pass back as item attributes and item attribute values. Each pair must be delimited by the caret character (^), as in the example, "ATTR1=A^ATTR2=B^ATTR3=C". If a specified value name does not exist as an item attribute, Oracle Workflow creates the item attribute for you, of type varchar2.

correlation Specify an optional correlation identifier for the message to be enqueued. Oracle8 Advanced Queues allow you to search a queue for messages based on a specific correlation value. You can use the Like comparison operator, '%', to specify the identifier string. If null, the Workflow Engine creates a correlation identifier.

DequeueOutbound

Syntax

```
procedure DequeueOutbound
    (dequeuemode in number,
     navigation in number default 1,
     correlation in varchar2 default null,
     itemtype in varchar2 default null,
     payload out system.wf_payload_t,
     message_handle in out raw,
     timeout out boolean);
```

Description

Dequeues a message from the outbound queue for some agent to consume.

Attention: If you call this procedure within a loop, you must remember to set the returned message handle to null, otherwise, the procedure dequeues the same message again. This may not be the behavior you want and may cause an infinite loop.

Arguments (input)

dequeuemode A value of DBMS_AQ.BROWSE, DBMS_AQ.LOCKED, or DBMS_AQ.REMOVE, corresponding to the numbers 1, 2 and 3 respectively, to represent the locking behavior of the dequeue. A mode of DBMS_AQ.BROWSE means to read the message from the queue without acquiring a lock on the message. A mode of DBMS_AQ.LOCKED means to read and obtain a write lock on the message, where the lock lasts for the duration of the transaction. A mode of DBMS_AQ.REMOVE means read the message and delete it.

navigation Specify DBMS_AQ.FIRST_MESSAGE or DBMS_AQ.NEXT_MESSAGE, corresponding to the number 1 or 2 respectively, to indicate the position of the message that will be retrieved. A value of DBMS_AQ.FIRST_MESSAGE retrieves the first message that is available and matches the correlation criteria. The first message is inherently the beginning of the queue. A value of DBMS_AQ.NEXT_MESSAGE retrieves the next message that is available and matches the correlation criteria, and lets you read through the queue. The default is 1.

correlation Specify an optional correlation identifier for the message to be dequeued. Oracle8 Advanced Queues allow you to search a queue for messages based on a specific correlation value. You can use the Like comparison operator, '%', to specify the identifier string. If null, the Workflow Engine creates a correlation identifier based on the item type.

itemtype The item type of the event.

message_handle Specify an optional message handle ID for the specific event to be dequeued. If you specify a message handle ID, the correlation identifier is ignored.

Attention: The timeout output returns TRUE when there is nothing further to read in the queue.

Example Following is an example of code that loops through the outbound queue and displays the output.

```
declare

   event           system.wf_payload_t;
   i               number;
   msg_id          raw(16);
   queuename       varchar2(30);
   navigation_mode number;
   end_of_queue    boolean;

begin
   queuename:=wf_queue.OUTBOUNDQUEUE;
   i:=0;
   LOOP
      i:=i+1;

      -- always start with the first message then progress
to next
      if i = 1 then
         navigation_mode   := dbms_aq.FIRST_MESSAGE;
      else
         navigation_mode   := dbms_aq.NEXT_MESSAGE;
      end if;

      -- not interested in specific msg_id. Leave it null so
```

```
      --as to loop through all messages in queue
      msg_id :=null;

      wf_queue.DequeueOutbound(
                        dequeuemode     => dbms_aq.BROWSE,
                        payload         => event,
                        navigation      => navigation_mode,
                        message_handle  => msg_id,
                        timeout         => end_of_queue);

      if end_of_queue then
         exit;
      end if;

      -- print the correlation itemtype:itemKey
      dbms_output.put_line('Msg '||to_char(i)||' = '||
                          event.itemtype||':'||event.itemkey
                          ||' '||event.actid||' '
                          ||event.param_list);
   END LOOP;

end;
/
```

DequeueEventDetail

Syntax

```
procedure DequeueEventDetail
    (dequeuemode in number,
     navigation in number default 1,
     correlation in varchar2 default null,
     itemtype in out varchar2,
     itemkey out varchar2,
     actid out number,
     function_name out varchar2,
     param_list out varchar2,
     message_handle in out raw,
     timeout out boolean);
```

Description

Dequeue from the outbound queue, the full event details for a given message. This API is similar to DequeueOutbound except it does not reference the payload type. Instead, it outputs itemkey, actid, function_name, and param_list, which are part of the payload.

Attention: If you call this procedure within a loop, you must remember to set the returned message handle to null, otherwise, the procedure dequeues the same message again. This may not be the behavior you want and may cause an infinite loop.

Arguments (input)

dequeuemode A value of DBMS_AQ.BROWSE, DBMS_AQ.LOCKED, or DBMS_AQ.REMOVE, corresponding to the numbers 1, 2 and 3 respectively, to represent the locking behavior of the dequeue. A mode of DBMS_AQ.BROWSE means to read the message from the queue without acquiring a lock on the message. A mode of DBMS_AQ.LOCKED means to read and obtain a write lock on the message, where the lock lasts for the duration of the transaction. A mode of DBMS_AQ.REMOVE means read the message and update or delete it.

navigation Specify DBMS_AQ.FIRSTMESSAGE or DBMS_AQ.NEXTMESSAGE, corresponding to the number 1 or 2 respectively, to indicate the position of the message that will be retrieved. A value of DBMS_AQ.FIRSTMESSAGE retrieves the first message that is available and matches the correlation criteria. It also resets the position to the beginning of the queue. A value of

DBMS_AQ.NEXTMESSAGE retrieves the next message that is available and matches the correlation criteria. The default is 1.

correlation Specify an optional correlation identifier for the message to be dequeued. Oracle8 Advanced Queues allow you to search a queue for messages based on a specific correlation value. You can use the Like comparison operator, '%', to specify the identifier string. If null, the Workflow Engine creates a correlation identifier based on the item type.

acctname The Oracle Workflow database account name. If acctname is null, it defaults to the pseudocolumn USER.

itemtype Specify an optional item type for the message to dequeue if you are not specifying a correlation.

message_handle Specify an optional message handle ID for the specific event to be dequeued. If you specify a message handle ID, the correlation identifier is ignored.

Attention: The timeout output returns TRUE when there is nothing further to read in the queue.

PurgeEvent

Syntax

```
procedure PurgeEvent
    (queuename in varchar2,
     message_handle in raw);
```

Description Removes an event from a specified queue without further processing.

Arguments (input)

queuename	The name of the queue from which to purge the event.
message_handle	The message handle ID for the specific event to purge.

PurgeItemType

Syntax

```
procedure PurgeItemType
    (queuename in varchar2,
     itemtype in varchar2 default null,
     correlation in varchar2 default null);
```

Description

Removes all events belonging to a specific item type from a specified queue without further processing.

Arguments (input)

queuename The name of the queue from which to purge the events.

itemtype An optional item type of the events to purge.

correlation Specify an optional correlation identifier for the message to be purged. Oracle8 Advanced Queues allow you to search a queue for messages based on a specific correlation value. You can use the Like comparison operator, '%', to specify the identifier string. If null, the Workflow Engine creates a correlation identifier based on the item type.

ProcessInboundQueue

Syntax

```
procedure ProcessInboundQueue
     (itemtype in varchar2 default null,
      correlation in varchar2 default null);
```

Description

Reads every message off the inbound queue and records each message as a completed event. The result of the completed event and the list of item attributes that are updated as a consequence of the completed event are specified by each message in the inbound queue. See: EnqueueInbound: page 8 – 116.

Arguments (input)

itemtype	An optional item type of the events to process.
correlation	If you wish to process only messages with a specific correlation, enter a correlation identifier. If correlation is null, the Workflow Engine creates a correlation identifier based on the item type.

GetMessageHandle

Syntax

```
function GetMessageHandle
    (queuename in varchar2,
     itemtype in varchar2,
     itemkey in varchar2,
     actid in number,
     correlation in varchar2 default null,
     return raw;
```

Description Returns a message handle ID for a specified message.

Arguments (input)

queuename	The name of the queue from which to retrieve the message handle.
itemtype	The item type of the message.
itemkey	The item key of the message. An item key is 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 instance.
actid	The function activity instance ID that this message is associated with.
correlation	Specify an optional correlation identifier for the message. If the correlation is null, the Workflow Engine creates a correlation identifier based on the item type.

DeferredQueue

Syntax `function DeferredQueue`

Description Returns the name of the queue and schema used by the background engine for deferred processing.

InboundQueue

Syntax `function InboundQueue`

Description Returns the name of the inbound queue and schema. The inbound queue contains messages for the Workflow Engine to consume.

OutboundQueue

Syntax `function OutboundQueue`

Description Returns the name of the outbound queue and schema. The outbound queue contains messages for external agents to consume.

ClearMsgStack

Syntax `procedure ClearMsgStack;`

Description Clears the internal stack. See: Developer APIs for the Inbound Queue: page 8 – 113.

CreateMsg

Syntax

```
procedure CreateMsg
    (itemtype in varchar2,
     itemkey in varchar2,
     actid in number);
```

Description

Creates a new message in the internal stack if it doesn't already exist. See: Developer APIs for the Inbound Queue: page 8 – 113.

Arguments (input)

itemtype	The item type of the message.
itemkey	The item key of the message. An item key is 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 instance.
actid	The function activity instance ID that this message is associated with.

WriteMsg

Syntax

```
procedure WriteMsg
    (itemtype in varchar2,
     itemkey in varchar2,
     actid in number);
```

Description

Writes a message from the internal stack to the inbound queue. See: Developer APIs for the Inbound Queue: page 8 – 113.

Arguments (input)

itemtype	The item type of the message.
itemkey	The item key of the message. An item key is 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.
actid	The function activity instance ID that this message is associated with.

SetMsgAttr

Syntax

```
procedure SetMsgAttr
    (itemtype in varchar2,
     itemkey in varchar2,
     actid in number,
     attrName in varchar2,
     attrValue in varchar2);
```

Description

Appends an item attribute to the message in the internal stack. See: Developer APIs for the Inbound Queue: page 8 – 113.

Arguments (input)

itemtype	The item type of the message.
itemkey	The item key of the message. An item key is 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 instance.
actid	The function activity instance ID that this message is associated with.
attrName	The internal name of the item attribute you wish to append to the message.
attrValue	The value of the item attribute you wish to append.

SetMsgResult

Syntax

```
procedure SetMsgResult
    (itemtype in varchar2,
     itemkey in varchar2,
     actid in number,
     result in varchar2);
```

Description

Sets a result to the message written in the internal stack. See: Developer APIs for the Inbound Queue: page 8 – 113.

Arguments (input)

itemtype	The item type of the message.
itemkey	The item key of the message. An item key is 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 instance.
actid	The function activity instance ID that this message is associated with.
result	The completion result for the message. Possible values are determined by the activity's Result Type.

Document Management APIs

The following document management APIs can be called by user interface (UI) agents to return URLs or javascript functions that enable integrated access to supported document management systems. All supported document management (DM) systems accommodate a URL interface to access documents.

The document management APIs allow you to access documents across multiple instances of the same DM system, as well as across multiple instances of DM systems from different vendors within the same network.

The document management APIs are defined in a PL/SQL package called FND_DOCUMENT_MANAGEMENT:

- get_launch_document_url: page 8 – 135
- get_launch_attach_url: page 8 – 136
- get_open_dm_display_window: page 8 – 137
- get_open_dm_attach_window: page 8 – 138
- set_document_id_html: page 8 – 139

See Also

Standard API for an Oracle Workflow PL/SQL Stored Procedure: page 7 – 2

get_launch_document_url

Syntax

```
procedure get_launch_document_url
    (username in varchar2,
     document_identifier in varchar2,
     display_icon in Boolean,
     launch_document_url out varchar2);
```

Description

Returns an anchor URL that launches a new browser window containing the DM integration screen that displays the specified document. The screen is a frame set of two frames. The upper frame contains a customizable company logo and a toolbar of Oracle Workflow–integrated document management functions. The lower frame displays the specified document.

Arguments (input)

username	The username of the person accessing the document management system.
document_ identifier	The document identifier for the document you wish to display. The document identifier should be stored as a value in an item attribute of type document. You can retrieve the document identifier using the *GetItemAttrDocument* API. See: GetItemAttrDocument: page 8 – 42 and SetItemAttrDocument: page 8 – 38.
display_icon	True or False. True tells the procedure to return the URL with the paper clip attachment icon and translated prompt name, whereas False tells the procedure to return only the URL. This argument provides you the flexibility needed when you call this procedure from a form– or HTML–based UI agent.

get_launch_attach_url

Syntax

```
procedure get_launch_attach_url
    (username in varchar2,
     callback_function in varchar2,
     display_icon in Boolean,
     launch_attach_url out varchar2);
```

Description

Returns an anchor URL that launches a new browser window containing a DM integration screen that allows you to attach a document. The screen is a frame set of two frames. The upper frame contains a customizable company logo and a toolbar of Oracle Workflow–integrated document management functions. The lower frame displays the search screen of the default document management system.

Arguments (input)

username	The username of the person accessing the document management system.
callback_ function	The URL you would like to invoke after the user selects a document to attach. This callback function should be the callback_url syntax that is returned from the *set_document_id_html* API.
display_icon	True or False. True tells the procedure to return the URL with the paper clip attachment icon and translated prompt name, whereas False tells the procedure to return only the URL. This argument provides you the flexibility needed when you call this procedure from a form– or HTML–based UI agent.

get_open_dm_display_window

Syntax `procedure get_open_dm_display_window`

Description Returns a javascript function that displays an attached document from the current UI. The javascript function is used by all the document management functions that the user can perform on an attached document. Each DM function also gives the current DM integration screen a name so that the Document Transport Window can call back to the javascript function in the current window.

get_open_dm_attach_window

Syntax `procedure get_open_dm_attach_window`

Description Returns a javascript function to open a Document Transport Window when a user tries to attach a document in the current UI. The javascript function is used by all the document management functions that the user can perform to attach a document. Each DM function also gives the current DM integration screen a name so that the Document Transport Window can call back to the javascript function in the current window.

set_document_id_html

Syntax

```
procedure set_document_id_html
    (frame_name in varchar2,
     form_name in varchar2,
     document_id_field_name in varchar2
     document_name_field_name in varchar2,
     callback_url out varchar2);
```

Description

Returns a callback URL that gets executed when a user selects a document from the DM system. Use this procedure to set the document that is selected from the document management Search function to the specified destination field of an HTML page. The destination field is the field from which the user launches the DM integration screen to attach a document. Pass the returned callback URL as an argument to the *get_launch_attach_url* API.

Arguments (input)

frame_name The name of the HTML frame that you wish to interact with in the current UI.

form_name The name of the HTML form that you wish to interact with in the current UI.

document_id_field_name The name of the HTML field in the current UI that you would like to write the resulting document identifier to. The resulting document identifier is determined by the document the user selects from the document management Search function. The document identifier is a concatenation of the following values:

DM:<*node_id*>:<*document_id*>:<*version*>

<*nodeid*> is the node ID assigned to the document management system node as defined in the Document Management Nodes web page. See: To Define a Document Node: page 2 – 31.

<*documentid*> is the document ID of the document, as assigned by the document management system where the document resides.

<*version*> is the version of the document. If a version is not specified, the latest version is assumed.

document_name_field_name	The name of the HTML field in the current UI that you would like to write the resulting document name to.

Overview of the Oracle Workflow Notification System

Oracle Workflow communicates with users by sending notifications. Notifications contain messages that may request users to take some type of action and/or provide users with information. You define the notification activity and the notification 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 the Notifications web page in an HTML browser. A user can also receive notifications in their E–mail applications. E–mail notifications can contain HTML content or include other documents as optional attachments. The Notification System delivers the messages and processes the incoming responses.

Notification Model

A notification activity in a workflow process consist of a design–time message and a list of message attributes. In addition, there may be a number of runtime named values called item type attributes from which the message attributes draw their values.

The Workflow Engine moves through the workflow process, evaluating each activity in turn. Once it encounters a notification activity, the engine makes a call to the Notification System *Send*( ) or *SendGroup( )* API to send the notification.

Sending Notification Messages

The *Send*( ) or *SendGroup*( ) API are called by the Workflow Engine when it encounters a notification activity. These APIs do the following:

- Check that the performer role of the notification activity is valid.
- Identify the notification preference for of the performer role.
- Look up the message attributes for the message.
  - If a message attribute is of source SEND, the *Send*() or *SendGroup*( ) API retrieves its value from the item type attribute that the message attribute references. If the procedure cannot find an item type attribute, it uses the default value of the message attribute, if available. The Subject and Body of the message may include message attributes of source SEND, which the *Send*( ) or *SendGroup*( )

API token replaces with each attribute's current value when creating the notification.

– If a message includes a message attribute of source RESPOND, the *Send*( ) or *SendGroup*() API checks to see if it has a default value assigned to it. The procedure then uses these RESPOND attributes to create the default response section of the notification.

- 'Construct' the notification content by inserting relevant information into the Workflow Notification tables.
- Update the notification activity's status to 'NOTIFIED' if a response is required or to 'COMPLETE' if no response is required.

  Note: If a notification activity sends a message that is for the performer's information only (FYI), 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.

  Note: In the case of a voting activity, the status is updated to 'WAITING' instead of 'NOTIFIED'. See: Special Handling of Voting Activities: page 8 – 144

If the performer role of a notification has a notification preference of MAILTEXT, MAILHTML, MAILATTH or SUMMARY, the notification is flagged with the corresponding value in the Notification table. The Notification Mailer, which polls the Notification table for these flags, then generates an E–mail version of that notification and sends it to the performer. See: Implementing the Notification Mailer: page 2 – 38.

Users who view their notifications from the Notifications Web page, regardless of their notifications preferences, are simply querying the Workflow Notification tables from this interface.

A notification recipient can perform one of four actions with the notification:

- Respond to the notification or close the notification if it does not require a response. See: Processing a Notification Response: page 8 – 143.
- Forward the notification to another role. See: Forwarding a Notification: page 8 – 143.
- Transfer ownership of the notification to another role. See: Transferring a Notification: page 8 – 144.

- Ignore the notification and let it time out. See: Processing a Timed Out Notification: page 8 – 144.

Processing a Notification Response

After a recipient responds, the Notifications web page or Notification Mailer assigns the response values to the notification response attributes and calls the notification *Respond*( ) API. The *Respond( )* API first calls a notification callback function to execute the notification activity's post–notification function (if it has one) in RESPOND mode. The post–notification function may interpret the response and perform tightly–coupled post–response processing. If the post–notification function raises an exception, the response is aborted. See: Post–notification Functions: page 8 – 10.

If no exception is raised, *Respond( )* marks the notification as closed and then calls the notification callback function again in SET mode to update the corresponding item attributes with the RESPOND notification attributes values. If the notification message prompts for a response that is specified in the Result tab of the message's property page, that response value is also set as the result of the notification activity.

Finally, *Respond*( ) calls *WF_ENGINE.CompleteActivity*( ) to inform the engine that the notification activity is complete so it can transition to the next qualified activity.

Forwarding a Notification

If a recipient forwards a notification to another role, the Notifications web page calls the Notification System's *Forward( )* API.

> **Note:** The Notification System is not able to track notifications that are forwarded via E–mail. It records only the eventual responder's E–mail address and any Respond message attributes values included in the response.

The *Forward( )* API validates the role, then calls a notification callback function to execute the notification activity's post–notification function (if it has one) in FORWARD mode. As an example, the post–notification function may verify whether the role that the notification is being forwarded to has appropriate authority to view and respond to the notification. If it doesn't, the post–notification function may return an error and prevent the Forward operation from proceeding. See: Post–notification Functions: page 8 – 10.

Forward( ) then forwards the notification to the new role, along with any appended comments.

> **Note:** *Forward( )* does not update the owner or original recipient of the notification.

Transferring a Notification

If a recipient transfers the ownership of a notification to another role, the Notification web page calls the Notification System's *Transfer( )* API.

> **Note:** Recipients who view notifications from an E–mail application cannot transfer notifications. To transfer a notification, the recipient must use the Notifications web page.

The *Transfer( )* API validates the role, then calls a notification callback function to execute the notification activity's post–notification function (if it has one) in TRANSFER mode. As an example, the post–notification function may verify whether the role that the notification is being transferred to has appropriate authority. If it doesn't, the post–notification function may return an error and prevent the Transfer operation from proceeding. See: Post–notification Functions: page 8 – 10.

Transfer( ) then assigns ownership of the notification to the new role, passing along any appended comments. Note that a transfer is also recorded in the comments of the notification.

Processing a Timed Out Notification

Timed out notification or subprocess activities are initially detected by the background engine. Background engines set up to handle timed out activities periodically check for activities that have time out values specified. If an activity does have a time out value, and the current date and time exceeds that time out value, the background engine marks that activity's status as 'TIMEOUT' and calls the Workflow Engine. The Workflow Engine then resumes by trying to execute the activity to which the <Timeout> transition points.

Special Handling of Voting Activities

A voting activity by definition is a notification activity that:

- Has its roles expanded, so that an individual copy of the notification message is sent to each member of the Performer role.

- Has a message with a specified Result, that requires recipients to respond from a list of values.
- Has a post–notification function associated with it that contains logic in the RUN mode to process the polled responses from the Performer members to generate a single response that the Workflow Engine interprets as the result of the notification activity. See: Voting Activity: page 4 – 50.

Once the Notification System sends the notification for a voting activity, it marks the voting activity's status as 'NOTIFIED'. The voting activity's status is updated to 'WAITING' as soon as some responses are received, but not enough responses are received to satisfy the voting criteria.

The individual role members that each receive a copy of the notification message can then respond or forward the notification if they use either of the two notification interfaces to view the notification. They can also transfer the notification if they use the Notifications web page.

The notification user interface calls the appropriate *Respond()*, *Forward()*, or *Transfer()* API depending on the action that the performer takes. Each API in turn calls the notification callback function to execute the post–notification function in RESPOND, FORWARD, or TRANSFER mode, respectively. When the Notification System finishes executing the post–notification function in FORWARD or TRANSFER mode, it carries out the Forward or Transfer operation, respectively.

When the Notification System completes execution of the post–notification function in RESPOND mode, the Workflow Engine the runs the post–notification function again in RUN mode. It calls the function in RUN mode after all responses are received to execute the vote tallying logic.

Also if the voting activity is reset to be reexecuted as part of a loop, or if it times out, the Workflow Engine runs the post–notification function in CANCEL or TIMEOUT mode, respectively. The logic for TIMEOUT mode in a voting activity's post–notification function should identify how to tally the votes received up until the timeout.

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.

Many of these Notification APIs also have corresponding Java methods that you can call from any Java program to integrate with Oracle Workflow. The following list indicates whether the Notification APIs are available as PL/SQL functions/procedures, as Java methods, or both. See: Oracle Workflow Java Interface: page 8 – 4.

Attention: Java is case–sensitive and all Java method names begin with a lower case letter to follow Java naming conventions.

- Send: page 8 – 148—PL/SQL and Java
- SendGroup: page 8 – 152—PL/SQL
- Forward: page 8 – 154—PL/SQL and Java
- Transfer: page 8 – 155—PL/SQL and Java
- Cancel: page 8 – 156—PL/SQL and Java
- CancelGroup: page 8 – 157—PL/SQL
- Respond: page 8 – 158—PL/SQL and Java
- Responder: page 8 – 159—PL/SQL and Java
- VoteCount: page 8 – 160—PL/SQL and Java
- OpenNotificationsExist: page 8 – 161—PL/SQL and Java
- Close: page 8 – 162—PL/SQL and Java
- AddAttr: page 8 – 163—PL/SQL and Java
- SetAttribute: page 8 – 164—PL/SQL and Java
- GetAttrInfo: page 8 – 166—PL/SQL and Java
- GetInfo: page 8 – 167—PL/SQL and Java
- GetText: page 8 – 168—PL/SQL
- GetShortText: page 8 – 169—PL/SQL
- GetAttribute: page 8 – 170—PL/SQL and Java
- GetAttrDoc: page 8 – 171—PL/SQL and Java
- GetSubject: page 8 – 172—PL/SQL and Java
- GetBody: page 8 – 173—PL/SQL and Java

- GetShortBody: page 8 – 174—PL/SQL
- TestContext: page 8 – 175—PL/SQL
- AccessCheck: page 8 – 176—PL/SQL and Java
- WorkCount: page 8 – 177—PL/SQL and Java
- GetNotifications: page 8 – 178—Java
- GetNotificationAttributes: page 8 – 179—Java

Send

PL/SQL 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;
```

Java Syntax

```
public static BigDecimal send
    (WFContext wCtx,
     String role,
     String messagType,
     String messageName,
     String dueDate,
     String callback,
     String context,
     string sendComment,
     BigDecimal priority)
```

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, 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.

> **Note:** If you are using the Oracle Workflow Notification System and its E–mail–based, web–based or forms–based notification client, the *Send* procedure implicitly calls the *WF_ENGINE.CB* callback function. If you are using your own custom notification system that does not call the Workflow Engine, then you must define your own callback function following a standard format and specify its name for the callback argument. See: Custom Callback Function: page 8 – 149.

Arguments (input)	**wCtx**	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
	role	The role name assigned as the performer of the notification activity.
	msg_type or messagType	The item type associated with the message.
	msg_name or messageName	The message internal name.
	due_date or dueDate	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 or sendComment	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.

Custom Callback Function

A default callback function can be called at various points by the actions of the WF_NOTIFICATION APIs. You may provide your own custom callback function, but it 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, ERROR, TESTCTX, FORWARD, TRANSFER, or RESPOND as the action requested. 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, ERROR to set the associated notification activity to a status of 'ERROR', TESTCTX to test the current context by calling the item type's Selector/Callback function, FORWARD to execute the post–notification function in FORWARD mode, TRANSFER to execute the post–notification function in TRANSFER mode, and RESPOND to execute the post–notification function in RESPOND mode.
context	The context passed to *SEND*( ) or *SendGroup( )*. The format is <*itemtype*>:<*itemkey*>:<*activityid*>.
attr_name	An attribute name to set/get if command is GET or SET.
attr_type	An attribute type if command is SET or GET.
text_value	Value of a text attribute if command is SET or value of text attribute returned if command is GET.
number_value	Value of a number attribute if command is SET or value of a number attribute returned if command is GET.
date_value	Value of a date attribute if command is SET or value of a date attribute returned if command GET.

Note: The arguments text_value, number_value, and date_value are mutually exclusive. That is, use only one of these arguments depending on the value of the attr_type argument.

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', 'NUMBER', textval,
numval, dateval)
```

Example 2 When the user responds to the notification, the callback is called again, once for each RESPOND attribute.

```
   your_callback('SET', context, 'STATUS', 'TEXT',
'COMPLETE', numval, dateval);
```

Example 3 Then finally the Notification System calls the 'COMPLETE' command to indicate the response is complete.

```
   your_callback('COMPLETE', context, attrname, attrtype,
textval, numval, dateval);
```

SendGroup

PL/SQL 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, 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.

> **Note:** If you are using the Oracle Workflow Notification System and its E–mail–based, web–based or forms–based notification client, the *Send* procedure implicitly calls the *WF_ENGINE.CB* callback function. If you are using your own custom notification system, then you must define your own callback function following a standard format and specify its name for the callback argument. See: Custom Callback Function: page 8 – 149.

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 4 – 50.

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 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

PL/SQL Syntax

```
procedure FORWARD
(nid in number,
 new_role in varchar2,
 forward_comment in varchar2 default null);
```

Java Syntax

```
public static boolean forward
    (WFContext wCtx,
     BigDecimal nid,
     String newRole
     String comment)
```

Description

This procedure delegates a notification to a new role to perform work, even though the original role recipient still maintains ownership of the notification activity. Also implicitly calls the Callback function specified in the Send or SendGroup function with FORWARD mode. A comment can be supplied to explain why the forward is taking place. Existing notification attributes (including due date) are not refreshed or otherwise changed. The Delegate feature in the Notification System calls this procedure. Note that when you forward a notification, the forward is recorded in the USER_COMMENT field of the notification.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
nid	The notification id.
new_role or newRole	The role name of the person the note is reassigned to.
forward_comment or comment	An optional forwarding comment.

Transfer

PL/SQL Syntax

```
procedure TRANSFER
(nid in number,
 new_role in varchar2,
 forward_comment in varchar2 default null);
```

Java Syntax

```
public static boolean transfer
    (WFContext wCtx,
     BigDecimal nid,
     String newRole
     String comment)
```

Description

This procedure forwards a notification to a new role and transfers ownership of the notification to the new role. It also implicitly calls the Callback function specified in the Send or SendGroup function with TRANSFER mode. A comment can be supplied to explain why the forward is taking place. The Transfer feature in the Notification System calls this procedure. Note that when you transfer a notification, the transfer is recorded in the USER_COMMENT field of the notification.

Attention: Existing notification attributes (including due date) are not refreshed or otherwise changed except for ORIGINAL_RECIPIENT, which identifies the owner of the notification.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
nid	The notification id.
new_role or newRole	The role name of the person the note is transferred to.
forward_comment or comment	An optional comment to append to notification.

Cancel

PL/SQL Syntax

```
procedure CANCEL
(nid in number,
 cancel_comment in varchar2 default null);
```

Java Syntax

```
public static boolean cancel
    (WFContext wCtx,
     BigDecimal nid,
     String comment)
```

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)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
nid	The notification id.
cancel_comment or comment	An optional comment on the cancellation.

CancelGroup

PL/SQL 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 4 – 50.

Arguments (input)

gid	The notification group id.
cancel_comment	An optional comment on the cancellation.

Respond

PL/SQL Syntax

```
procedure  RESPOND
(nid in number,
 respond_comment in varchar2 default null,
 responder in varchar2 default null);
```

Java Syntax

```
public static boolean respond
    (WFContext wCtx,
     BigDecimal nid,
     String comment,
     String responder)
```

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).

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)

wCtx Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.

nid The notification id

comment An optional comment on the response

responder The user who responded to the notification.

Responder

PL/SQL Syntax

```
function RESPONDER
(nid in number)
returns varchar2;
```

Java Syntax

```
public static String responder
    (WFContext wCtx,
     BigDecimal nid)
```

Description

This function returns the responder of a closed notification.

If the notification was closed using the 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. See: Respond: page 8 – 158.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
nid	The notification id

VoteCount

PL/SQL Syntax

```
procedure VoteCount
  (gid in number,
   ResultCode in varchar2,
   ResultCount out number,
   PercentOfTotalPop out number,
   PercentOfVotes out number);
```

Java Syntax

```
public static WFTwoDDataSource voteCount
    (WFContext wCtx,
     BigDecimal nid,
     String resultCode)
```

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 4 – 50.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
gid	The notification group id.
ResultCode	Result code to be tallied.

OpenNotificationsExist

PL/SQL Syntax

```
function OpenNotificationsExist
  (gid in number)
   return boolean;
```

Java Syntax

```
public static boolean openNotificationsExist
    (WFContext wCtx,
     BigDecimal gid)
```

Description

This function returns 'TRUE' if any notification associated with the specified notification group ID is 'OPEN', otherwise it returns 'FALSE'.

Use this procedure only if you are writing your own custom Voting activity. See: Voting Activity: page 4 – 50.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
gid	The notification group id.

Close

PL/SQL Syntax

```
procedure Close
(nid in number,
 responder in varchar2 default null);
```

Java Syntax

```
public static boolean close
    (WFContext wCtx,
    BigDecimal nid,
    String responder)
```

Description This procedure Closes a notification.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
nid	The notification id.
responder	The user or role who responded to the notification.

AddAttr

PL/SQL Syntax

```
procedure AddAttr
  (nid in number,
   aname in varchar2);
```

Java Syntax

```
public static boolean addAttr
    (WFContext wCtx,
     BigDecimal nid,
     String aName)
```

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)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
nid	The notification id.
aname	The attribute name.
avalue	The attribute value.

SetAttribute

PL/SQL 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);
```

Java Syntax

```
public static boolean setAttrText
    (WFContext wCtx,
     BigDecimal nid,
     String aName,
     String aValue)

public static boolean setAttrNumber
    (WFContext wCtx,
     BigDecimal nid,
     String aName,
     BigDecimal aValue)

public static boolean setAttrDate
     WFContext wCtx,
     BigDecimal nid,
     String aName,
     String aValue)
```

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)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.

nid	The notification id.
aname	The attribute name.
avalue	The attribute value.

GetAttrInfo

PL/SQL Syntax

```
procedure GetAttrInfo
(nid in number,
 aname in varchar2,
 atype out varchar2,
 subtype out varchar2,
 format out varchar2);
```

Java Syntax

```
public static WFTwoDataSource getAttrInfo
    (WFContext wCtx,
     BigDecimal nid,
     String aName)
```

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)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
nid	The notification id.
aname	The attribute name.

GetInfo

PL/SQL 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);
```

Java Syntax

```
public static WFTwoDDataSource getInfo
    (WFContext wCtx,
     BigDecimal nid)
```

Description

Returns the role that the notification is sent to, the item type of the message, the name of the message, the notification priority, the due date and the status for the specified notification.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
nid	The notification id.

GetText

PL/SQL Syntax

```
function GetText
  (some_text in varchar2,
   nid in number,
   disptype in varchar2 default '')
  return varchar2;
```

Description

Substitutes tokens in an arbitrary text string using token values from a particular notification. This function may return up to 32K characters. You cannot use this function in a view definition or in an Oracle Forms Developer form. For views and forms, use *GetShortText*( ) which truncates values at 1950 characters.

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.

disptype The display type of the message body that you are token substituting the text into. Valid display types are:

- wf_notification.doc_text, which returns text/plain
- wf_notification.doc_html, which returns text/html
- wf_notification.doc_attach, which returns null

The default is null.

GetShortText

PL/SQL Syntax

```
function GetShortText
  (some_text in varchar2,
   nid in number)
  return varchar2;
```

Description

Substitutes tokens in an arbitrary text string using token values from a particular notification. This function may return up to 1950 characters. This function is meant for use in view definitions and Oracle Forms Developer forms, where the field size is limited to 1950 characters. Use *GetText*( ) in other situations where you need to retrieve up to 32K characters.

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

PL/SQL 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;
```

Java Syntax

```
public static String getAttrText
    (WFContext wCtx,
     BigDecimal nid,
     String aName)

public static BigDecimal getAttrNumber
    (WFContext wCtx,
     BigDecimal nid,
     String aName)

public static String getAttrDate
     WFContext wCtx,
     BigDecimal nid,
     String aName)
```

Description

Returns the value of the specified message attribute.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
nid	The notification id.
aname	The message attribute name.

GetAttrDoc

PL/SQL Syntax

```
function GetAttrDoc
  (nid in number,
   aname in varchar2,
   disptype in varchar2)
   return varchar2;
```

Java Syntax

```
public static String getAttrDoc
    (WFContext wCtx,
     BigDecimal nid,
     String aName,
     String dispType)
```

Description

Returns the displayed value of a Document–type attribute. The referenced document appears in either plain text or HTML format, as requested.

If you wish to retrieve the actual attribute value, that is, the document key string instead of the actual document, use *GetAttrText( )*.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
nid	The notification id.
aname	The message attribute name.
disptype	The display type of the document you wish to return. Valid display types are: • wf_notification.doc_text, which returns text/plain • wf_notification.doc_html, which returns text/html • wf_notification.doc_attach, which returns null

GetSubject

PL/SQL Syntax

```
function GetSubject
(nid in number)
 return varchar2
```

Java Syntax

```
public static String getSubject
    (WFContext wCtx,
     BigDecimal nid)
```

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.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
nid	The notification id

GetBody

PL/SQL Syntax

```
function GetBody
(nid in number,
 disptype in varchar2 default '')
 return varchar2;
```

Java Syntax

```
public static String getBody
    (WFContext wCtx,
     BigDecimal nid,
     String dispType)
```

Description

Returns the HTML or plain text message body for the notification, depending on the message body type specified. Any message attribute in the body is token substituted with the value of the corresponding notification attribute. This function may return up to 32K characters. You cannot use this function in a view definition or in an Oracle Applications form. For views and forms, use *GetShortBody*( ) which truncates values at 1950 characters.

Note that the returned plain text message body is *not* formatted; it should be wordwrapped as appropriate for the output device. Body text may contain tabs (which indicate indentation) and newlines (which indicate paragraph termination).

Arguments (input)

wCtx Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.

nid The notification id.

disptype The display type of the message body you wish to fetch. Valid display types are:

- wf_notification.doc_text, which returns text/plain
- wf_notification.doc_html, which returns text/html
- wf_notification.doc_attach, which returns null

The default is null.

GetShortBody

PL/SQL Syntax

```
function GetShortBody
(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 notification attribute. This function may return up to 1950 characters. This function is meant for use in view definitions and Oracle Forms Developer forms, where the field size is limited to 1950 characters. Use *GetBody*( ) in other situations where you need to retrieve up to 32K characters.

Note that the returned plain text message body is *not* formatted; it should be wordwrapped as appropriate for the output device. Body text may contain tabs (which indicate indentation) and newlines (which indicate paragraph termination).

If an error is detected, this function returns the body unsubstituted or null if all else fails, rather than raise exceptions.

> **Note:** This function is intended for displaying messages in forms or views only.

Argument (input)

nid	The notification id.

TestContext

PL/SQL Syntax

```
function TestContext
(nid in number)
 return boolean;
```

Description

Tests if the current context is correct by calling the Item Type Selector/Callback function. This function returns TRUE if the context check is OK, or if no Selector/Callback function is implemented. It returns FALSE if the context check fails.

Argument (input)

nid The notification id.

AccessCheck

PL/SQL Syntax

```
function AccessCheck
(access_str in varchar2)
 return varchar2;
```

Java Syntax

```
public static String accessCheck
    (WFContext wCtx,
     String accessString)
```

Description

Returns a username if the notification access string is valid and the notification is open, otherwise it returns null. The access string is automatically generated by the Notification Mailer and is used to verify the authenticity of both text and HTML versions of E–mail notifications.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
access_str or accessString	The access string, in the format: *nid/nkey* where *nid* is the notification ID and *nkey* is the notification key.

WorkCount

PL/SQL Syntax

```
function WorkCount
(username in varchar2)
 return number;
```

Java Syntax

```
public static BigDecimal workCount
    (WFContext wCtx,
    String userName)
```

Description Returns the number of open notifications assigned to a role.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
username	The internal name of a role.

GetNotifications

Java Syntax

```
public static WFTwoDDataSource getNotifications
    (WFContext wCtx,
    String itemType,
    String itemKey)
```

Description Returns a list of notifications for the specified item type and item key.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
itemType	The internal name of the item type.
itemKey	A string derived from the application object's primary key. The string uniquely identifies the item within the item type. The item type and key together identify the process instance.

GetNotificationAttributes

Java Syntax

```
public static WFTwoDDataSource getNotificationAttributes
    (WFContext wCtx,
    BigDecimal nid)
```

Description Returns a list of notification attributes and their corresponding values for the specified notification ID.

Arguments (input)

wCtx	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 4.
nid	The notification ID.

CHAPTER

9

Oracle Workflow Home Page

This chapter discusses the Oracle Workflow home page, where users and administrators can centrally access all the web–based features of Oracle Workflow.

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.

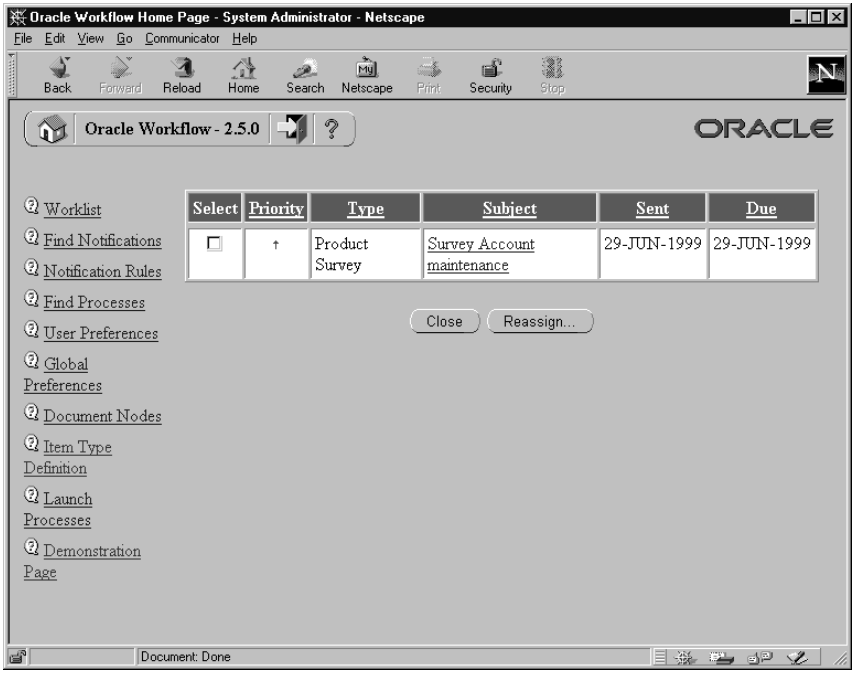
▶ To Access the Oracle Workflow Home Page

1. Use a web browser to connect to the URL for the home page:

   `<webagent>/wfa_html.home`

   `<webagent>` represents the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 12.

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



2. The web page identifies the current version of Oracle Workflow.

   The web page also displays your current Worklist of notifications to provide you quick access to the work that is awaiting.

3. A toolbar appears in the upper left corner of the Oracle Workflow home page, as well as on every other Oracle Workflow web page. The Home icon returns you to the Oracle Workflow home page. The name of the current page appears in the middle of the toolbar. The Logout icon logs you out of your current Oracle Workflow web session and the Help icon displays online help for the current screen.

Attention: If your Oracle Workflow web security is set up using Oracle Application Server, the Logout icon is not available, because Oracle Application Server does not support logging out of a web session.

4. Choose the Worklist link to redisplay your list of workflow notifications in the entire screen. You can close or reassign your notifications directly from the Worklist or you can drill down to the details of each specific notification and close, reassign, or respond to them individually. See: To View Notifications from the Worklist: page 10 – 17.

5. Choose the Find Notifications link to locate notifications that match specific criteria and act on those notifications. See: To Find Notifications: page 10 – 14.

6. Choose the Notification Rules link to view and define your automatic notification routing rules. If you are logged in as a role with workflow administrator privileges, the Find Automatic Notification Processing Rules web page appears, letting you first display the routing rules for the role you specify. See: To Define a Rule for Automatic Notification Routing: page 10 – 24.

7. Choose the Find Processes link to 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 11 – 8.

8. Choose the User Preferences link to set the preferences that control how you interact with Oracle Workflow. See: Setting User Preferences: page 9 – 4.

9. If you are logged in as a role with workflow administrator privileges, you can choose the Global Preferences link to set global preferences that control how users interact with Oracle Workflow. See: Setting Global User Preferences: page 2 – 12.

10. If you are logged in as a role with workflow administrator privileges, you can choose the Document Nodes link to define the document management system nodes that you want Oracle

Workflow to be able to access. See: Defining Document Management Repositories: page 2 – 31.

11. Choose the Item Type Definition link to access the Find Item Type web page. Use the Find Item Type web page to query for a specific item type definition to display in the Item Type Definition page. See: Item Type Definition Page: page 3 – 21.
12. If you are logged in as a role with workflow administrator privileges, you can choose the Launch Processes link to test a specific workflow process definition. See: Testing Workflow Definitions: page 12 – 2.
13. If you are logged in as a role with workflow administrator privileges, you can choose the Demonstration Page link to access the Demonstration home page. You can use the Demonstration home page to launch any of the demonstration workflow processes provided with Oracle Workflow. See: Sample Workflow Processes: page 13 – 2.

Setting User Preferences

You can control how you interact with Oracle Workflow by specifying user preferences that you can set from the User Preferences web page. The values that you specify in the User Preferences web page override the default global values set by your workflow administrator in the Global User Preferences web page.

▶ **To Set User Preferences**

1. Use a web browser to connect to the Oracle Workflow home page:

   `<webagent>/wfa_html.home`

   From the Oracle Workflow home page, choose the User Preferences link.

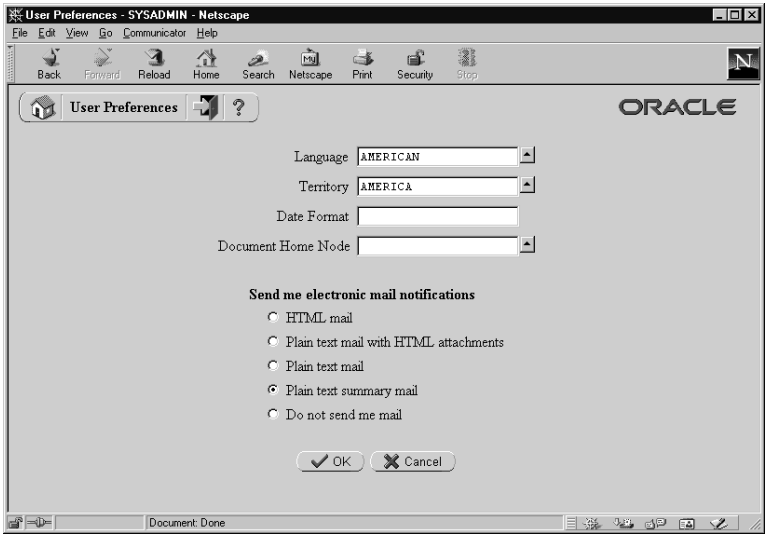
   Alternatively, you can connect directly to the User Preferences web page:

   `<webagent>/wf_pref.edit`

   `<webagent>` represents the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 12.

Attention: These are secured pages, so if you have not yet logged on as a valid user in the current web session, you will be prompted to do so before the page appears.

2. The User Preferences web page displays a summary of your current user preferences. Choose Update to modify these preferences.



3. In the Language and Territory fields, use the list of values to select the NLS_LANGUAGE and NLS_TERRITORY combination that defines the default language–dependent behavior and territory–dependent formatting of your notification sessions.

4. In the Date Format field, specify an Oracle8–compliant date format to use for your database session. An example of an Oracle8–compliant date format is DD–Mon–RRRR. If you do not specify a date format, then the date format defaults to DD–MON–YYYY.

   Note: Oracle Workflow may include a time element when relevant for certain displayed dates, even if you do not include a time format with your date format. If you specify a time format along with your date format, then in those situations when Oracle Workflow displays a time element, you will see two time elements following your date.

5. In the Document Home Node field, use the list of values to select the default Document Management repository that you want to access when you attempt to attach or access an attached document management document.

6. In the 'Send me electronic mail notifications' section, check a notification preference:

   – HTML mail—send you notifications as HTML E–mail. Your mail reader must be able to display HTML formatting in the message body.

   – Plain text mail with HTML attachments—send you notifications as plain text E–mail but include the HTML–formatted version of the notifications as attachments.

   – Plain text mail—send you notifications as plain text E–mail.

   – Plain text summary mail—send you a summary of all notifications as plain text E–mail. You must use the Notifications web page to take action on individual notifications.

   – Do not send me mail—do not send you notifications as E–mail. You must view the notifications and take action from the Notifications web page.

7. Check OK once you are satisfied with your changes.

See Also

Notification Preferences: page 2 – 39

CHAPTER

10

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 three 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 or query the Workflow Notifications Web page for your notifications. See: Setting Up an Oracle Workflow Directory Service: page 2 – 17.

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, documents from third party document management vendors 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 manage the notifications for you.

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 'Plain text mail', 'HTML mail', or 'Plain text mail with attachments' in the User Preferences web page and your workflow administrator sets up the Notification Mailer to run.

If your E–mail reader can only support plain text messages with no attachments, set your notification preference to 'Plain text mail'.

If your E–mail reader can interpret and display HTML–formatting in the body of a message, select 'HTML mail' as your notification preference. HTML mail provides direct links to supporting

information sources that you may need access to to complete a notification.

If your E–mail reader can only display plain text in the body of a message, but can also display attachments to the message, set your notification preference to 'Plain text mail with Attachments'.

An E–mail notification that requires a response maintains an 'Open' status until you respond to the notification. If an E–mail notification does not require a response, such as an FYI (For your information) notification, Oracle Workflow automatically updates its status to 'Closed'. Once you read a 'Closed' message, you can delete it from your inbox.

There are two response methods for plain text E–mail notifications: templated response or direct response. Your workflow administrator determines the response method for your organization when setting up the Notification Mailer. For the templated response method, you reply using the template of response prompts provided in the notification and enter your response values between the double quotes (" ") following each prompt. For the direct response method, you enter your response values directly as the first lines of your reply.

Both templated and direct response E–mail notifications are based on standard message templates defined in Oracle Workflow Builder. Both describe the syntax the reply should follow and list the information needed to confirm the notification. Both types of messages also include any custom site information, the due date of the notification, and any information necessary to process the response. See: Modifying Your Message Templates: page 2 – 56.

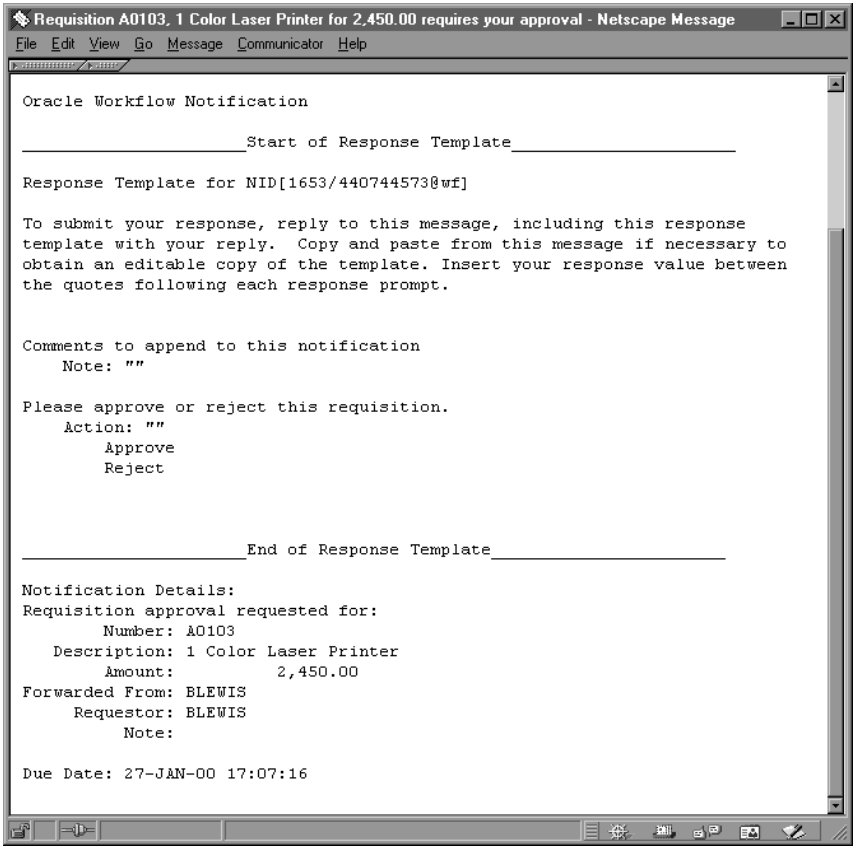
When you respond to a notification by E–mail, your reply message must include the notification ID (NID) and access key from the original notification message. The Notification Mailer can process your response properly only if you include the correct NID and access key combination in your response. You can ensure that your reply contains the NID and access key either by including the entire original message in your reply or by using a response template that includes the NID line.

> **Note:** The notification access key is a distinct random key generated by the Notification System for each NID. The access key serves as a password that allows only users who actually received the notification containing the key to respond to that notification.

See Also

Starting the Notification Mailer: page 2 – 44

▶ To Respond to a Plain Text E–mail Notification Using Templated Response



1. Your plain text E–mail notification includes information that is helpful for you to respond to the notification. Depending on the notification, the information may appear as references to other sources or as attachments. Some attachments, depending on their content may only be viewable if you display your notification using the Notification web pages. See: Viewing Notifications from a Web Browser: page 10 – 13.
2. To respond to your notification, use the Reply command in your mail application to reply to the original E–mail notification.

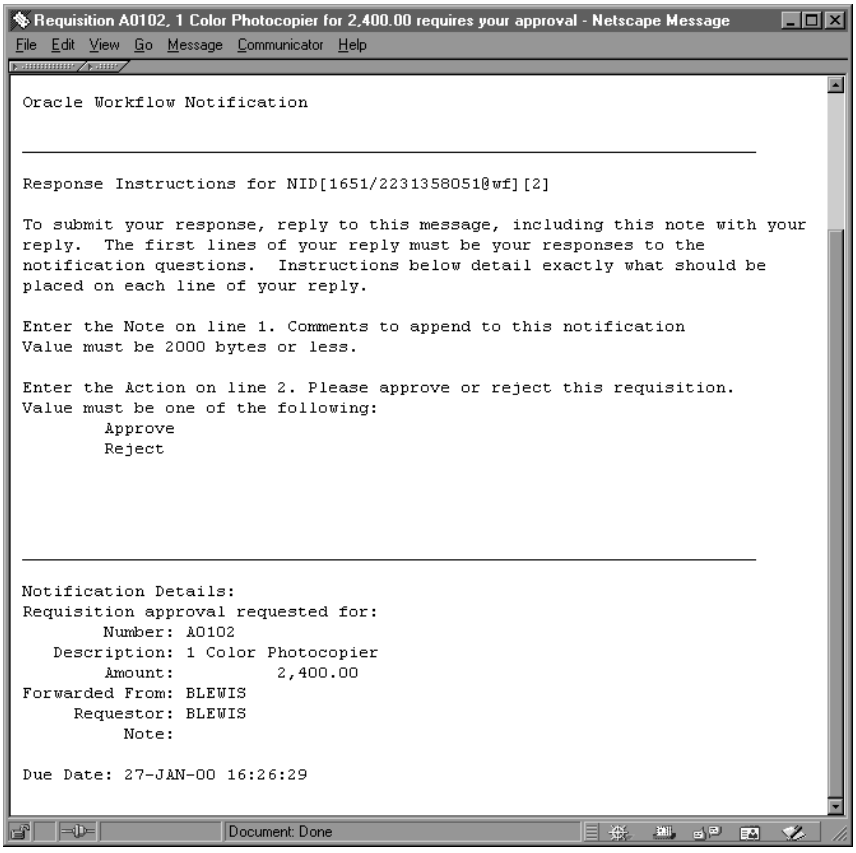
3. Include the response template from the original notification in your reply. In addition to the response prompts, the response template includes the special notification ID and access key that the Notification Mailer requires to identify the notification you are responding to. If your mail application includes an editable copy of the original message when it generates the reply message, you can use that copy to enter your response values. Otherwise, copy and paste from the original message to obtain a copy of the response template that you can edit.
4. Follow the response template instructions and insert your response values between the double quotes (" ") following each response prompt. The Notification System interprets your response values literally, so a value in uppercase is interpreted differently from the same value in lowercase.
5. When you are satisfied with your response, use the Send command of the mail application to send your reply.

   Note: 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.

See Also

Plain Text E–mail: page 2 – 40

▶ To Respond to a Plain Text E–mail Notification Using Direct Response



1. Your plain text E–mail notification includes information that is helpful for you to respond to the notification. Depending on the notification, the information may appear as references to other sources or as attachments. Some attachments, depending on their content may only be viewable if you display your notification using the Notification web pages. See: Viewing Notifications from a Web Browser: page 10 – 13.
2. To respond to your notification, use the Reply command in your mail application to reply to the original E–mail notification.
3. Include the text of the original notification message in your reply. This text contains the special notification ID and access key that the Notification Mailer requires to identify the notification you are responding to.

4. Follow the syntax instructions in the notification message 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 (" "). 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 incorrectly interpreted 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.

5. When you are satisfied with your response, use the Send command of the mail application to send your reply.

   Note: 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 10 – 1 (Page 1 of 1)

Valid Response A – Approve
Approve
Let me know if this item meets expectations.
01–JAN–1998
1000.00

Table 10 – 2 (Page 1 of 1)

Valid Response B – Reject
Reject
Too expensive.

Table 10 – 3 (Page 1 of 1)

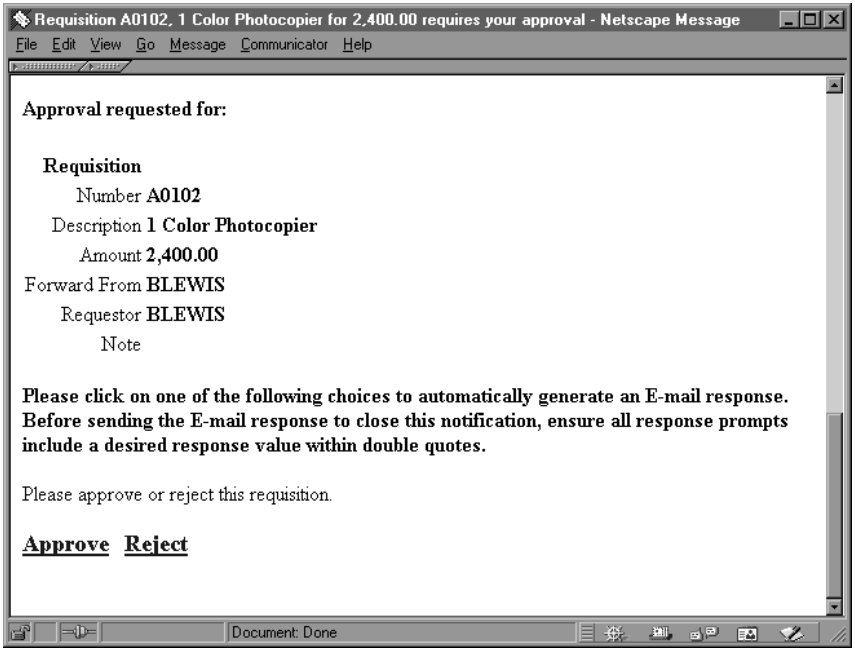
Valid Response C – Reject
Reject
”This item is too expensive. Please find a replacement that is of lower cost, or else
include additional justification for why this item should be approved.”
01–JAN–1998
1000.00

Table 10 – 4 (Page 1 of 1)

See Also

Plain Text E–mail: page 2 – 40

▶ To Respond to an HTML E–mail Notification



1. Your HTML–formatted E–mail notification includes information that is helpful for you to respond to the notification. Depending on the notification, the information may appear as links to other sources or as attachments.

Attention: An HTML–formatted E–mail notification always includes one attachment. The attachment is called Notification Detail Link and it provides a direct link to your notification in the Notification Details web page.

Note that your Web browser must support JavaScript and Frames to open this attachment. When you open the Notification Detail Link attachment, it automatically attempts to establish a web session with your web server. In doing so, it authenticates your access, verifies that your notification is still open, and displays a message if the notification is already closed.

You may respond directly to your notification from this Notification Details page, bypassing the need to process your response through the Notification Mailer. If you decide to respond from the Notification Details page, skip the remaining steps. See: To View the Details of a Notification: page 10 – 18.

2. When you are done reviewing all the information for the notification, click on one of the response links shown at the end of the notification.

3. Each response link automatically generates a plain text E–mail reply. The reply contains the correct Reply To: E–mail address as well as a response template in the message body. The response template consists of the required notification ID and access key that identify the notification you are responding to and a response prompt edited with your selected response.

Warning: Do not include any HTML–formatting in the E–mail response.

4. Depending on the notification, the auto–generated E–mail response template may also prompt you for other information in addition to your selected response. Supply responses by editing the response value text between the double quotes (" ") following each response prompt.

5. When you are satisfied with your response, use the Send command of the mail application to send your reply.

Note: 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.

See Also

HTML–Formatted E–mail: page 2 – 41

▶ To Respond to a Plain Text E–mail Notification with an HTML Attachment

1. Your plain text E–mail notification with attachments includes information that is helpful for you to respond to the notification. Depending on the notification, the information may appear inline in the message body, as links to other reference sources or as attachments to the message. In addition, the notification always includes at least two attachments:

   - HTML Message Body—an HTML–formatted version of the notification message.

- Notification Detail Link—a direct link to your notification displayed in the Notification Details web page.

2. When you are done reviewing all the information for the notification, you can respond to the notification in one of three ways:

   - Use your mail reader's Reply command to respond, following the instructions in the plain text message body. See: To Respond to a Plain Text E–mail Notification Using Templated Response: page 10 – 4 and To Respond to a Plain Text E–mail Notification Using Direct Response: page 10 – 6.
   - Display the HTML Message Body attachment and respond by selecting one of the response links at the bottom of the HTML message body. See: To Respond to an HTML E–mail Notification: page 10 – 10.
   - Choose the Notification Detail Link attachment to display the Notification Details web page. See: To View the Details of a Notification: page 10 – 18.

     Note: Your Web browser must support JavaScript and Frames to open this attachment. When you open the Notification Detail Link attachment, it automatically attempts to establish a web session with your web server. In doing so, it authenticates your access, verifies that your notification is still open, and displays a message if the notification is already closed.

See Also

Plain Text E–mail with an HTML Attachment: page 2 – 42

▶ **To Reassign a Notification to Another User:**

- Use the "Forward" feature in your mail reader to forward or reassign an E–mail notification to another user. Do not use the "Reassign" button on the HTML attachment.

Attention: When you forward a notification to another user via E–mail, you are simply asking that user to respond to the notification on your behalf. Note that you still maintain ownership of the notification. If you want to transfer the notification and ownership of the notification to another user, you can only do so from the Notifications web pages. See: Viewing Notifications from a Web Browser: page 10 – 13.

Viewing Notifications from a Web Browser

You can use any Web browser that supports JavaScript and Frames to view and respond to your notifications in the Notifications Web page.

▶ **To Access 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 1.
2. If you are not using Oracle Self–Service Web Applications, you can access your worklist in one of several ways.
   - To navigate directly to your current worklist of open notifications, enter:

     `<webagent>/wfa_html.worklist[?orderkey=<orderkey>&status=<status>&user=<user>]`

     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 web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 12.

     – *<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, the worklist will be ordered by 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 with a status of OPEN.

     – *<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: Setting Global User Preferences: page 2 – 12.

- To display a worklist of notifications that match specific search criteria, go to the Find Notifications web page by entering:

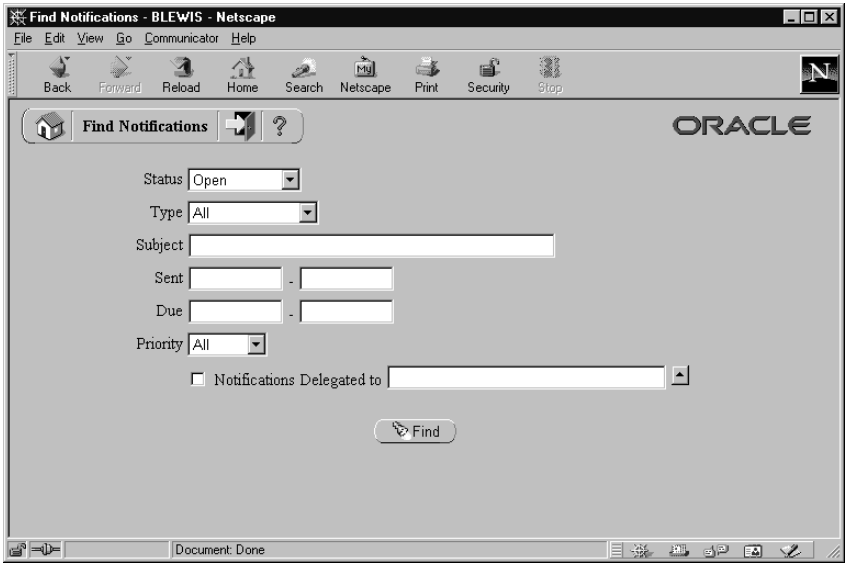
  `<webagent>/wfa_html.find`

- You can also navigate to the Notifications Worklist or Find Notifications web pages from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 9 – 2.

3. If you are not using Oracle Self–Service Web Applications and you are accessing any of the Oracle Workflow URLs for the first time in your web browser session, Oracle WebServer prompts you for a valid username and password to log on.
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 10 – 17.

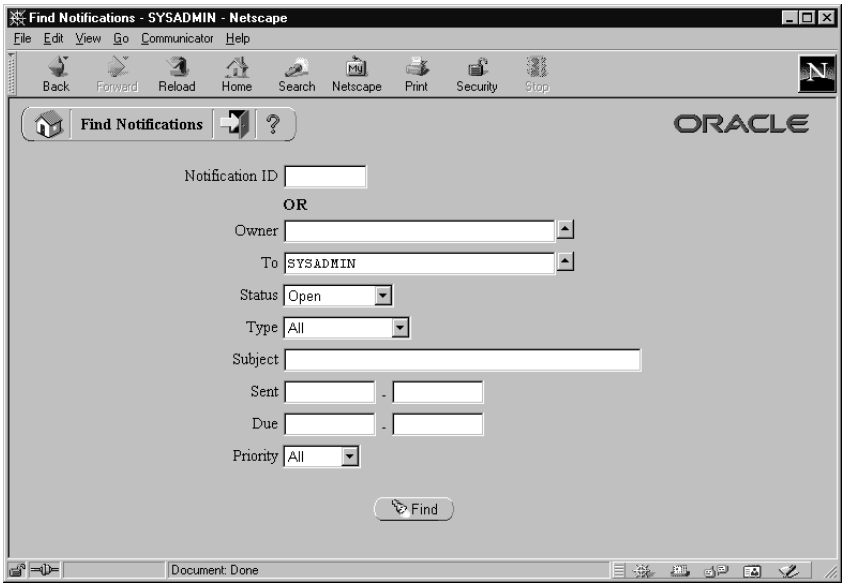
▶ To Find Notifications



1. The Find Notifications window lets you enter search criteria to locate specific notifications. If you are logged on to the current session as a regular workflow user, you can specify criteria to search for any notification(s) you own. The search criteria are:

- Status – choose a notification status of Canceled, Closed, Invalid Reply, or Open. Choose All to display notifications of any status.
- Type – choose the item type of the notification(s). Choose All to display notifications of any item type.
- Subject – enter the subject of the notification you wish to search for. This field accepts case insensitive text strings and interprets the percent sign (%) as a wildcard.
- Sent – enter the date or range of dates from which the notification(s) were sent. Use the default date format of your database.
- Due – enter the date or range of dates by which the notification(s) should be completed. Use the default date format of your database.
- Priority – choose High, Normal or Low as the priority of the notification(s) you wish to find or choose All to display notifications of any priority.
- Notifications Delegated to – check this criteria to search for notifications that you have forwarded to a specified role but yet still own. Click on the adjacent field's up arrow icon to display a list of roles from which to choose. See: Using a List of Values: page 10 – 22.

2. As a user with workflow administrator privileges, you can also search for notifications that you do not own. See: Setting Global User Preferences: page 2 – 12



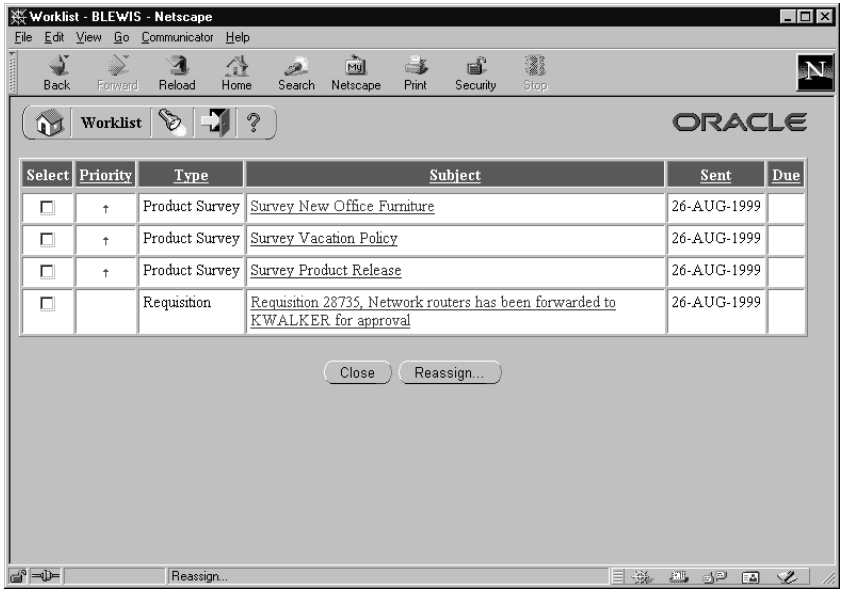
In addition to being able to specify any of the standard search criteria listed above except for "Notifications Delegated to", you also have the following criteria options:

- Notification ID—enter a specific notification ID. Note that if you specify a notification ID, all other search criteria are ignored.
- Owner—enter a role to identify all notifications owned by that role. Click on the field's up arrow icon to display a list of roles from which to choose. See: Using a List of Values: page 10 – 22.
- To—enter a role to identify all notifications sent to that role. Click on the field's up arrow icon to display a list of roles from which to choose. See: Using a List of Values: page 10 – 22.

  Note: To identify notifications where the original owner delegated the work to another role (without transferring ownership), specify different roles in the Owner and To fields. This combination of criteria is equivalent to selecting the Notifications Delegated to criterion in the standard Find Notifications screen.

3. Choose the Find button to open the Worklist window.

▶ To View Notifications from the Worklist



1. The Notifications Worklist either displays the notifications that match your search criteria if you navigated from the Find Notifications page, or lists all your open notifications if you navigated directly to this page.

   The Worklist displays the following information for each notification:

   - Priority—a high or low priority icon represents the urgency of the notification.
   - Type—the item type with which the workflow process and notification is associated.
   - Subject—a description of the notification.

     Note: If a notification is Open and requires a response, a Response Required icon appears next to its Subject link.
   - Sent—date when the notification was delivered.
   - Due—date by which the notification should be completed.

2. Click on any column heading to sort your notifications by that column in ascending order.

3. A Find icon in the toolbar lets you navigate back to the Find Notifications screen at any time so you can use search criteria to reduce the Worklist to a smaller subset of notifications.
4. The Worklist lets you simultaneously close multiple FYI–type notifications that do not require a response. Simply check Select for the each FYI–type notification you wish to close, and then choose Close.
5. You can also collectively reassign a group of notifications. Check Select for the notifications you wish to reassign, then choose Reassign... A Reassign page appears that lets you specify to whom and how you wish to reassign the notification(s). See:To Reassign a Notification to Another User: page10 – 21.
6. You can navigate to the full details of any notification and act on the notification's Subject link.

▶ To View the Details of a Notification

1. In the Detail Notification page, the full details of the notification appear in the upper frame, and the response section of the notification appears in the lower frame. You can scroll through or resize these frames.

2. The upper frame may include links embedded in the message body to additional information sources for the notification. There are two types of embedded links:

   - A reference URL link that opens another Web browser window and connects to a specified URL.
   - A DM document link that opens another Web browser window to display a document management integration window. The new browser window displays a specified document stored in a specified document management system. See:To Access a DM Document Sent by a Notification: page 10 – 31.

3. The upper frame may also include attachment icons that appear after the message body. These icons also link to additional information sources for the notification. There are four types of attachment links:

   - A reference URL link that opens another Web browser window and connects to a specified URL.
   - A PL/SQL document link that displays the contents of a document generated from a PL/SQL function.
   - A DM document link that opens another Web browser window to display a document management integration window. The new browser window displays a specified document stored in a specified document management system. See:To Access a DM Document Sent by a Notification: page 10 – 31.

   - If you are using Oracle Workflow embedded in Oracle Applications, an Oracle Applications form link that drills down to a Oracle Applications form referenced by the underlying message attribute. Depending on how the message attribute is defined, the Oracle Applications form can automatically display appropriate context information.

Attention: Attached form icons appear in a notification message only if the Worklist web page was initially launched by Oracle Applications from a menu. See: Setting the Socket Listener Activated Profile Option: page 2 – 28.

Attention: The Notification System first verifies with Oracle Applications whether the recipient's responsibility has the appropriate security to open the linked form. If the

responsibility is not allowed to open the form, the attached form icon is disabled and a message of such nature is displayed.

4. 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 last as a set of buttons to choose from as shown in the figure above. The buttons represent the possible choices to the response prompt. 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, the response section indicates that. 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 display more information about the response attribute.

     Note: A response field may have a paper clip icon next to it. Such a field requires you to specify a DM document as a response. Click on the paper clip icon to display a document management integration window where you can search for and select a DM document. See:To Respond to a Notification with a DM Document: page 10 – 34.

     Note: If you launch the Notification Worklist from Oracle Applications, your response section may display an attached form icon that lets you drill down to an Oracle Applications form to complete your response.

5. Once you submit your response, the Detail Notification page returns you to the Worklist, where the notification you just responded to now displays a status of Closed.

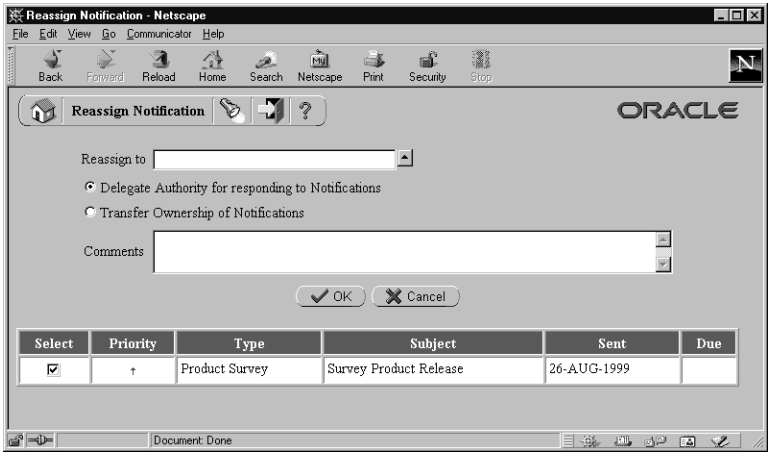
   Note: If you revisit a Closed notification, the Response section indicates that the Response has been submitted and displays the values that were submitted as the response.

6. A Find icon in the toolbar of the Detail Notification page lets you navigate back to the Find Notifications screen at any time to search for and display other notifications.

▶ To Reassign a Notification to Another User

1. You can reassign a notification in one of two ways:

   - In the Worklist page, you check Select for one or more notifications and choose Reassign....
   - In the Worklist page, click on the subject link of the notification you wish to reassign. In the Detail Notification page that appears, choose Reassign... in the Response frame of the notification.



2. In the Reassign page that appears, a summary of the selected notification(s) appear towards the bottom of the screen.

   Note: If you clicked on the Reassign button in the Response frame of a specific notification, you do not see a summary of selected notifications.

3. In the 'Reassign to' field, click on the up–arrow icon to display a window that lets you search for a list of roles to choose from. See: Using a List of Values: page 10 – 22.

4. After choosing a role, specify how you wish to reassign the notification. Select 'Delegate Authority for Responding to Notifications' if you want to give someone else authority to respond to the notification on your behalf. With this option, Oracle Workflow maintains that you own the notification. Or select 'Transfer Ownership of Notifications', if you want to give someone else complete ownership and responsibility of the notification.

5. Enter any comments you want to pass along to the new role. Choose Reassign. Once the notification is reassigned, the web browser returns you to your Worklist page, where the reassigned notification is no longer in your worklist.

Attention: Your workflow administrator may implement special logic to verify that the role that you attempt to delegate or transfer a notification to is legitimate or to restrict reassignment of notifications altogether. If so, you may get a warning message to that effect when you attempt to reassign a notification.

6. A Find icon in the toolbar of the Reassign page lets you navigate back to the Find Notifications screen at any time to search for and display other notifications.

▶ Using a List of Values

1. In the field that supports a list of values, click on the field's up–arrow icon to display a window that contains a list of values.

2. In the Find field, enter search criteria and choose Find to retrieve a subset of values that match your criteria. If you do not specify any search criteria and simply choose Find, you retrieve the complete list of values.

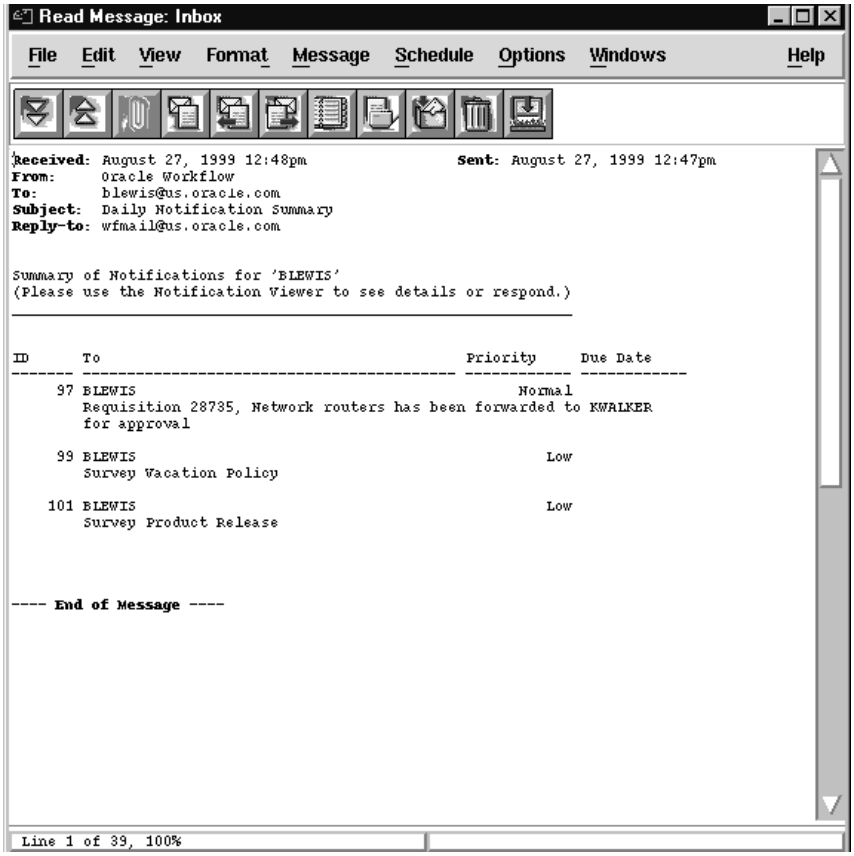
3. If the list of values contain multiple columns, you can resize the column widths by dragging the column divider to the left of right. You can also switch the order of the columns by selecting the column heading and dragging it left or right.

4. Click on a value from the list and choose OK to select that value and close the list of values window. The value you select gets populated in the original field.

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 'Plain text summary mail' in the User Preferences web page. 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: page 2 – 44.

You can receive your E–mail notification summary using any E–mail reader. The following example shows a notification summary received through Oracle Internet Messaging as the mail server and Netscape Messenger as the mail client.



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 – 56.

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.

Defining Rules for Automatic Notification Processing

Use Oracle Workflow Automatic Notification Processing 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 Automatic Notification Processing web page lets you define the rules for automatic notification processing. 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 result in one of three actions: reassigning the notification to another user, responding to or closing the notification, or simply delivering the notification to the original recipient with no further action.

Each time the Notification System sends or reassigns a notification to a role, Oracle Workflow 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, Oracle Workflow applies the rule and discontinues any further rule matching.

If a rule reassigns a notification, Oracle Workflow performs rule matching again against the new recipient role's list of rules. Oracle Workflow 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, Oracle Workflow 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 Processing

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 web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 12.
- *<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: Setting Global User Preferences: page 2 – 12.

If you have workflow administrator privileges, you can display a web page that lets you find the routing rules for a specified role. Enter:

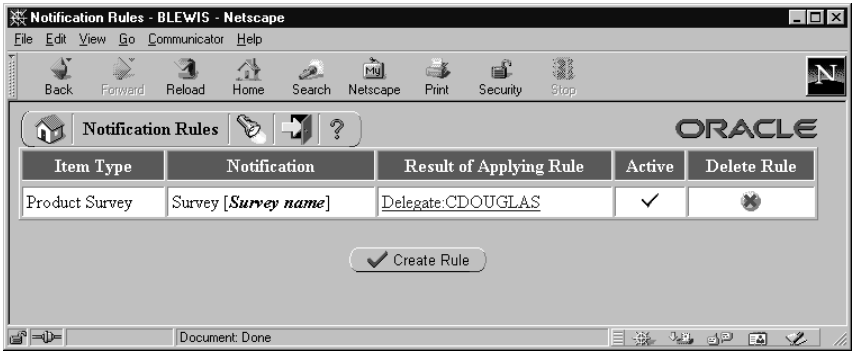
`<webagent>/wf_route.find`

Enter the user ID of a role and choose Find.

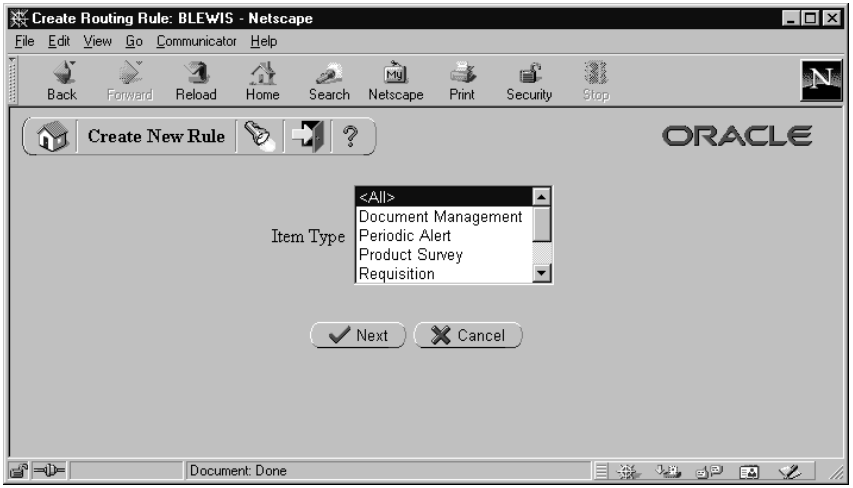
Attention: Both of these URLs access secured pages, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears.

Note: You can also access the Notification Rules web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 9 – 2.

2. The Notification Rules page for the role appears, listing all existing rules for the current role. Choose Create Rule.

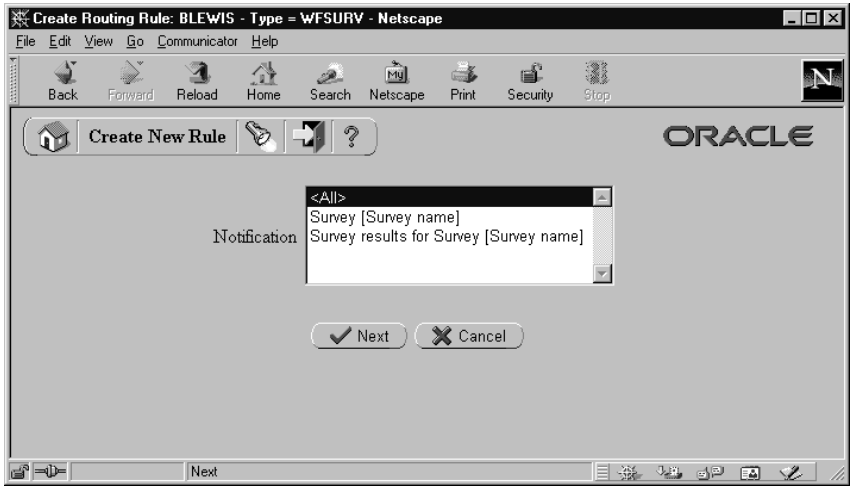


3. In the Item Type poplist field, select the item type to which this rule applies or select <All> if you want this rule to apply to notifications associated with any item type.



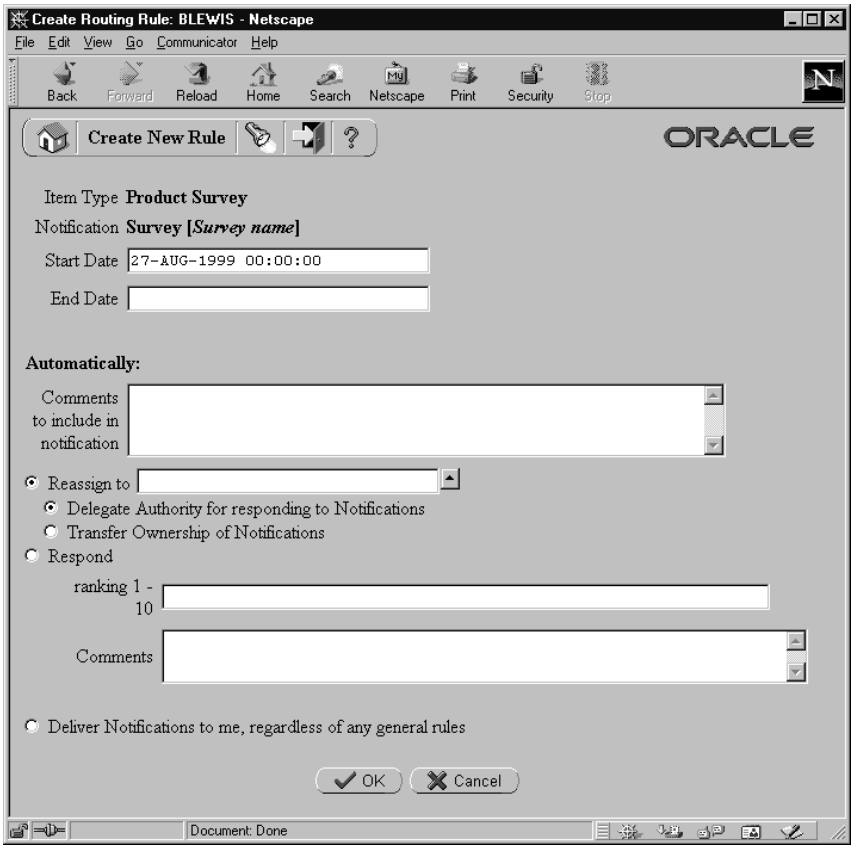
4. Choose Next to proceed or choose Cancel if you want to cancel this rule and return to the previous page.
5. If you selected <All> as the item type to apply the rule to, then skip to step 8. If you selected a specific item type, then proceed to the next step to choose a notification from that item type to which you want your rule to apply.

6. In the Notification field, select the notification message to which this rule applies or select <All> if you want this rule to apply to all notifications in the item type.



7. Choose Next to proceed or choose Cancel if you want to cancel this rule and return to the previous page.

8. The final Create New Rule page appears. The fields in this page vary depending on the item type(s) and notification(s) that you are creating this rule for. For example, if your rule pertains to all item types, you can automatically reassign all the notifications to another user, but you cannot define an automatic response to all the notifications since different notifications have different response attributes.



9. Enter values in the Start Date and End Date fields to specify the period that this rule should be active. Specify the date using the default date format of your database and specify a time using the format HH24:MI:SS if your default date format does not have a time component.

   If you leave Start Date blank, the rule is effective immediately. If you leave End Date 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 Automatic Notification Processing 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, Oracle Workflow picks one rule at random to apply.

10. In the "Comments to include in notification" 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 when the notification is reassigned or automatically responded to.
11. Choose the action that you want this rule to perform:
    - "Reassign to"—forward the notification to a designated role.
    - "Respond"—respond to the message with a set of predefined response values.
    - "Deliver Notifications to me, regardless of any general rules"—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 your 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 to 'Deliver Notifications to me,...'.
12. If your rule action is "Reassign to", click on the up–arrow icon to display a window that lets you search for a role to reassign to. See: Using a List of Values: page 10 – 22.
13. After choosing a role, specify how you wish to reassign the notifications. Select 'Delegate Authority for Responding to Notifications' if you want to give the new role authority to respond to the notification on your behalf. With this option, Oracle Workflow maintains that you still own the notifications, but the recipient role of the notifications is now the role that you are reassigning your notifications to. Select 'Transfer Ownership of Notifications', if you want to give the new role complete ownership and responsibility of the notification.

    Attention: Your workflow administrator may implement special logic to verify that the role that you attempt to delegate or transfer the notifications to is legitimate or to restrict reassignment of notifications altogether. If so, you may get a warning message to that effect when you attempt to create a reassigning rule.
14. If your rule action is "Respond", set the values that you want to automatically respond with.
15. Choose OK to save this rule and return to the Notification Rules page to display an updated list of your role's routing rules. You can also choose Cancel at any time if you want to cancel this rule and return to the previous page.

▶ To Update or Delete an Automatic Notification Processing Rule

1. Connect to the URL for the Automatic Notification Processing web page:

   `<webagent>/wf_route.list`

   `<webagent>` represents the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 12.

2. The Automatic Notification Processing page for the role appears. In the "Result of Applying Rule" column, click on the rule you wish to update.

3. In the Modify Rule page make your changes to the rule and choose OK to save your changes.

   You can also choose Cancel to undo your changes and go back to the previous page.

4. To delete a rule, in the "Delete Rule" column, choose 'X' for the rule you wish to delete.

Document Management Integration in Notifications

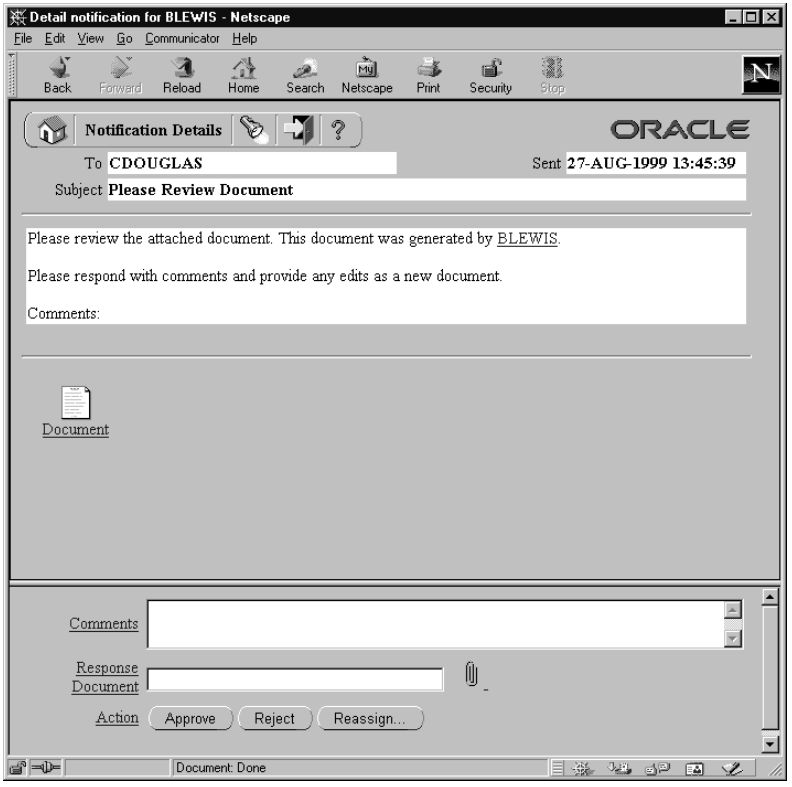
To access a referenced document management (DM) system document in a specific workflow notification, you can view the notification from the Detail Notification web page and select the document link to display the document in a DM integration screen. See: To Access a DM Document Sent by a Notification: page 10 – 31.

A notification may also require that you respond by attaching a document. In this case, select the paper clip icon that appears next to the field that requires a document response. Oracle Workflow displays a DM integration screen that allows you to search for a specific document to respond with. See: To Respond to a Notification with a DM Document: page 10 – 34.

Oracle Workflow provides you with a set of standard buttons for document management functions that are uniform across all document management systems. These buttons appear at the top of the DM integration screen. Once you select one of these buttons, Oracle Workflow presents you with your document management system's native user interface to complete the function, eliminating the need for you to learn a different UI to interact with your existing document management system. Some of the standard document management functions offered by Oracle Workflow include the ability to search for a document, check out/check in a document, and review a document's history.

▶ To Access a DM Document Sent by a Notification

1. When you view a notification from the Detail Notification web page, you may see a reference link to a DM document in the body of the message. The DM document link allows you access to the corresponding document management system to view and manipulate the document.



2. Click on the link to display a DM integration screen in a new browser window.

   Note: Since the document management system maintains secured access to its documents and functions, you must log on as a valid user the first time you access the document management system from the current web session.

3. The screen initially displays a Launch Document toolbar and the content of the referenced DM document. The buttons on the toolbar are fixed regardless of the document management system you are integrating with and represent standard functions that any document management system can perform.

   When you select a toolbar button, the integrated document management system displays a native screen that prompts you for information necessary to complete the function. You do not need to learn a new user interface to accomplish the tasks that are familiar to you in your document management system.

Note: When the document management integration screen appears, a separate Document Management Transport Window appears. Do not close this window until you have created or selected a document in the document management system. Once you create or select a document, the Document Management Transport Window automatically closes.

4. Click on any of the following Oracle Workflow DM toolbar buttons. The standard functions associated with the buttons are listed below. Each function is effectively carried out by the integrated DM system and although the functions are described in a general manner below, realize that the implementation of these functions may vary between DM vendors and that different vendors take different approaches to the user interface.

   - Document Name—displays the name of the referenced document.
   - Display—display the content of the document in its native format. In some exceptions, the document must be downloaded and viewed from a locally installed application. The document management system informs you if so.
   - Fetch—fetch the contents of the document to your local file system. This allows you to review and modify the document content without being connected to Oracle Workflow or the document management system.
   - Check Out—check out the document from the document management system and fetch the document onto your local file system. If you check out the latest revision of the document, you lock the document assuming it is not locked by someone else.

     If the document is already locked, or if you check out an earlier revision of the document, this function checks out the document in read–only mode and fetches a copy to your local file system. You can edit the local copy, but you cannot check it in unless you rename the document.
   - Check In—check in a new version of a document into the document management system. Provide detailed comments that explain your changes or a description of the document if this is a new document. The document is unlocked once it is checked in so that others may revise it.

- Unlock—remove a lock on the document without checking in a new version of the document. Use this function if you check out the wrong document or if you do not plan to edit the document you've checked out.
- Show History—display the history of and information about the referenced document. Depending on how the integrated document management system implements this function, you may see information such as content, comments, archive history, and metadata (author, date changes, and so on) associated with the check in history of the document. You may also see information such as who is currently locking the document, its current revision number, its current document size and format, and so on.
- Help—display online help.

▶ **To Respond to a Notification with a DM Document**

1. When you view notifications from the Detail Notification web page, certain response fields may display adjacent paper clip icons. These fields require you to attach a DM document as part of your notification response. The paper clip icon provides you direct access to an integrated document management system so you can attach a new or existing DM document to your notification response.
2. Click on the paper clip icon to display a DM integration screen in a new browser window.
3. The screen initially displays a Launch Document toolbar and your default document management system's native search screen. The buttons on the toolbar are fixed regardless of the document management system you are integrating with and represent standard functions that any document management system can perform.

Note: Your default document management system is defined in the User Preferences web page. See: Setting User Preferences: page 9 – 4.

When you select a toolbar button, the integrated document management system displays a native screen that prompts you for information necessary to complete the function. You do not need

to learn a new user interface to accomplish the tasks that are familiar to you in your document management system.

Note: When the document management integration screen appears, a separate Document Management Transport Window appears. Do not close this window until you have created or selected a document in the document management system. Once you create or select a document, the Document Management Transport Window automatically closes.

4. Click on any of the following Launch Document toolbar buttons. The standard functions associated with the buttons are listed below. Each function is effectively carried out by the integrated DM system and although the functions are described in a general manner below, realize that the implementation of these functions may vary between DM vendors and that different vendors take different approaches to the user interface.

   - Change Document Home—displays the Change Document Home web page so you can choose a different document management node from which to select your document. Your organization can integrate multiple document management installations with Oracle Workflow. Each installation is called a node.
   - Node Name—displays the name of the current DM node.
   - Search—search across all authorized documents and optionally across multiple document versions for a document matching the content or metadata criteria that you specify. From the list of matching documents and associated versions, you can review the content of a document and choose to perform an action like referencing the document in the notification response. The richness of the search function is solely dependent on the underlying document management system.
   - Create New—create a new document in the current document management system for a file stored in your local file system. You must define the document, specify the folder where you would like store the document, and specify the access rules for the document.
   - Browse—browse for a document by navigating through the document management system folder hierarchy.
   - Help—display online help.

CHAPTER

11

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 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

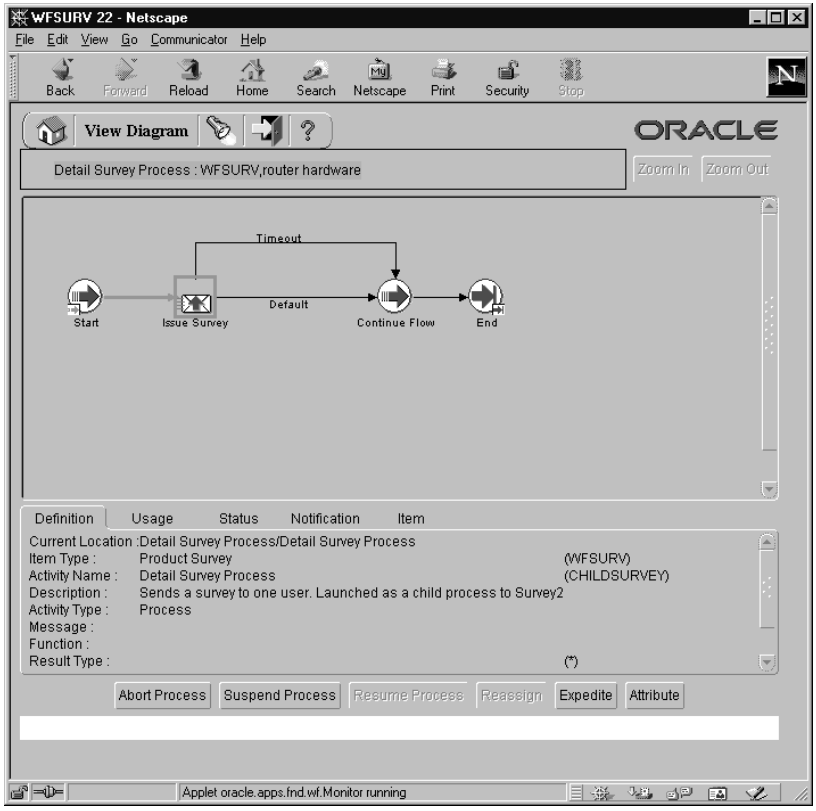
Oracle Workflow Views: page 8 – 108

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 11 – 7.

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 key 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 any empty space in the process diagram window 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 11 – 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

```
Current Location: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, Notice, or Function
Message:         Message internal name
Function:        Name of PL/SQL procedure called by activity
Result Type:     Result type display name   (internal name)
*Cost:           Function activity cost in seconds
*On Revisit:     IGNORE, LOOP, or RESET
*Error Process:  Error Item Type and Error Process internal
name assigned to activity, if any
```

Usage Tab

```
Current Location: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
Timeout:         N minutes or item attribute internal name
```

Status Tab

```
Current Location:Activity Display Name
Status:          Activity status
Result:          Activity result            (result code)
Begin Date:      Date activity begins
End Date:        Date activity ends
Due Date:        Date activity is due to timeout
*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

```
Current Location: 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
Due Date:        Date activity is due to timeout

(If the selected activity is a notification that requires a
response, then instead of displaying the above information,
this tab displays the message response attributes as
<message_attribute> <type (Format)> <value>.  If the
selected activity is a polling-type notification activity,
where Expand Roles is checked and a response is required,
then this tab displays the message response attributes as
described above for each recipient.  If the selected
activity is a notification activity, where Expand Roles is
on, but no response is required, then the recipient shown is
simply the role, rather than the individual users in the
role.)

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 8 – 28.
- 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 8 – 26.

- 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 8 – 27.
- 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 8 – 52.
- 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 8 – 53.
- 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.GetDiagramURL( )*. 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 8 – 102.

> **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 WebDB, Oracle Application Server, or Oracle Web Application Server 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: TRANSLATE: page 8 – 67.

> **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 9 – 2

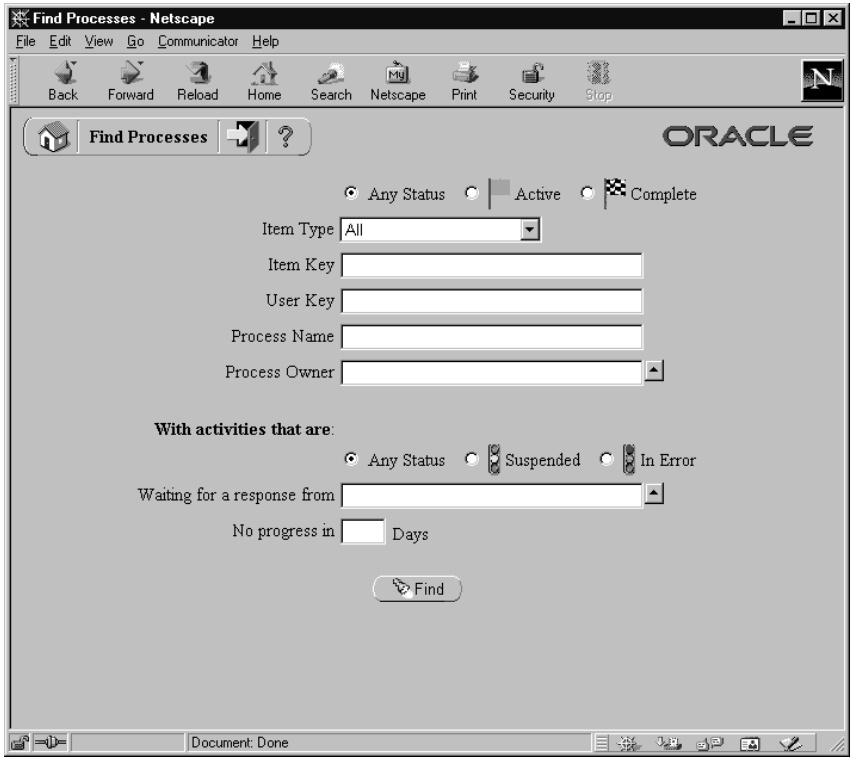
Using the Find Processes Web Page

The Oracle Workflow Find Processes web page allows you to query for a list of workflow process instances that match certain search criteria. From the Process List that appears, you can select a single process instance to review in more detail.

You can view details about the completed notification activities in the Notifications List web page that appears, or choose Advanced Options to display details about other activities in the Activities List web page.

You can also navigate to the from the Notifications List to administer the process or see a graphical representation of the process and its status.

▶ 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 8 – 18.
   - Process Name—Specify the display name of a process activity.

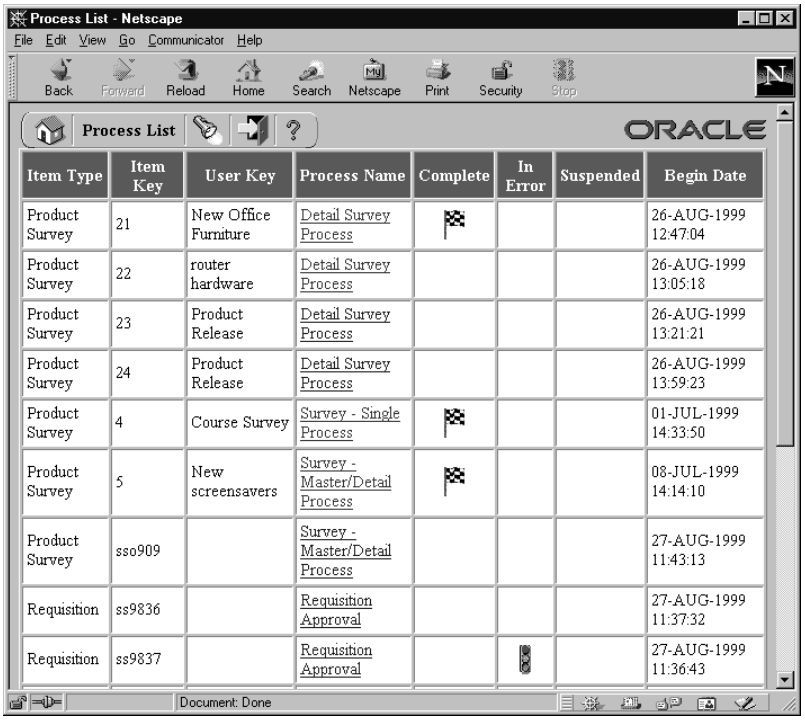
2. If you log on as a user with Workflow Administrator privileges, you can search for and display any process instance, even if you do not own the process. In the Process Owner field, enter the internal name of any role defined in WF_ROLES to list only processes owned by that role. Alternatively, leave 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 display all matching process instances in the Process List web page.

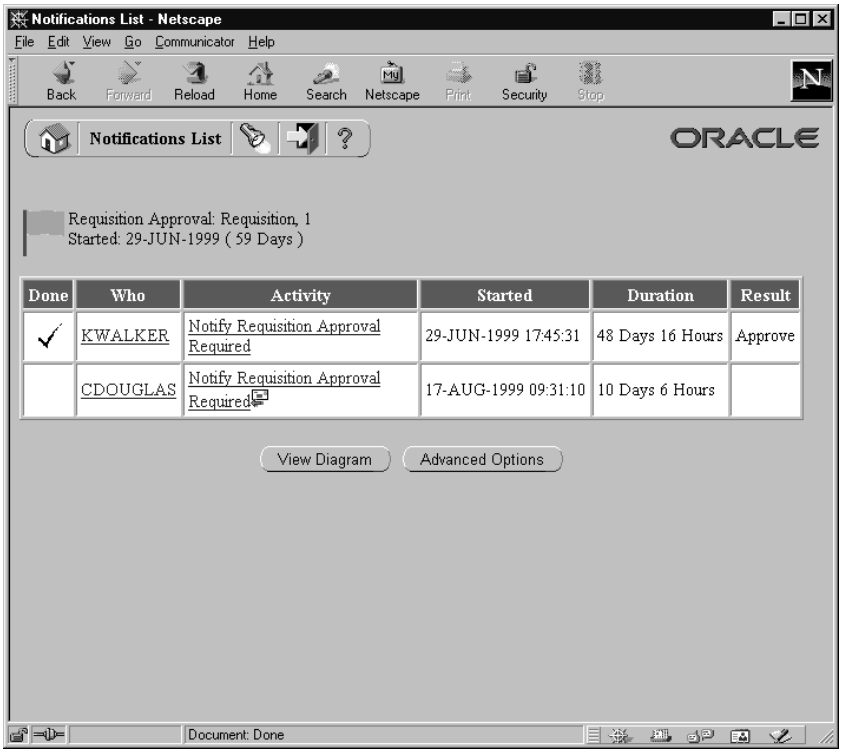
▶ To Review the Process List



Item Type	Item Key	User Key	Process Name	Complete	In Error	Suspended	Begin Date
Product Survey	21	New Office Furniture	Detail Survey Process				26-AUG-1999 12:47:04
Product Survey	22	router hardware	Detail Survey Process				26-AUG-1999 13:05:18
Product Survey	23	Product Release	Detail Survey Process				26-AUG-1999 13:21:21
Product Survey	24	Product Release	Detail Survey Process				26-AUG-1999 13:59:23
Product Survey	4	Course Survey	Survey - Single Process				01-JUL-1999 14:33:50
Product Survey	5	New screensavers	Survey - Master/Detail Process				08-JUL-1999 14:14:10
Product Survey	sso909		Survey - Master/Detail Process				27-AUG-1999 11:43:13
Requisition	ss9836		Requisition Approval				27-AUG-1999 11:37:32
Requisition	ss9837		Requisition Approval				27-AUG-1999 11:36:43

1. The Process List provides a summary of all workflow process instances that match your search criteria as specified in the Find Processes web page.
2. The process instances are listed in ascending order first by item type, then by item key.
3. The Process List summarizes the status of each process instance, as indicated by the Complete, In Error, and Suspended columns.
4. Choose a process link in the Process Name column to display the Notifications List for that process instance.

▶ To Review the Notifications List



1. The Notifications List shows for the selected process instance, all the current notifications that have been sent that require a special Result response. In other words, these are the notification activities that allow the process to branch based on the recipient's response.
2. The Notification List summarizes 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, as well as what its result is.

   Note: If the process itself is in an error state, and the cause of the error was from a notification, the result of that notification may appear as a link in the Result column. Choose that link to display the cause of the error.
3. Choose a user link in the Who column if you want to send email to the user that a notification has been assigned to.

Note: You can display a helpful hint about any link on the Notifications List web page by placing your cursor over the link. The hint appears in your web browser's status bar.

4. Choose a notification activity link in the Activity column if you want to view the full definition of a notification activity.

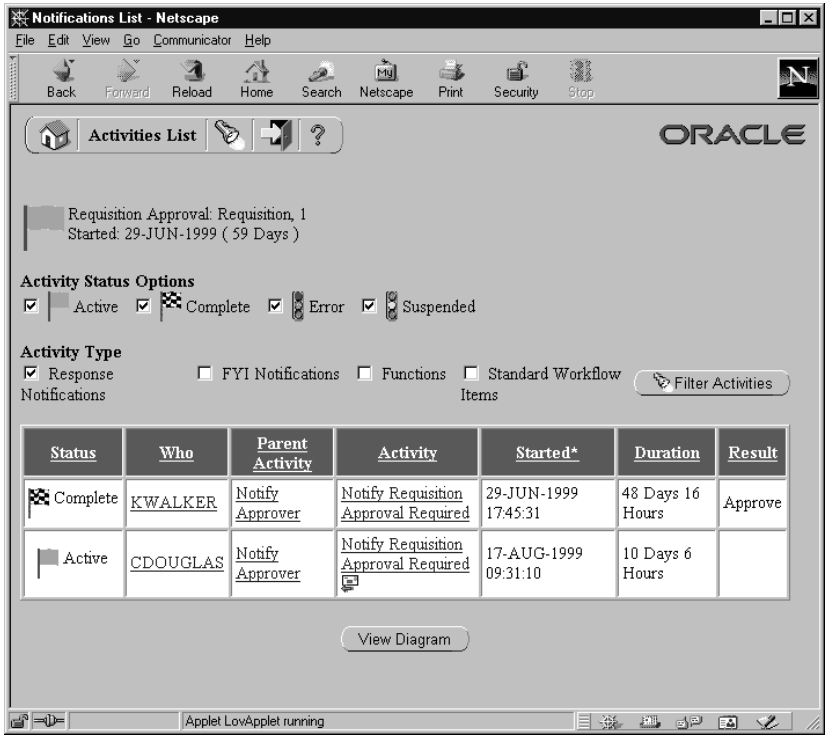
5. If a notification activity is still open and requires a response, and you are logged on with workflow administrator privileges, an icon will appear after the notification activity name in the Activity column. You can click on this icon to go to the Notification Details page where you can directly respond to this notification. When complete your response, choose the browser Back button to return to the Notifications List.

6. Choose Advanced Options to go to the Activities List web page where you can specify advanced criteria to search and display specific activities of interest for the process. See: To Filter Activities in the Activities List: page 11 – 14.

7. You can also choose the View Diagram button to display the selected process instance in the Workflow Monitor for a graphical representation of the process status. If you connected to the current web 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 11 – 2.

Attention: If the process you select is a member of a parent/child process, a parent/child hierarchy list appears on the left hand side. The hierarchy list show links to corresponding parent and child instances of the current process instance. The links invoke the Notifications List on the selected parent or child instance.

▶ To Filter Activities in the Activities List



1. The Activities List web page lets you specify various criteria to filter for specific activities of interest.
2. Use the Activity Status Options check boxes to specify any activity status of interest. A status of Active also includes activities that are in the Notified, Deferred and Waiting state.
3. Use the Activity Type check boxes to specify the types of activities you want to view. You can choose to display notification activities that require a response, notification activities that do not require a response, process or function activities, and/or activities that belong to the Standard item type.
4. Once you finish selecting your criteria, choose Filter Activities to display the activities that match your criteria.
5. The resulting activities summary list includes the following columns of information:
   - Status—the status of the activity, which is either Active, Complete, Error or Suspend.

- Who—the performer of the activity. If the activity is a function activity, the Workflow Engine is the performer. If the performer is a person, you can click on the link to that person's name to send mail to that individual. Note that if an activity is a notification activity that has Expand Roles on, multiple rows of that same activity appear in the summary, with the individual members of the role listed in the Who column.
- Parent Activity—the process activity that this activity belongs to, unless the activity itself is the top level process. The parent activity provides a link to the details of its definition.
- Activity—the name of the activity. This activity provides a link to the details of its definition.
- Started—the date and time when the activity was initiated.
- Duration—the amount of time taken to complete the activity, shown as one unit lower than the most significant unit of time taken. If the activity took only seconds to complete, then only seconds are shown.
- Result—the result of the activity. If the activity has a status of Error, then the result provides a link to the error name, error message, and error stack associated with the error.

  Note: You can display a helpful hint about any link on the Activities List web page by placing your cursor over the link. The hint appears in your web browser's status bar.

6. You can sort the activities summary list based on any column by clicking on a column header. An asterisk (*) appears next to the column title to indicate that it is being used for sorting. If the asterisk is to the left of the column title, the sort order is ascending. If the asterisk is to the right of the column title, the sort order is descending. Clicking multiple times on the same column title reverses the sort order.
7. You can also choose the View Diagram button to display the process instance in the Workflow Monitor for a graphical representation of the process status. If you connected to the current web 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 11 – 2.

See Also

Setting Up an Oracle Workflow Directory Service: page 2 – 17

Setting Global User Preferences: page 2 – 12

CHAPTER

12

Testing a Workflow Definition

This chapter tells you how to test your workflow definitions using the Oracle Workflow Launch Processes web page.

Testing Workflow Definitions

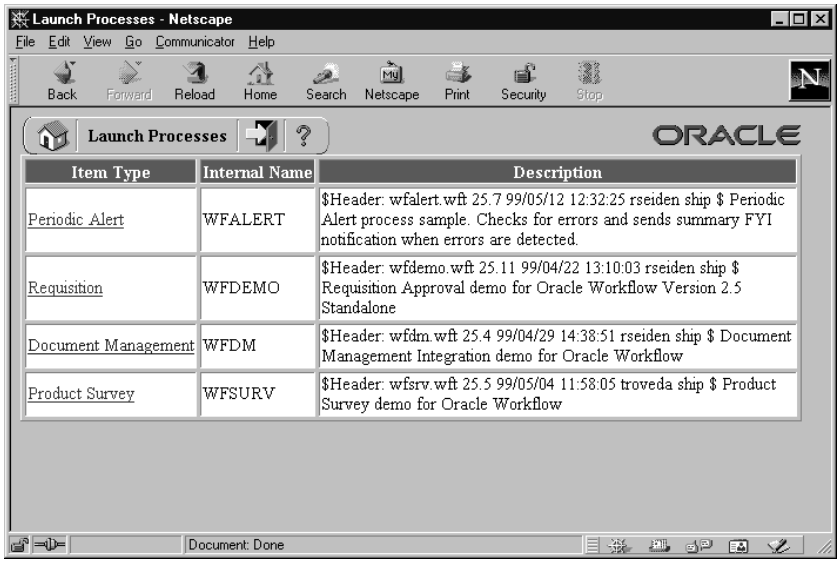
Oracle Workflow provides a web–based interface called Launch Processes for you to test any workflow definition you define and save to the database. Launch Processes is accessible only to users belonging to the Workflow Administrator role.

Although you can run the Launch Processes web page against any Oracle Workflow database, we advise that you create a separate environment for testing purposes. To test a workflow definition, you should set up the following in your test environment:

- Define test users/roles. You can test against the users and roles predefined in the Oracle Workflow demonstration data model. See: Installing the Requisition Data Model: page 13 – 5.
- If you are using the standalone version of Oracle Workflow and you plan to use the Notifications web page to view notifications, you need to define your test users/roles in your web security system. Refer to your Oracle Application Server documentation for more information.
- If you plan to use E–mail to view notifications, you can send all notifications to a single test E–mail address by setting the TEST_ADDRESS parameter in the Notification Mailer configuration file. See: To create a configuration file for the Notification Mailer: page 2 – 47.

▶ **To Test a Workflow Definition:**

1. Use a web browser to connect to the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 9 – 2.
2. Select the Launch Processes link to display the Launch Processes web page.



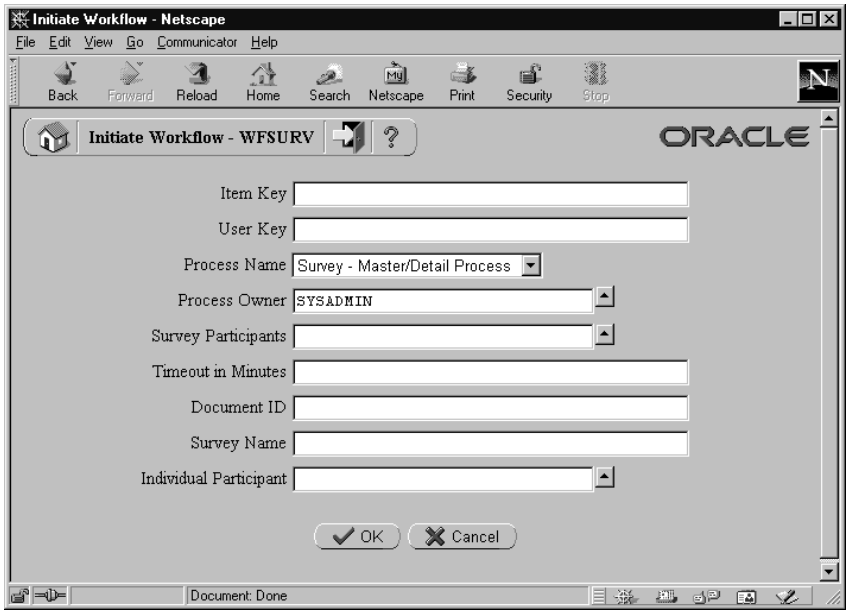
Attention: Note that you can also connect to this page directly using the secured URL:

`<webagent>/wf_initiate.ItemType`

`<webagent>` represents the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 12.

Attention: This is a secured page, so if you have not yet logged on as a valid workflow administrator in the current web session, you will be prompted to do so before the page appears.

3. The Launch Processes page displays all the item type definitions stored in the database except the Oracle Workflow seeded item types: Wferror, Wfmail, and Wfstd. The internal name and description for each item type also appears. Select the item type that owns the workflow process definition you wish to test.



4. Use the Initiate Workflow web page to specify the details for the process you wish to launch. To initiate an instance of a workflow process, you need to specify:

   - A unique item key for the process instance.
   - A user–defined key that you want to use to identify the process.
   - The name of the process to test.
   - An optional process owner.
   - Values for any item type attributes associated with the item type of the process.

   Select OK. To initiate the workflow process, the Initiate Workflow web page calls the Workflow Engine *CreateProcess* and *Startprocess* APIs for the item type and item key specified. It also calls the Workflow Engine *SetItemOwner* and *SetItemAttr* APIs to set the process owner and all the item type attributes to the values specified.

5. The Workflow Monitor Activities List for your initiated process instance appears. The Activities List displays the status of the activities that have been executed. You can also select the View Diagram button to display the status of the process graphically in the Workflow Monitor. See: To Filter Activities in the Activities List: page 11 – 14.

6. If the process you are testing contains notifications, you can navigate back to the Workflow Home page and select the Find Notifications link to find the outstanding Notifications that require responses to complete the process. Alternatively, if you prefer to test the notification responses via E–mail, you can connect to the E–mail test account you specified in the Notification Mailer to respond to the outstanding notifications for your process.

CHAPTER

13

Demonstration Workflow Processes

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

Sample Workflow Processes

The following sample workflow processes are included with Oracle Workflow. Each of these processes illustrate the integration of different Oracle Workflow features. You can use any of these processes to verify your installation of Oracle Workflow.

- Requisition Process—illustrates results–based branching, parallel branching, subprocesses, timeouts, looping, and integration of PL/SQL documents in a notification. See: Requisition Process: page 13 – 4.
- Product Survey Process—illustrates the implementation of a voting activity, integration of PL/SQL documents in a notification, and the coordination of master/detail processes. See: Product Survey Process: page 13 – 33.
- Document Review Process—illustrates document management integration and looping. See: Document Process: page 13 – 48.
- Error Check Process—illustrates how to simulate Oracle Alert's periodic alert functionality in a workflow process and how to use the Standard Wait activity. See: Error Check Process: page 13 – 53.

Attention: If you are using the standalone version of Oracle Workflow, Oracle Installer gives you the option to install all or none of the sample workflow processes listed above.

If you are using the version of Oracle Workflow embedded in Oracle Applications, AutoInstall installs only the Document Review and Error Check processes. These two sample workflow processes do not require the creation of supporting data models.

You can initiate these sample workflows from the Workflow Demonstrations home page or the Launch Processes web page. You can access the Workflow Demonstrations home page using the URL:

`<webagent>/wf_demo.home`

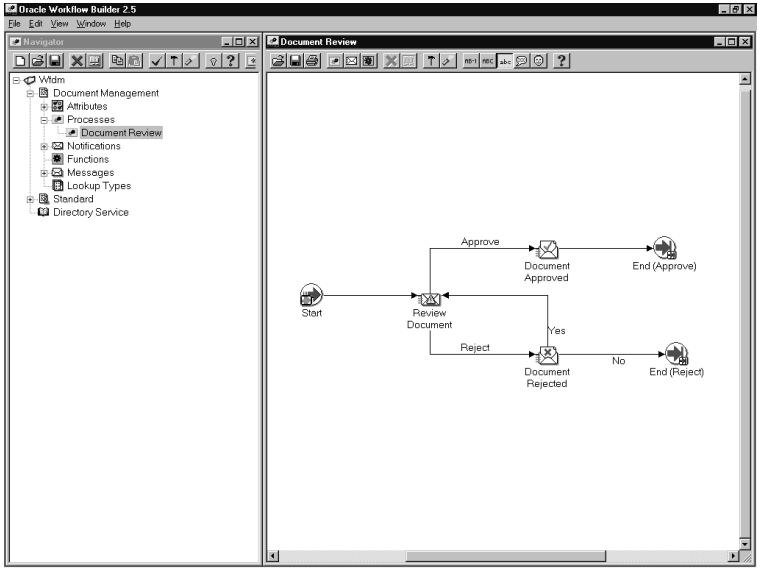
The Workflow Demonstrations home page displays your notifications Worklist in the right–hand frame and in the left–hand frame lists links that let you initiate each of the sample workflows from a different web page.

See Also

Testing Workflow Definitions: page 12 – 2

Displaying the Process Diagram of a Sample Workflow

You can view the process diagram of a sample process in the Process window of Oracle Workflow Builder.



▶ To Display a Sample Process in Oracle Workflow Builder

1. Choose Open from the File menu, and connect to your Oracle Workflow database.

   Alternatively, you can connect to any of the sample workflow definitions files, located in the Oracle Workflow *wf/data/<language>* subdirectory on your PC.

2. Expand the data source, then the item type branch within that data source.

3. Expand the Processes branch then double–click on a process activity to display the process diagram in a Process window.

Requisition Process

The Requisition process is an example of a workflow process that gets initiated when you create a new requisition to purchase an item. The Requisition process is based on two tables that store approval hierarchy and spending authority information.

When you submit a requisition in this demonstration, the process sends a notification to the next manager in the approval hierarchy to approve the requisition. If the spending limit of the approving manager is less than the requisition amount, the process forwards the requisition to the next higher manager in the approval hierarchy until it finds a manager with the appropriate spending limit to approve the requisition. Each intermediate manager must approve the requisition to move it to the next higher manager. Once a manager with the appropriate spending limit approves the requisition, the process ends with a result of Approve.

The process can end with a result of rejected if:

- Any manager rejects the requisition.
- The requisition amount is greater than the highest spending limit.
- The requisition's requestor does not have a manager.

You can set up and initiate this example process if you are using the standalone version of Oracle Workflow. If you are using Oracle Workflow embedded in Oracle Applications, you should consider this process mainly as an example for explanation purposes and not for demonstration use. The files necessary to 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 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 process based on a fictitious requisition: run a script or submit a requisition using a web–based interface. Both methods require that you provide the name of the employee who prepared the requisition, the requisition amount, the requisition number, a requisition description, a requisition process owner and the name of the workflow process to initiate.

This section describes the Requisition process in detail to give you an understanding of what each activity in this workflow accomplishes.

Installing the Requisition Data Model

The Requisition data model is available for installation only for the standalone version of Oracle Workflow. The data model is installed using Oracle Installer. Oracle Installer asks if you wish to install the sample workflows and their corresponding data models when you install the Oracle Workflow server. The files used in the installation are copied to the *demo* and *demo/<language>* subdirectories of your Oracle Workflow server directory structure.

Attention: For the Requisition process demonstration to work properly, you should perform the steps required to set up Oracle Workflow after you install the Requisition data model. See: Overview of Setting Up: page 2 – 4.

The installation does the following:

- Calls the script *wfdemou.sql* to create a database account for each of the users listed in the seed data table shown below. The script creates public grants and synonyms so that these accounts have full access to Oracle Workflow's web–based user interface.
- 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: BLEWIS, KWALKER, CDOUGLAS, and SPIERSON.

– WF_REQDEMO_EMP_AUTHORITY—maintains the spending limit for each employee. The limit for each employee follows the employee's userid: BLEWIS:500, KWALKER:1000, CDOUGLAS:2000, and SPIERSON:3000.

- 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			
BLEWIS			x	
KWALKER			x	
CDOUGLAS			x	x
SPIERSON		x		x

Table 13 – 1 (Page 1 of 1)

Attention: Each user has an E–mail address of 'WFINVALID' and each role has an E–mail address identical to its role name. You can change the users' and roles' E–mail addresses to other values by calling the Directory Service APIs *SetAdHocUserAttr* or *SetAdHocRoleAttr*. Alternatively, if you want E–mail notifications for all the users and roles to go to a single E–mail inbox, you can specify a test E–mail address in the configuration file of the Notification Mailer. See: To create a configuration file for the Notification Mailer: page 2 – 47.

Attention: Also all users except BLEWIS have a Notification Preference of 'MAILHTML', which allows them, in addition to viewing notifications from the Notifications Web page, to get individual notifications via E–mail. BLEWIS has a Notification Preference of 'SUMMARY', which allows him, in addition to viewing notifications from the Notifications Web page, to receive a periodic E–mail summarizing all his currently open notifications. Note that a Notification Mailer must be set up to deliver E–mail notifications.

Attention: Your Oracle Workflow directory service views must map to the WF_LOCAL_USERS, WF_LOCAL_ROLES

and WF_LOCAL_USER_ROLES tables to include the users and roles of the Requisition data model. See: Setting Up an Oracle Workflow Directory Service: page 2 – 17.

- Calls the scripts *wfdemos.sql* and *wfdemob.sql* to create the PL/SQL spec and body for packages called WF_REQDEMO and WF_DEMO. This package contains:
  - The PL/SQL stored procedures associated with the demonstration home page.
  - The PL/SQL stored procedures called by the function activities used in the Requisition Process workflow.
  - The PL/SQL procedure WF_REQDEMO.Create_Req called by the Oracle Workflow web agent to generate the web–based interface page for the Requisition process demonstration.
- Runs the Workflow Resource Generator to load messages from *wfdemo.msg* into the database. The messages are used by the web–based interface page for the Requisition process demonstration.
- Loads the Requisition Process workflow definition from *wfdemo.wft* into the database. You can view this process in Oracle Workflow Builder.

Initiating the Requisition Workflow

You can use any of the following methods to initiate the Requisition workflow:

- Run the script *wfrund.sql.*
- Access the Requisition Demonstration web page from the Workflow Demonstrations home page.
- Use the Launch Processes web page. See: Testing Workflow Definitions: page 12 – 2.

You can also create your own custom end–user application interface to let users create requisitions that automatically initiate the Requisition process workflow. You must, however, customize the application interface such that when a user saves the requisition to the application database, the application calls a PL/SQL stored procedure similar to *WF_REQDEMO.StartProcess* that initiates the Requisition process. See: Sample StartProcess Function: page 13 – 22.

▶ **To Run *wfrund.sql***

1. Enter the following command to run the script *wfrund.sql* in SQL*PLUS:

   ```
   sqlplus <username>/<password>@<alias> @wfrund.sql
   <req_num> <req_desc> <req_amount> <requestor>
   <req_process_owner> <process_int_name> <item_type>
   ```

   Replace <*username*>/<*password*>@<*alias*> with the username, password, and alias for the database account where you installed the demonstration data model.

   Replace <*req_num*> with the requisition number that uniquely identifies the requisition.

   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 10 – 2 and Viewing Notifications from a Web Browser: page 10 – 13.

   You can also access the Workflow Monitor to view the status of the workflow process. See: Using the Find Processes Web Page: page 11 – 8.

▶ **To Use the Requisition Demonstration Web Page**

1. Enter the following URL in a web browser to access the Workflow Demonstration web page, then click on the Requisition Approval link to display the Requisition Approval web page:

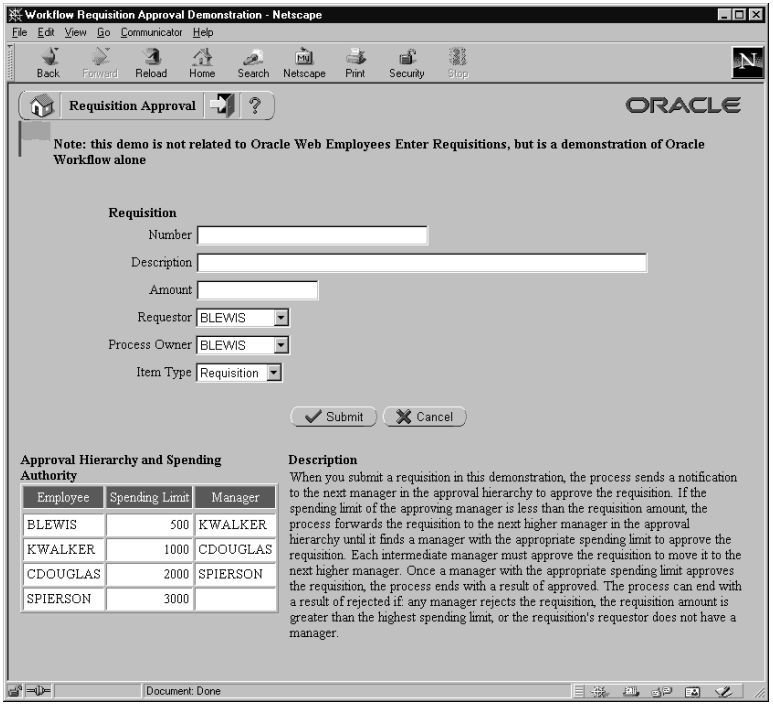
   ```
   <webagent>/wf_demo.home
   ```

<webagent> represents the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 12.

Alternatively, you can enter the following URL to directly display the Requisition Approval web page:

<webagent>/wf_reqdemo.create_req

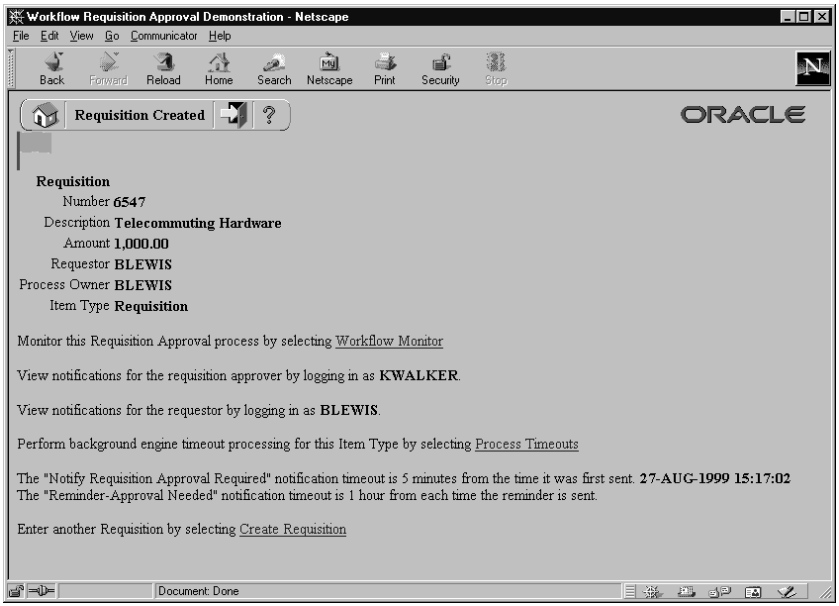
Attention: These are both secured pages, so if you have not yet logged on as a valid workflow user in the current web session, you will be prompted to do so before the page appears.



2. Enter a unique requisition number.
3. Specify a unique requisition description of 80 characters or less.
4. Enter a requisition amount. The amount should be a number without formatting.
5. Use the poplist fields to specify a requisition requestor and process owner. The names on these poplists are limited to the names of the roles in the demonstration data model.

6. Following the requisition input fields is the Approval Hierarchy and Spending Authority table and a description of how the Requisition demonstration process works. The Approval Hierarchy and Spending Authority table summarizes the contents of the demonstration data model.

7. Choose submit to initiate the Requisition process and to navigate to the Requisition Created confirmation page.



8. In addition to telling you what roles you should log in as to view the process' notifications, the confirmation page also contains a HTML link to the Workflow Monitor where you can choose View Diagram to display the process diagram for the requisition you submitted in ADMIN mode. See: Workflow Monitor: page 11 – 2.

Attention: If you are using the Oracle Application Server or Oracle Web Application Server as your web server, then to log on 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 Demonstration web page.

The Requisition Item Type

The Requisition process is associated with an item type called Requisition. Currently there are two workflow processes associated with Requisition: Requisition Approval and Notify Approver.

If you examine the property page of Requisition, you see that it has a persistence type of Temporary and persistence number of days of 0. This means that the run time data associated with any work items for this item type are eligible for purging as soon as they complete. You also see that it calls a selector function named *WF_REQDEMO.SELECTOR*. This selector function is an example PL/SQL stored procedure that returns the name of the process to run when more than one process exists for a given item type. The selector function in this example returns REQUISITION_APPROVAL or 'Requisition Approval' as the process to run.

The Requisition item type also has several attributes associated with it. These attributes reference information in the demonstration application tables. The attributes are used and maintained by function activities as well as notification activities throughout the process.

Requisition Item Type Attributes

Display Name	Description	Type	Length/Format/ Lookup Type
Forward From Username	Username of the person that the requisition is forwarded from	Role	
Forward To Username	Username of the person that the requisition is forwarded to	Role	
Requestor Username	Username of the requisition preparer	Role	
Requisition Amount	Requisition amount	Number	9,999,999,999.99
Requisition Number	Unique identifier of a requisition	Text	
Requisition Description	Unique user identifier of a requisition	Text	30
Requisition Process Owner	Username of the requisition owner	Role	
Note	Note	Text	
Monitor URL	Monitor URL	URL	
Requisition Document	Requisition Document is generated by PL/SQL	Document	
Reminder Requisition Document	Reminder Requisition Document is generated by PL/SQL	Document	

Table 13 – 2 (Page 1 of 1)

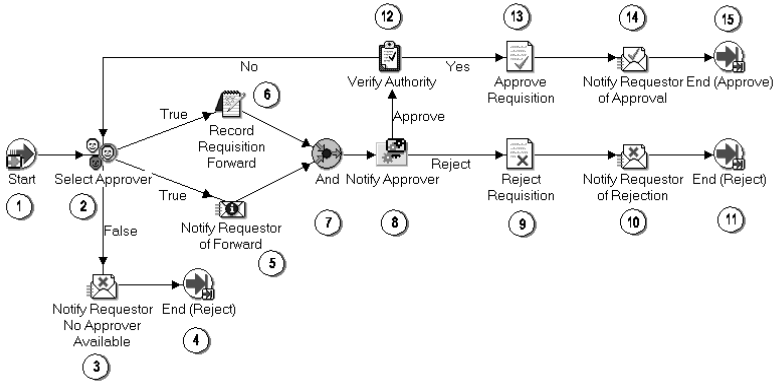
Summary of the Requisition Approval Process

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 process has a result type of Approval, indicating that when the process completes, it has a result of Approve or Reject (the lookup codes in the Approval lookup type associated with the Standard item type). This process activity is also runnable, indicating that it can be initiated as a top level process to run by

making calls to the Workflow Engine *CreateProcess* and *StartProcess* APIs.

The Details property page of the process activity indicates that the Requisition process has an error process assigned to it that is initiated only when an error is encountered in the process. The error process is associated with an item type called WFERROR and is called DEFAULT_ERROR. For example, if you attempt to initiate the Requisition Approval process with a requisition that is created by someone who is not listed in the employee approval hierarchy, the Workflow Engine would raise an error when it tries to execute the Select Approver activity. This error would initiate WFERROR/DEFAULT_ERROR, which is the Default Error Process. See: Default Error Process: page 6 – 20.

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 workflow begins when you run a script called *wfrund.sql* or submit a requisition using the Requisition Demonstration web page. In both cases, you must provide a requisition requestor, requisition number, requisition amount, requisition description, and process owner. See: Initiating the Requisition Workflow: page 13 – 7.

The workflow begins at node 1 with the Start activity.

At node 2, the process attempts to select an approver for the requisition. If an approver cannot be found for the requisition, the requestor is notified and the process ends with the final process result

of Reject. If an approver is found, then the requestor is notified of who that approver is and a function records in the application that the requisition is being forwarded to the approver. Both of these activities must complete before the approver is actually notified in node 8.

Node 8 is a subprocess that requests the approver to approve the requisition by a specified period of time and if the approver does not respond by that time, the subprocess performs a timeout activity to keep sending a reminder to the approver until the approver responds. If the approver rejects the requisition, the requisition gets updated as rejected in node 9, and the requestor is notified in node 10. The process ends at this point with a result of Reject.

If the approver approves the requisition, the process transitions to node 12 to verify that the requisition amount is within the approver's spending limit. If it is, the process approves the requisition in node 13, and notifies the requestor in node 14. The process ends in this case with a result of Approve.

Requisition Process Activities

Following is a description of each activity listed by the activity's display name. You can create all the components for an activity in the graphical Oracle Workflow Builder except for the PL/SQL stored procedures that the function activities call. Function activities can execute functions external to the database by integration with Oracle8 Advanced Queues or execute PL/SQL stored procedures which you must create and store in the Oracle RDBMS. All the function activities in the Requisition process execute PL/SQL stored procedures. The naming convention for the PL/SQL stored procedures used in the Requisition process is:

`WF_REQDEMO.<`*`PROCEDURE`*`>`

`WF_REQDEMO` is the name of the package that groups all the procedures used by the Requisition process. *<PROCEDURE>* represents the name of the procedure.

Several activities are described in greater depth to give you an idea of how they are constructed. See: Example Function Activities: page 13 – 25 and Example Notification Activities: page 13 – 30.

Start (Node 1)

This is a Standard function activity that simply marks the start of the process.

Function *WF_STANDARD.NOOP*

Result Type None

Prerequisite Activities None

Select Approver (Node 2)

This function activity determines who the next approver is for the requisition by checking the 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 *WF_REQDEMO.SelectApprover*

Result Type Boolean

Prerequisite Activities None

Notify Requestor No Approver Available (Node 3)

This activity notifies the requisition preparer that no appropriate approver could be found for the requisition. The message includes 'Send' attributes that display the requisition number, requisition description, requisition amount, and who the last approver was, if there was any.

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 *WF_REQDEMO.Forward_Req*

Result Type None

Prerequisite Activities Select Approver

And (Node 7)

This Standard function activity merges two or more parallel branches in the flow only when the activities in all of those branches complete.

Function *WF_STANDARD.ANDJOIN*

Result Type None

Prerequisite Activities Must have at least two separate activities that each transition into this activity.

Notify Approver (Node 8)

This activity is a subprocess that notifies the approver that an action needs to be taken to either approve or reject the requisition. To view the subprocess, double–click on Notify Approver under the Processes branch in the navigator tree. The subprocess sends a notification to the approver and if the approver does not respond within a specified time, sends another reminder notification to the approver to take action. See: Summary of the Notify Approver Subprocess: page 13 – 19.

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	*WF_REQDEMO.Approve_Req*
Result Type	None
Prerequisite Activities	Select Approver, Notify Approver, Verify Authority

Notify Requestor of Approval (Node 14)

This activity notifies the requisition preparer that the requisition was approved. The message includes "Send" attributes that display the requisition number, requisition description, requisition amount, approver name, and comments from the approver.

If you display the property page of the activity node, you see that the activity is assigned to a performer whose name is stored in an item type attribute named Requestor Username.

Message	Requisition Approved
Result Type	None
Prerequisite Activities	Select Approver, Notify Approver, Verify Authority

End (Nodes 4, 11, and 15)

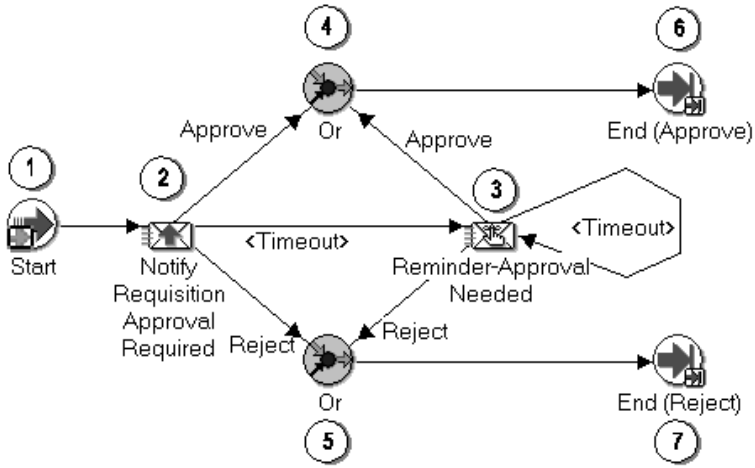
This function activity marks the end of the process. Although the activity itself does not have a result type, each node of this activity in the process must have a process result assigned to it. The process result is assigned in the property page of the activity node. Since the Requisition process activity has a result type of Approval, each End activity node must have a process result matching one of the lookup codes in the Approval lookup type.

Function	*WF_STANDARD.NOOP*
Result Type	None
Prerequisite Activities	Start

Summary of the Notify Approver Subprocess

To view the properties of the Notify Approver subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. The Notify Approver subprocess has a result type of Approval, indicating that when the subprocess completes, it has a result of Approve or Reject (based on the lookup codes in the Approval lookup type). It is not runnable, indicating that it cannot be initiated as a top level process to run, but rather can only be run when called by another higher level process as a subprocess.

When you display the Process window for the Notify Approver subprocess, you see that the subprocess consists of 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 Approve to the top level Requisition process. Similarly, if the approver rejects the requisition, the subprocess ends at node 7 and returns the result Reject to Requisition 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 Activities None

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 a special RESULT attribute and a "Respond" attribute. The RESULT attribute has a display name of Action and prompts the approver to respond with a value of `'APPROVE'` or `'REJECT'` from the lookup type called Approval. The value that the approver selects becomes the result that determines which activity branch the Workflow Engine transitions to next.

The "Respond" attribute is called Note and this attribute prompts the approver for optional comments to include in the notification response.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in an item type attribute named Forward To Username.

Message Requisition Approval Required

Result Type Approval

Prerequisite Activities Select Approver

Reminder–Approval Needed (Node 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 a special RESULT attribute and a "Respond" attribute. The RESULT attribute has a display name of Action and prompts the approver to respond with a value of `'APPROVE'` or `'REJECT'` from the lookup type called Approval. The value that the approver selects becomes the result that determines which activity branch the Workflow Engine transitions to next.

The "Respond" attribute is called Note and this attribute prompts the approver for optional comments to include in the notification response.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in an item type attribute named Forward To Username.

Message Requisition Approval Required Reminder

Result Type Approval

Prerequisite Activities Select Approver, Notify Requisition Approval Required

Or (Nodes 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 *WF_STANDARD.ORJOIN*

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 *WF_STANDARD.NOOP*

Result Type None

Prerequisite Activities Start

Sample StartProcess Function

Both *wfrund.sql* and the Requisition Demonstration web page call a PL/SQL stored procedure named *WF_REQDEMO.StartProcess* to initiate the Requisition process.

To examine *StartProcess* in more detail, we divide the procedure into several sections and number each section with the notation 1⇒ for easy referencing. The numbers 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 => 'FORWARD_TO_USERNAME',
                                                 avalue => RequestorUsername);
10⇒             wf_engine.SetItemAttrText       (itemtype => itemtype,
                                                 itemkey  => itemkey,
                                                aname => 'REQUISITION_PROCESS_OWNER',
                                                 avalue => ProcessOwner);
11⇒             wf_engine.SetItemAttrText       (itemtype => itemtype,
                                                 itemkey  => itemkey,
                                                 aname => 'MONITOR_URL',
                                                 avalue => wf_monitor.GetUrl
                                                (wfa_html.base_url, itemtype, itemkey,
                                                'NO'));
12⇒             wf_engine.SetItemAttrText       (itemtype => itemtype,
                                                 itemkey  => itemkey,
                                                 aname => 'REQ_DOCUMENT',
                                                 avalue => 'PLSQL:
                                                wf_reqdemo.create_req_document/
                                                ||ItemType||':'||ItemKey);
13⇒             wf_engine.SetItemAttrText       (itemtype => itemtype,
```

```
                                                    itemkey  => itemkey,
                                                    aname => 'REM_DOCUMENT',
                                                    avalue => 'PLSQL:wf_reqdemo.
                                                   reminder_req_document/'||Item-
                                                   Type||':'||ItemKey);
14⇒             wf_engine.SetItemOwner          (itemtype => itemtype,
                                                    itemkey  => itemkey,
                                                    owner => ProcessOwner);
15⇒             wf_engine.StartProcess          (itemtype => itemtype,
                                                    itemkey => itemkey );
16⇒  end StartProcess;
```

1⇒ This section represents the specification of the procedure, which includes the list of parameters that must be passed to *StartProcess*. It uses the same parameter values that you pass to the *wfrund.sql* script or to the field values entered in the Requisition Demonstration web page (WF_REQDEMO.Create_Req).

2⇒ The declarative part of the procedure body begins in this section. *StartProcess* consists of calls to various Workflow Engine PL/SQL APIs. See: Workflow Engine APIs: page 8 – 14.

Since all of these APIs require an item type and item key input, we define `ItemType` and `ItemKey` as local arguments. The argument `ItemType` is defined as `'WFDEMO'`, which is the internal name for the Requisition item type. The argument `ItemKey` is the value of the `RequisitionNumber` parameter that is passed to the *StartProcess* procedure.

3⇒ The executable part of the procedure body begins here. This section calls the *CreateProcess* Workflow Engine API. This API creates a new runtime instance of the Requisition process, whose internal name is `'WFDEMO'`, and is identified by the item type and item key that is supplied. See: CreateProcess: page 8 – 16.

> **Note:** If you do not pass a value for *<process_int_name>* to the *wfrund.sql* script, the selector function for the Requisition item type determines what process to run.

4⇒ This section calls the SetItemUserKey Workflow Engine API to mark the new runtime instance of the Requisition process with an end–user key. The end–user key makes it easier for users to query and identify the process instance when it is displayed. See: SetItemUserKey: page 8 – 18.

5, 6, 7, 8, 9, 10, 11, 12, and 13⇒ These sections call either the *SetItemAttributeText* or *SetItemAttributeNumber* Workflow Engine APIs to set values for the item type attributes defined for this process. The

attributes are `REQUISITION_NUMBER`, `REQUISITION_DESCRIPTION`, `REQUISITION_AMOUNT`, `REQUESTOR_NAME`, `FORWARD_TO_USERNAME`, `REQUISITION_PROCESS_OWNER`, `MONITOR_URL`, `REQ_DOCUMENT`, and `REM_DOCUMENT`, respectively. See: SetItemAttribute: page 8 – 36.

14⇒ This section calls the SetItemOwner Workflow Engine API to mark the new runtime instance of the Requisition process with a process owner user name. Users can query for process instances by process owner. See: SetItemOwner: page 8 – 21.

15⇒ This section calls the Oracle Workflow Engine *StartProcess* API to invoke the Requisition process for the item type and item key specified. See: StartProcess: page 8 – 22.

Example Function Activities

In general, a function activity must have the following information specified in its Activity property page:

- Internal name for the activity.
- Display name for the activity.
- Result type for the activity, which can be none or the name of a predefined lookup type.
- Name of the PL/SQL stored procedure that the activity calls.

Also, the PL/SQL stored procedure that a function activity calls must comply with a specific API. See: Standard API for PL/SQL Procedures Called by Function Activities: page 7 – 2.

You can view the scripts that create the *`WF_REQDEMO`* stored procedure package used by the Requisition 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⇒    l_forward_to_username := wf_reqdemo.GetManager( l_forward_from_username);
7⇒    wf_engine.SetItemAttrText                (itemtype => itemtype;
                                              itemkey => itemkey,
                                              aname => 'FORWARD_TO_USERNAME';
                                              avalue => l_forward_to_username);
8⇒    if (l_forward_to_username is null) then
               resultout :='COMPLETE:F';
```

```
      else
               resultout :='COMPLETE:T';
      end if;
9⇒    end if;
10⇒   if (funcmode = 'CANCEL') then
               resultout :='COMPLETE';
               return;
      end if;
11⇒   if (funcmode = 'TIMEOUT') then
               resultout :='COMPLETE';
               return;
      end if;
12⇒   exception
               when others then
                        wf_core.context('WF_REQDEMO','SelectorApprover',itemtype,
                        itemkey,actid,funcmode);
                        raise;
13⇒   end SelectApprover;
```

1⇒ The local arguments `l_forward_from_username,` and `l_forward_to_username` are declared in this section.

2⇒ If the value of `funcmode` is RUN, then retrieve the name of the last person that this requisition was forwarded to for approval by assigning `l_forward_to_username` to the value of the `FORWARD_TO_USERNAME` item type attribute, determined by calling the Workflow Engine API *GetItemAttrText*. See: GetItemAttribute: page 8 – 41.

3⇒ If the value of `l_forward_to_username` is null, then it means that the requisition has never been forwarded for approval. In this case, assign it the value of the `REQUESTOR_USERNAME` item type attribute, determined by calling the Workflow Engine API *GetItemAttrText*.

4⇒ Assign `l_forward_from_username` to the value of `l_forward_to_username`.

5⇒ This section assigns the value of `l_forward_from_username` to the `FORWARD_FROM_USERNAME` item type attribute by calling the Workflow Engine *SetItemAttrText* API.

6⇒ This section calls the function *GetManager* to return the manager of the previous approver stored in `l_forward_from_username,` from

the WF_REQDEMO_EMP_HIERARCHY table and assigns that manager's name to `l_forward_to_username`.

7⇒ This section assigns the value of `l_forward_to_username` to the `FORWARD_TO_USERNAME` item type attribute by calling the Workflow Engine *SetItemAttrText* API.

8⇒ If `l_forward_to_username` is null, meaning there is no manager above the previous approver in the hierarchy, then assign `resultout` to be `COMPLETE:F`. Otherwise, assign `resultout` to be `COMPLETE:T`.

9⇒ This ends the check on `funcmode` =' RUN'.

10⇒ If the value of `funcmode` is CANCEL, then assign `resultout` to be COMPLETE.

11⇒ If the value of `funcmode` is TIMEOUT, then assign `resultout` to be COMPLETE.

12⇒ This section calls WF_CORE.CONTEXT if an exception occurs.

13⇒ 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 `'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 8 – 41.

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 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, `'APPROVE'` or `'REJECT'` are defined in a lookup type called Approval. These values are defined by the message's special Result attribute, whose display name is Action. These values are also the possible results of the notification activity, as defined by the Result Type field in the Activity property page.

Message

The content of the notification is defined in the message called Requisition Approval Required:

Subject `Requisition &REQUISITION_NUMBER, &REQUISITION_DESCRIPTION for &REQUISITION_AMOUNT requires your approval`

Body `&REQ_DOCUMENT`

Message 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 message attributes referenced in the subject line and body of the message have to be defined with a source of 'Send'.

In this example, the message body contains a single message attribute called `REQ_DOCUMENT`. `REQ_DOCUMENT` is a PL/SQL Document–type attribute that references an item type attribute of the same name. When you initiate the Requisition process and the Workflow Engine runs the *StartProcess* procedure, it calls *SetItemAttrText( )* to set the `REQ_DOCUMENT` item attribute to the following value:

```
'PLSQL:wf_reqdemo.create_req_document/'||ItemType||':
'||ItemKey
```

This value calls the PL/SQL function `wf_reqdemo.create_req_document` to create a PL/SQL document for the specified itemtype:itemkey.

This message also contains a special result message attribute called Action and a 'Respond' message attribute called Note.

The result message attribute is defined in the Result tab of the message's property page. The result attribute prompts the approver to respond with a value from a list of possible values provided by the lookup type specified. The response, in turn, becomes the result of the Notify Requisition Approval Required activity. In this case, the possible response values are 'APPROVE' or 'REJECT', as defined by the Approval lookup type. This result determines which activity the process transitions to next.

The 'Respond' message attribute Note is of type 'Text'. This attribute prompts the approver to enter optional comments when responding to the notification.

> **Note:** To view the content of any message, double–click on the message in the navigator tree or select the message and choose Properties from the Edit menu.

Process Node Properties

If you display the properties of the Notify Requisition Approval Required activity node in the Notify Approver subprocess diagram you should see that this node is set to Normal because it is neither the start nor end activity in the process.

You should also see that the Performer is set to the `Forward To Username` item type attribute, indicating that the notification gets sent to the user whose name is stored in the item type attribute called `'Forward To Username'`. The value of `'Forward To Username'` is determined earlier in the Requisition process by the activity called Select Approver.

Product Survey Process

You can initiate a sample workflow process that sends a survey to elicit individual responses from a group. There are two different survey processes that you can initiate. Each survey process implements the survey method differently. However, both survey processes are based on a table that stores survey responses and a sequence that creates unique survey IDs.

You can set up and initiate this example process if you are using the standalone version of Oracle Workflow. If you are using Oracle Workflow embedded in Oracle Applications, you should consider this process mainly as an example for explanation purposes and not for demonstration use. The files necessary to 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.

You can initiate a Product Survey process from the Oracle Workflow Launch Processes web page or from the Workflow Demonstrations web page. When you initiate a Product Survey process, you must specify a survey requestor role, a survey participant role, a survey name, a timeout in number of minutes, and the process name to run. You can select one of two different process names:

- Survey – Single Process—Initiates a single process to send a survey to all participants in the role.
- Survey – Master/Detail Process—Initiates a master process that identifies all participants in a role and then spawns a detail survey process for each participant. Each detail survey process sends a personalized survey to a single participant.

When you choose Single Process, the process sends a notification with Expand Roles enabled to the survey participants role. This causes the notification system to send an individual copy of the survey to each user in that group role. A post–notification function associated with the survey notification activity validates the responses and writes them to a table once the notification times out or all responses are received. The process then sends an FYI notification to the survey participants with the results of the survey. The process ends without a result.

When you choose Master/Detail Process, a master process determines all the participant users in the survey role and creates a detail work item for each user. The master process then waits until all detail work items complete before continuing. The detail work item is a detail process that sends a survey notification to a single user. A post–notification function associated with the survey notification activity in the detail process validates the response received and writes it to a table. Once all detail work items time out or all detail responses are received, the Workflow Engine returns control to the master process. An FYI notification in the master process then sends the results of the survey to all the survey participants. The process ends without a result.

Installing the Product Survey Data Model

The Product Survey data model is available for installation only for the standalone version of Oracle Workflow. The data model is installed using Oracle Installer. Oracle Installer asks if you wish to install the sample workflows and their corresponding data models when you install the Oracle Workflow server. The files used in the installation are copied to the *demo* and *demo/<language>* subdirectories of your Oracle Workflow server directory structure.

Attention: For the Product Survey process demonstration to work properly, you should perform the steps required to set up Oracle Workflow after you install the Requisition data model. See: Overview of Setting Up: page 2 – 4.

The installation does the following:

- Calls a script called *wfsrvc.sql* to create a table called WF_SURVEY_DEMO and a sequence called WF_SURVEYDEMO_S. The Product Survey process updates the table WF_SURVEY_DEMO with information from each survey participant's response.
- The data model for the Product Survey process also relies on the demonstration directory service used by the sample Requisition process. See: Installing the Requisition Data Model: page 13 – 6.
- Calls the scripts *wfsrvs.sql* and *wfsrvb.sql* to create the PL/SQL spec and body for a package called WF_SURVEYDEMO. This package contains:
  - The PL/SQL stored procedures called by the function activities used in the Product Survey workflow.

– The PL/SQL procedure WF_SURVEYDEMO.Create_Survey called by the Oracle Workflow web agent to generate the web–based interface page for the Product Survey demonstration.

- Loads the Product Survey workflow definition from *wfsrv.wft* into the database. You can view this process in Oracle Workflow Builder.

Initiating the Product Survey Workflow

You can use either of the following methods to initiate the Product Survey workflow:

- Access the Product Survey web page from the Workflow Demonstrations home page. See: To Use the Product Survey Web Page: page 13 – 35.
- Use the Launch Processes web page. See: Testing Workflow Definitions: page 12 – 2.

▶ To Use the Product Survey Web Page

1. Enter the following URL in a web browser to access the Workflow Demonstration web page, then click on the Product Survey link to display the Product Survey web page:

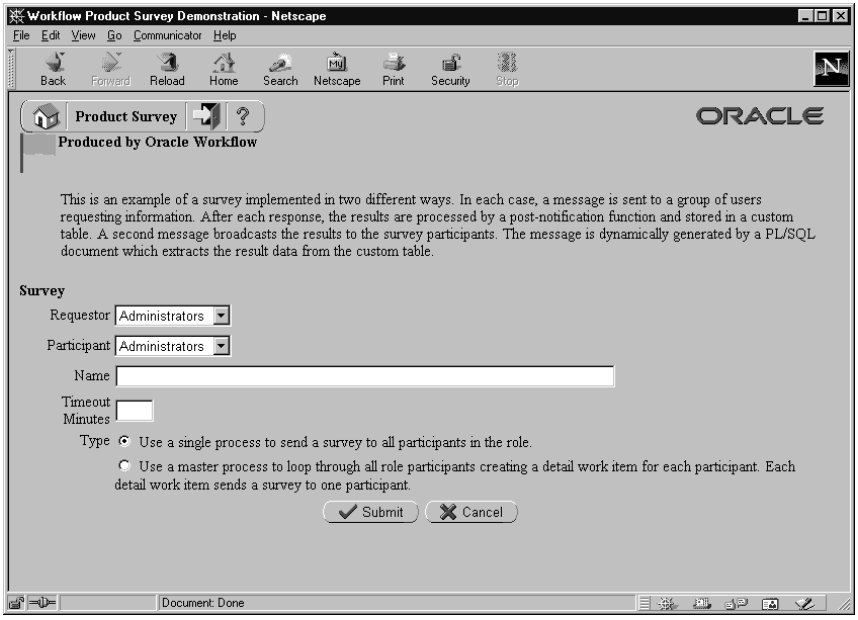
   `<webagent>/wf_demo.home`

   `<webagent>` represents the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 12.

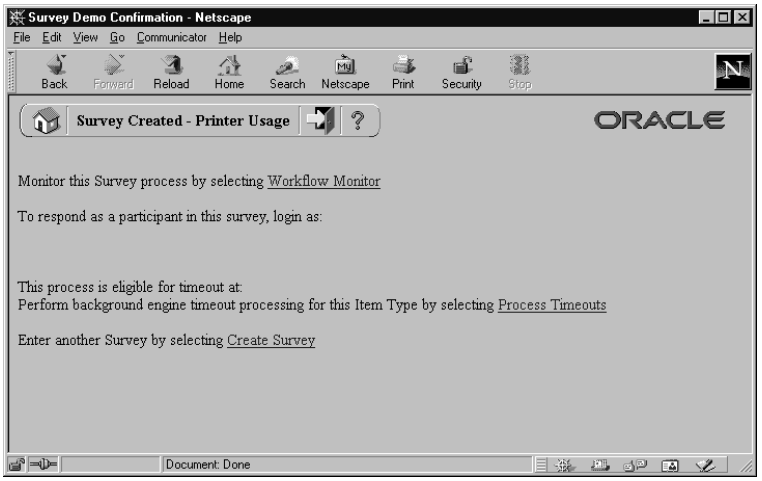
   Alternatively, you can enter the following URL to directly display the Product Survey web page:

   `<webagent>/wf_surveydemo.create_survey`

   Attention: These are both secured pages, so if you have not yet logged on as a valid workflow user in the current web session, you will be prompted to do so before the page appears.



2. Select a role name as the survey requestor.
3. Select a role name that represents the survey participants
4. Enter a name for your survey.
5. Specify a timeout value in minutes.
6. Check the type of survey process you wish to initiate:
   - Use a single process to send a survey to all participants in the role.
   - Use a master process to loop through all role participants creating a detail work item for each participant. Each detail work item sends a survey to one participant.
7. Choose submit to initiate the Product Survey process and to navigate to the Survey Created confirmation page.



8. In addition to telling you what roles you should log in as to view the process' notifications, the confirmation page also contains a HTML link to the Workflow Monitor Activities List where you can choose View Diagram to display the process diagram for the survey you submitted in ADMIN mode. See: Workflow Monitor: page 11 – 2.
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.
10. Select the Create Survey HTML link if you wish to enter and submit another survey in the Product Survey web page.

The Product Survey Item Type

The Survey – Single Process and Survey – Master/Detail Process are both associated with an item type called Product Survey. This item type identifies all product survey workflow processes available. Currently there are three workflow processes that support the two survey implementations:

- Survey – Single Process
- Survey – Master/Detail Process
- Detail Survey Process.

If you examine the property page of Product Survey, you see that it has a persistence type of Temporary and persistence number of days of 0.

This means that the run time data associated with any item instances for this item type are eligible for purging as soon as they complete. There is no Selector function specified because the process to start is specified when you initiate the Product Survey from the web–based interface.

The Product Survey item type also has several associated attributes. These attributes record information provided when you initiate a Product Survey. The attributes are used and maintained by function activities as well as notification activities throughout each process.

Product Survey Item Type Attributes

Display Name	Description	Type	Length/Format/Lookup Type
Document ID	Document ID (Defaults to item key)	Text	30
Survey Name	Survey name (Defaults to user key)	Text	80
Survey Participants	Survey participants role	Role	
Individual Participant	Role used by Detail Survey Process as recipient for survey notification	Role	
Timeout in Minutes	Dynamic timeout period for survey response	Number	

Table 13 – 3 (Page 1 of 1)

Summary of the Survey – Single Process

To view the properties of Survey – Single Process, select the process in the navigator tree, then choose Properties from the Edit menu. The Survey – Single Process is runnable, which means you can initiate it 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 Survey – Single Process has an associated error item type and error process called WFERROR and DEFAULT_ERROR, respectively. The DEFAULT_ERROR process of item type WFERROR is initiated

automatically when an error is encountered in Survey – Single Process. Currently the DEFAULT_ERROR process notifies the administrator of the error and provides options to retry, abort, or continue the process in error.

When you display the Process window for Survey – Single Process, you see that the process consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

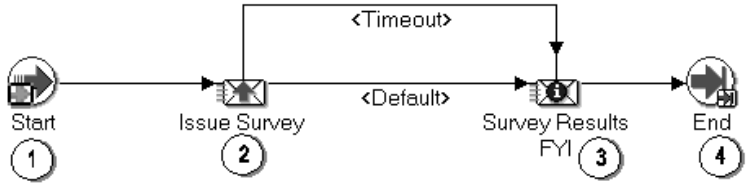
The Survey – Single Process workflow begins when you submit a survey using the Product Survey web page accessible from the Workflow Demonstrations home page. You must provide a survey requestor role, survey participant role, survey name, a timeout value in minutes, and check "Use a single process to send a survey to all participants in the role."

> **Note:** If you choose to initiate the survey process from the Launch Processes web page, you should select "Survey – Single Process" as the process name.

The workflow begins at node 1 with the Start activity.

At node 2, the process sends the survey to the participant role asking the role to rank the product and specify additional comments.

When the Workflow Engine receives all responses or the survey request times out, (based on the timeout period in minutes provided at initiation of the workflow), the process transitions to node 3 where the process sends a notification with the results of the survey to the participant role. The process ends at this point.



Start (Node 1)

This is a Standard function activity that simply marks the start of the process.

- Function—*WF_STANDARD.NOOP*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Issue Survey (Node 2)

This activity notifies the Survey Participant role members that their input is required for a product survey. The message includes two 'Send' attributes that display the Survey name and the timeout period in minutes.

The message also includes two 'Respond' attributes that request a ranking and comments.

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 attribute named Survey Participants. You will also see that the Timeout associated with this activity is stored in an item attribute named Timeout in Minutes.

- Message—Survey
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Expand Roles—Yes
- Notification Function—PLSQL:WF_SURVEYDEMO.PROCESS_SURVEY

Survey Results FYI (Node 3)

This activity notifies the survey participants of the results of the survey. The message includes two "Send" attributes that display the Survey

name and the Result Script. Result Script is of type PL/SQL Document and generates a PL/SQL document that summarizes the survey results.

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 attribute named Survey Participants.

- Message—Survey results
- Result Type—None
- Required—No
- Prerequisite Activities—Issue Survey
- Expand Roles—Yes
- Notification Function—None

End (Node 4)

This is a Standard function activity that simply marks the end of the process.

- Function—*WF_STANDARD.NOOP*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Summary of the Survey – Master/Detail Process

To view the properties of Survey – Master/Detail Process, select the process in the navigator tree, then choose Properties from the Edit menu. The Survey – Master/Detail Process is runnable, indicating that you can initiate it 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 Survey – Master/Detail Process has an associated error item type and error process called WFERROR and DEFAULT_ERROR, respectively. The DEFAULT_ERROR process of item type WFERROR is initiated automatically when an error is encountered in Survey – Master/Detail Process. Currently the DEFAULT_ERROR process notifies the

administrator of the error and provides options to retry, abort, or continue the process in error.

When you display the Process window for Survey – Master/Detail Process, you see that the process consists of five unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

The Survey – Master/Detail Process workflow begins when you submit a survey using the Product Survey web page accessible from the Workflow Demonstrations home page. You must provide a survey requestor role, survey participant role, survey name, a timeout value in minutes, and check "Use a master process to loop through all role participants creating a detail work item for each participant. Each detail work item sends a survey to one participant."

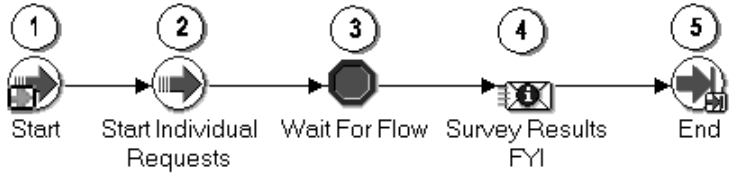
> **Note:** If you choose to initiate the survey process from the Launch Processes web page, you should select "Survey – Master/Detail Process" as the process name.

The workflow begins at node 1 with the Start activity.

At node 2, the process determines the individual users that are members of the survey participants role and starts a detail work item to send a survey to each user.

At node 3, the process waits for all of the detail work items to complete before continuing.

When all detail work items are complete the process transitions to node 4 where the process sends a notification with the results of the survey to the participant role. The process ends at this point.



Start (Node 1)

This is a Standard function activity that simply marks the start of the process.

- Function—*WF_STANDARD.NOOP*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Start individual requests (Node 2)

This function activity starts a detail work item for each member of the Survey Participants role, thus creating an individual survey for each user. The function copies all relevant item attribute values from the Survey – Master/Detail Process to each Detail Survey Process work item. In addition, the Master/Detail Process item key is set as the parent item for each Detail Survey Process work item.

- Function—*WF_SURVEYDEMO.START_CHILDREN*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Item Attributes Retrieved by Function—USERKEY, TIMEOUT_MINUTES
- Item Attributes Set by Function—for each Detail Survey Process work item: USERKEY, TIMEOUT_MINUTES

Wait For Flow (Node 3)

This is a standard function activity that is used here to pause the flow until the corresponding detail processes complete a specified activity.

- Function—*WF_STANDARD.WAITFORFLOW*
- Result Type—None
- Required—Yes

- Prerequisite Activities—Start individual requests
- Activity Attributes Retrieved by Function
  - Continuation Activity Label: Constant, CONTINUEFLOW
  - Continuation Flow: Constant, Detail
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Survey Results FYI (Node 4)

This activity notifies the survey participants of the results of the survey. The message includes two 'Send' attributes that display the Survey name and the Result Script. Result Script is of type PL/SQL Document and generates a PL/SQL document that summarizes the survey results.

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 attribute named Survey Participants.

- Message—Survey results
- Result Type—None
- Required—No
- Prerequisite Activities—Wait For Flow
- Expand Roles—Yes
- Notification Function—None

End (Node 5)

This is a Standard function activity that simply marks the end of the process.

- Function—*WF_STANDARD.NOOP*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Summary of the Detail Survey Process

To view the properties of Detail Survey Process, select the process in the navigator tree, then choose Properties from the Edit menu. The Detail Survey Process is not runnable, indicating that you cannot initiate it 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 Detail Survey Process does not have an associated error item type and error process. If an error is encountered, the error process initiated will be determined by the error item type and error process associated with the parent process of the detail work item, Survey – Master/Detail Process.

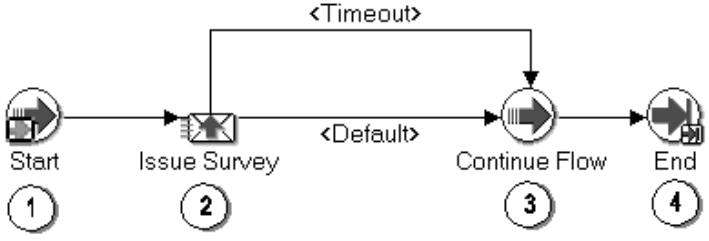
When you display the Process window for Detail Survey Process, you see that the process consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

The Detail Survey Process workflow begins when the parent process Survey – Master/Detail Process creates a detail work item from the Start Individual Requests activity.

The workflow begins at node 1 with the Start activity.

At node 2, the process sends the survey to an individual role asking for product ranking and comments.

When responses from all detail work items are received or all survey requests time out, (based on the timeout period in minutes provided at initiation of the workflow), the process transitions to node 3 where the process continues the flow of the parent process, Survey – Master/Detail Process. The process ends at this point.



Detail Survey Process Activities

Start (Node 1)

This is a Standard function activity that simply marks the start of the process.

- Function—*WF_STANDARD.NOOP*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Issue Survey (Node 2)

This activity notifies the individual role that a response is required for a product survey. The message includes two 'Send' attributes that display the Survey name and the timeout period in minutes.

The message also includes two 'Respond' attributes that request a product ranking and comments.

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 attribute named Individual Participant. You will also see that the Timeout associated with this activity is stored in an item attribute named Timeout in Minutes.

- Message—Survey
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Expand Roles—No
- Notification
  Function—PLSQL:WF_SURVEYDEMO.PROCESS_SURVEY

Continue Flow (Node 3)

This is a standard function activity that is used here to mark the position in the detail process where, upon completion, the corresponding halted master process will continue.

- Function—*WF_STANDARD.CONTINUEFLOW*
- Result Type—None
- Required—Yes
- Prerequisite Activities—Issue Survey
- Activity Attributes Retrieved by Function
  - Waiting Activity Label: Constant, WAITFORFLOW
  - Waiting Flow: Constant, Master
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

End (Node 4)

This is a Standard function activity that simply marks the end of the process.

- Function—*WF_STANDARD.NOOP*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Document Review Process

The Document Review Process requests approvers to review and approve attached documents by integrating notifications with a document management system. Any user participating in the Oracle Workflow administration role can initiate the Document Review process from the Oracle Workflow Launch Processes web page or from the Workflow Demonstrations web page. You must provide the following item attribute values to launch the process: Item Key, User Key, Process Owner, Send Document, Document Owner, and Document Reviewer.

For the standalone version of Oracle Workflow, Oracle Installer installs the Document Review process definition if you indicate you want to install the sample workflow processes. For the version of Oracle Workflow embedded in Oracle Applications, AutoInstall automatically installs the process definition of the Document Review Process for you.

When you submit a Document Review request in this demonstration, the process sends a notification to the designated reviewer to approve a document and, optionally, allows the reviewer to provide an alternate document in response. If the reviewer approves the document, the process ends with a result of Approve. If the reviewer rejects the document, the requestor has the option to resubmit the document for approval. If the requestor chooses to resubmit the document for approval, the process loops back to send the Review Document notification. Otherwise, the process ends with a result of Reject.

See Also

Sample Workflow Processes: page 13 – 2

Testing Workflow Definitions: page 12 – 2

The Document Management Item Type

The Document Review process is associated with an item type called Document Management. This item type identifies all demonstration workflow processes associated with document management system 46 integration. Currently there is only one workflow process associated with Document Management: Document Review.

If you examine the property page of Document Management, you see that it has a persistence type of Temporary and persistence number of

days of 0. This means that the run time data associated with any work items for this item type are eligible for purging as soon as they complete. The item type does not have a Selector function because the process to start is specified when you initiate the Document Management process from the web–based interface.

The Document Management item type also has several associated attributes. These attributes record information provided when you initiate a Document Management process. The attributes are used and maintained by function activities as well as notification activities throughout each process.

Document Management Item Type Attributes

Display Name	Description	Type	Length/Format/ Lookup Type
Send Document	Document sent for review	Document	frame target – New Window
Document Owner	Owner of the document sent for review	Role	
Document Reviewer	Document reviewer role	Role	
Comments	Comments entered by reviewer	Text	
Response Document	Document provided with edits after review	Document	frame target – New Window

Table 13 – 4 (Page 1 of 1)

Summary of the Document Review Process

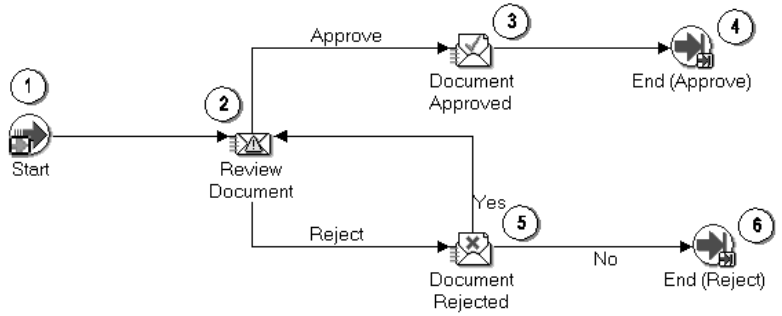
To view the properties of the Document Review process, select the process in the navigator tree, then choose Properties from the Edit menu. The Document Review process has a result type of Approval, indicating that when the process completes, it has a result of Approve or Reject (the lookup codes in the Approval lookup type associated with the Standard item type). This process activity is also runnable, indicating that you can initiate it 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 Document Review has an associated error item type and error process called WFERROR and DEFAULT_ERROR, respectively. The DEFAULT_ERROR process of item type WFERROR is initiated automatically when an error is encountered in Document Review. Currently the DEFAULT_ERROR process notifies the administrator of the error and provides options to retry, abort, or continue the process in error.

When you display the Process window for the Document Review process, you see that the process consists of six activity nodes. 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 workflow begins at node 1 with the Start activity.

At node 2, Review Document, the process sends a notification to the Document Reviewer requesting review and approval of the Send Document. If the reviewer approves the document, the process transitions to node 3, Document Approved, and notifies the requestor of the approval. The process ends in this case with a result of Approve. If the reviewer rejects the document, the process transitions to node 5, Document Rejected, and notifies the requestor of the rejection. The requestor has the option of resubmitting the document for approval or accepting the result. If the document is resubmitted, the process transitions back to node 2, Review Document. If the requestor accepts the result the process ends with a result of Reject.



Document Review Process Activities

Start (Node 1)

This is a Standard function activity that simply marks the start of the process.

- Function—*WF_STANDARD.NOOP*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Activity Attributes Retrieved by Function—None
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Review Document (Node 2)

This is a notification activity that sends a message to the document reviewer requesting approval and, optionally, a document with edits in return. The message includes two "Send" attributes, one that displays the Document Owner and the other a document attribute that appears in the notification as an attachment. When the attachment icon is selected, the document appears in a new window. The message also includes two "Respond" attributes that request Comments and a Response Document.

- Message—Document Send
- Result Type—Approval
- Required—Yes
- Prerequisite Activities—None
- Activity Attributes Retrieved by Notification—None

Document Approved (Node 3)

This is a notification activity that sends a message to the document review requestor stating that his/her document is approved. The message includes five "Send" attributes that display the Document Owner, Document Reviewer, a link to the Send Document, Comments from the reviewer, and an optional Response Document attachment. When the Response Document attachment is selected, the document appears in a new window.

- Message—Document Response—Approved
- Result Type—Approval
- Required—Yes
- Prerequisite Activities—Review Document
- Activity Attributes Retrieved by Notification—None

End (Nodes 4 and 6)

This is a Standard function activity that simply marks the end of the process.

- Function—*WF_STANDARD.NOOP*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Activity Attributes Retrieved by Function—None
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Document Rejected (Node 5)

This is a notification that sends a message to the document review requestor stating that his/her document is rejected. The message includes four "Send" attributes that display the Document Owner, Document Reviewer, a link to the Send Document, and an optional Response Document attachment. When the attachment icon is selected, the document appears in a new window. The message also includes a single "Respond" attribute that displays Comments from the reviewer and prompts for Comments from the requestor if the document review request is resubmitted.

- Message—Document Response – Rejected
- Result Type—Yes/No
- Required—Yes
- Prerequisite Activities—Review Document
- Activity Attributes Retrieved by Notification—None

Error Check Process

The Error Check Process scans the Oracle Workflow item activity statuses table for activities in error. The main purpose of this process is to illustrate how you can use Oracle Workflow to design a workflow process that has the same functionality as an Oracle Alert periodic alert. You can initiate the Error Check Process to scan for database exceptions at a specified frequency.

For the standalone version of Oracle Workflow, Oracle Installer installs the Error Check process definition if you indicate you want to install the sample workflow processes. For the version of Oracle Workflow embedded in Oracle Applications, AutoInstall automatically installs the process definition of the Error Check process for you.

When you launch the Error Check process in this demonstration, the process executes a function that scans the Workflow item activity statuses table looking for activities in error. If it finds errors, the process sends a notification listing the errors to the designated alert recipient. The process either continues or ends depending on whether the Error Check process is set to run only once or to run at a specified frequency. If you specify the process to run at a particular frequency, the process waits the necessary period of time and then scans the Workflow item activity statuses table again. The wait/scan loop continues until a specified end date is reached and the process ends.

You can initiate the Error Check process from the Launch Processes web page or from the Workflow Demonstrations web page. You must provide the Item Key, User Key, Process Owner, Alert Recipient, Start date of the process, End date of the process, and the Frequency (day of week, day of month, time of day, days, or Once only) with which you want to check for errors.

See Also

Sample Workflow Processes: page 13 – 2

Testing Workflow Definitions: page 12 – 2

The Periodic Alert Item Type

The Error Check process is associated with an item type called Periodic Alert. This item type identifies all processes associated with implementing periodic alert functionality through Oracle Workflow.

Currently there are two workflow processes associated with Periodic Alert: Error Check and User Defined Alert Action.

If you examine the property page of Periodic Alert, you see that it has a persistence type of Temporary and persistence number of days of 0. This means that the run time data associated with any item instances for this item type are eligible for purging as soon as they complete. A Selector function is not specified because the process to start is specified when you initiate a Periodic Alert process from the web–based interface.

The Periodic Alert item type also has several associated attributes. These attributes record information provided when you initiate a Periodic Alert process. The attributes are used and maintained by function activities as well as notification activities throughout each process.

Periodic Alert Item Type Attributes

Display Name	Description	Type	Length/Format/Lookup Type
Alert Recipient	Role to notify when alert exception is detected	Role	
Start date	Date to start alert check (default is today)	Date	DD–MON–YYYY HH24:MI:SS
End date	Date to end alert check (default is 12/31/2010 12:12:12 PM)	Date	DD–MON–YYYY HH24:MI:SS
Frequency	Frequency to check the alert	Lookup	Wait Mode
Frequency – day of month	Day of month required when Frequency is Day of Month	Lookup	Day of Month
Frequency – day of week	Day of week required when Frequency is Day of Week	Lookup	Day of Week
Frequency – time of day	Time optional, but used by Wait activity for any Frequency value	Date	HH24:MI
Frequency – days	Number of days required when Frequency is Relative Time	Number	
Once Only	Perform Alert Once Only	Lookup	Yes/No

Table 13 – 5 (Page 1 of 1)

Summary of the Error Check Process

To view the properties of the Error Check process, select the process in the navigator tree, then choose Properties from the Edit menu. The Error Check process is runnable, indicating that you can initiate it 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 Error Check has an associated error item type and error process called WFERROR and DEFAULT_ERROR, respectively. The DEFAULT_ERROR process of item type WFERROR is initiated automatically when an error is encountered in Error Check. Currently the DEFAULT_ERROR process notifies the administrator of the error and provides options to retry, abort, or continue the process in error.

When you display the process window for the Error Check process, you see that the process consists of six unique activities, several of which are reused to comprise the nine 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.

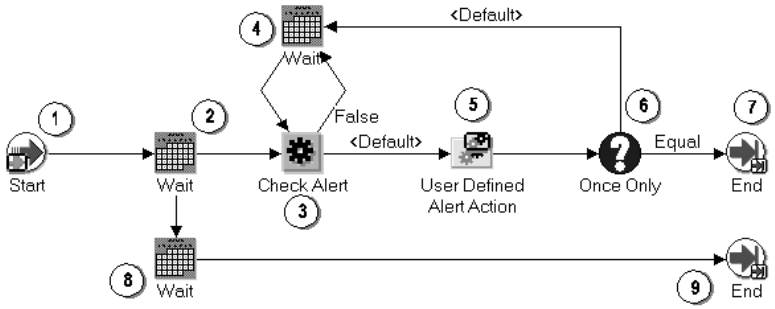
The Error Check workflow begins when you initiate the process from the Launch Processes web–based interface. You must provide the Item Key, User Key, Process Owner, Alert Recipient, Start date of the process, End date of the process, and the Frequency (day of week, day of month, time of day, days, or Once only) with which you want to check for errors.

The workflow begins at node 1 with the Start activity.

At node 2, the process pauses and waits until the Start Date. Once the wait time has elapsed, the process executes node 3, a function activity that scans the Workflow item activity statuses table for errors. If the function activity finds no errors, the process executes node 4 which pauses the process for some period of time based on the frequency you specify when you launch the process. Once this frequency–based wait time elapses, the process scans the status table for errors again. Otherwise, if it finds any errors, the process executes node 5, a process activity that sends a notification of the errors to the alert recipient. The process then executes node 6 to evaluate whether the Error Check process is to be run only once. If the process should only be run once, the process ends at node 7, otherwise the process returns to node 4, the frequency–based Wait activity.

While the set of activities between nodes 2 through 5 are being executed, the process also takes a parallel transition to node 8, another

Wait activity that serves to keep the work item scan loop going until the specified End Date is encountered. Once the end date is reached, the process ends at node 9.



Error Check Process Activities

Start (Node 1)

This is a Standard function activity that simply marks the start of the process.

- Function—*WF_STANDARD.NOOP*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Activity Attributes Retrieved by Function—None
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Wait (Node 2)

This is a Standard function activity that pauses the process for the time you specify.

To use a Wait activity in a process, you must set up at least one background engine to evaluate whether the wait period has elapsed so that it can complete the Wait activity.

- Function—*WF_STANDARD.WAIT*

- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Activity Attributes Retrieved by Function
  - Wait Mode: Constant, Absolute Date
  - Absolute Date: Item Attribute, Start date
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Check Alert (Node 3)

This is a function activity that scans for rows in the Workflow item activity statuses table looking for activities with a status of ERROR.

- Function—*WF_ALERT.CHECKALERT*
- Result Type—Boolean
- Required—Yes
- Prerequisite Activities—None
- Activity Attributes Retrieved by Function—None
- Item Attributes Created by Function—LAST_CHECKED
- Item Attributes Retrieved by Function—LAST_CHECKED
- Item Attributes Set by Function—LAST_CHECKED

Wait (Node 4)

This is a Standard function activity that pauses the process for the time you specify.

To use a Wait activity in a process, you must set up at least one background engine to evaluate whether the wait period has elapsed so that it can complete the Wait activity.

- Function—*WF_STANDARD.WAIT*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Activity Attributes Retrieved by Function

  - Wait Mode: Item Attribute, Frequency
  - Absolute Date: Item Attribute, Start date
  - Day of Month: Item Attribute, Day of Month
  - Day of Week: Item Attribute, Day of Week
  - Relative Time: Item Attribute, Frequency—days
  - Time of Day: Item Attribute, Frequency—time of day
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

User Defined Alert Action (Node 5)

This activity is a subprocess that performs a series of activities whenever an alert exception is detected. To view the subprocess, double–click on User Defined Alert Action under the Processes branch in the navigator tree. Currently, the subprocess sends a notification of the errors detected to the alert recipient.

- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Activity Attributes Retrieved by Function—None

Once Only (Node 6)

This is a standard function activity that compares one value to another.

- Function—*WF_STANDARD.COMPARE*
- Result Type—Comparison
- Required—Yes
- Prerequisite Activities—None
- Activity Attributes Retrieved by Function
  - Test Value: Item Attribute, Once Only
  - Reference Value: Constant, Y
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

End (Nodes 7 and 9)

This is a Standard function activity that simply marks the end of the process.

- Function—*WF_STANDARD.NOOP*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Activity Attributes Retrieved by Function—None
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Wait (Node 8)

This is a Standard function activity that pauses the process for the time you specify.

To use a Wait activity in a process, you must set up at least one background engine to evaluate whether the wait period has elapsed so that it can complete the Wait activity.

- Function—*WF_STANDARD.WAIT*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Activity Attributes Retrieved by Function
  - Wait Mode: Constant, Absolute Date
  - Absolute Date: Item Attribute, End date
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Summary of the User Defined Alert Action Process

To view the properties of the User Defined Alert Action process, select the process in the navigator tree, then choose Properties from the Edit menu. The User Defined Alert Action process is not runnable, indicating that you cannot initiate it as a top level process to run by

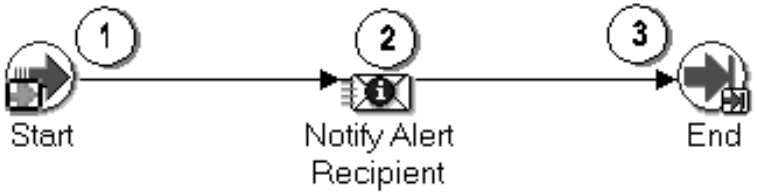
making calls to the Workflow Engine *CreateProcess* and *StartProcess* APIs.

The Details property page of the process activity indicates that this process activity does not have an associated error item type and error process. If an error is encountered, the error process initiated will be determined by the error item type and error process associated with the parent process, Error Check.

When you display the process window for the User Defined Alert Action process, you see that the process consists of three unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

The User Defined Alert Action process is initiated as a subprocess by the Error Check process.

The workflow begins at node 1 with the Start activity. At node 2, the process sends a notification listing the errors found to the alert recipient. The process ends at this point.



User Defined Alert Action Process Activities

Start (Node 1)

This is a Standard function activity that simply marks the start of the process.

- Function—*WF_STANDARD.NOOP*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Activity Attributes Retrieved by Function—None

- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

Notify Alert Recipient (Node 2)

This is a notification activity that sends an error report to the designated alert recipient.

The message includes one Send attribute, Error Report, which is a PL/SQL document attribute whose value is generated by the PLSQL procedure WF_ALERT.ErrorReport.

- Message—Exception detected FYI
- Result Type—None
- Required—No
- Prerequisite Activities—None

End (Node 3)

This is a Standard function activity that simply marks the end of the process.

- Function—*WF_STANDARD.NOOP*
- Result Type—None
- Required—Yes
- Prerequisite Activities—None
- Activity Attributes Retrieved by Function—None
- Item Attributes Set by Function—None
- Item Attributes Retrieved by Function—None

CHAPTER

14

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 14 – 4.
- Enable/disable a language—wfnlena.sql: page 14 – 9
- Run a workflow process—wfrun.sql: page 14 – 13.
- Start a background engine—wfbkg.sql: page 14 – 4.
- Show activities deferred for the next background engine execution—wfbkgchk.sql: page 14 – 5.
- Display a status report for an item
  - wfstatus.sql: page 14 – 13.
  - wfstat.sql: page 14 – 13.
- Show a notification's status—wfntfsh.sql: page 14 – 9.
- Reset the protection level for objects—wfprot.sql: page 14 – 9.
- Handle errored activities—wfretry.sql: page 14 – 11.
- Check for version and process definition errors
  - wfverchk.sql: page 14 – 14.
  - wfverupd.sql: page 14 – 14.
  - wfstdchk.sql: page 14 – 13.
- Check for invalid hanging foreign keys—wfrefchk.sql: page 14 – 10.
- Check the directory service data model—wfdirchk.sql: page 14 – 8.
- Clean up Workflow Queues in the system tables—wfqclean.sql: page 14 – 10.
- Change the internal name of a workflow object

  Note: Generally, you cannot update the internal name of a workflow object in Oracle Workflow Builder. However, if you load your process definition to a database, you can use one of these scripts to update a workflow object's internal name if no

runtime data exists for the object. You should only use these scripts to correct errors in an object's internal name during design time. Do not use these scripts to rename objects that are involved in running instances of processes.

  - wfchact.sql: page 14 – 5.
  - wfchacta.sql: page 14 – 6.
  - wfchita.sql: page 14 – 6.
  - wfchitt.sql: page 14 – 6.
  - wfchluc.sql: page 14 – 7.
  - wfchlut.sql: page 14 – 7.
  - wfchmsg.sql: page 14 – 7.
  - wfchmsga.sql: page 14 – 8.

- Remove data from Oracle Workflow tables
  - wfrmall.sql: page 14 – 11.
  - wfrmitms.sql: page 14 – 12.
  - wfrmitt.sql: page 14 – 12.
  - wfrmtype.sql: page 14 – 12.
  - wfrmita.sql: page 14 – 11.

  Note: In Oracle Applications, a standard concurrent program called "Purge Obsolete Workflow Runtime Data" is also available. See: FNDWFPR: page 14 – 4.

- Display the version of the Oracle Workflow server—wfver.sql: page 14 – 14.

FNDWFPR

For Oracle Workflow embedded in Oracle Applications, use the standard concurrent program FNDWFPR "Purge Obsolete Workflow Runtime Data" to purge old data from the Oracle Workflow runtime tables regularly.

Navigate to the Submit Requests form in Oracle Applications to submit the Purge Obsolete Workflow Runtime Data 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* and Submitting a Request, *Oracle Applications User's Guide*.

You can supply the following parameters for the "Purge Obsolete Workflow Runtime Data" concurrent program:

- Item Type—The item type to purge. Leaving this field blank defaults to purging the runtime data for all item types.
- Item Key—The item key to purge. Leaving this field blank defaults to purging the runtime data for all item keys.
- Age—Minimum age of data to purge, in days.

WFNLADD.sql

If you enable 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 – 25 and wfnlena.sql: page 14 – 9.

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. On completing it's current set of eligible activities to process, the background process waits for the specified

number of seconds before launching another background engine. This cycle continues until the indicated number of minutes have elapsed.

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 8 – 33

Setting up Background Workflow Engines: page 2 – 34

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 8 – 33

Setting up Background Workflow Engines: page 2 – 34

Wfchact.sql

Use wfchact.sql to change the internal name of an activity and update all references to the activity. See: Change the internal name of a workflow object: page 14 – 2.

Use the script as follows:

```
sqlplus <user/pwd> @wfchact <act_type> <old_act> <new_act>
```

Replace *<act_type>* with the item type that the activity you wish to update is associated with, replace *<old_act>* with the current internal

name of the activity, and replace *<new_act>* with the new internal name of the activity.

Wfchacta.sql

Use wfchacta.sql to change the internal name of an activity attribute and update all references to the activity attribute. See: Change the internal name of a workflow object: page 14 – 2.

Use the script as follows:

```
sqlplus <user/pwd> @wfchacta <act_type> <old_acta> <new_acta>
```

Replace *<act_type>* with the item type that the activity attribute you wish to update is associated with, replace *<old_acta>* with the current internal name of the activity attribute, and replace *<new_acta>* with the new internal name of the activity attribute.

Wfchita.sql

Use wfchita.sql to change the internal name of an item attribute and update all references to the item attribute. See: Change the internal name of a workflow object: page 14 – 2.

Use the script as follows:

```
sqlplus <user/pwd> @wfchita <item_type> <old_attr> <new_attr>
```

Replace *<item_type>* with the item type that the item attribute you wish to update is associated with, replace *<old_attr>* with the current internal name of the item attribute, and replace *<new_acta>* with the new internal name of the item attribute.

Wfchitt.sql

Use wfchitt.sql to change the internal name of an item type and update all references to the item type. See: Change the internal name of a workflow object: page 14 – 2.

Use the script as follows:

```
sqlplus <user/pwd> @wfchitt <old_type> <new_type>
```

Replace *<old_type>* with the current internal name of the item attribute, and replace *<new_type>* with the new internal name of the item attribute.

Wfchluc.sql

Use wfchluc.sql to change the internal name of a lookup code and update all references to the lookup code. See: Change the internal name of a workflow object: page 14 – 2.

Use the script as follows:

```
sqlplus <user/pwd> @wfchluc <lookup_type> <old_luc> <new_luc>
```

Replace *<lookup_type>* with the lookup type of the lookup code you wish to update, replace *<old_luc>* with the current internal name of the lookup code, and replace *<new_luc>* with the new internal name of the lookup code.

Wfchlut.sql

Use wfchlut.sql to change the internal name of a lookup type and update all references to the lookup type. See: Change the internal name of a workflow object: page 14 – 2.

Use the script as follows:

```
sqlplus <user/pwd> @wfchlut <old_lut> <new_lut>
```

Replace *<old_lut>* with the current internal name of the lookup type, replace *<new_lut>* with the new internal name of the lookup type.

Wfchmsg.sql

Use wfchmsg.sql to change the internal name of a message and update all references to the message. See: Change the internal name of a workflow object: page 14 – 2.

Use the script as follows:

```
sqlplus <user/pwd> @wfchmsg <msg_type> <old_msg> <new_msg>
```

Replace <msg_type> with the item type of the message you wish to update, replace <old_msg> with the current internal name of the message, replace <new_msg> with the new internal name of the message.

Wfchmsga.sql

Use wfchmsga.sql to change the internal name of a message attribute. This script does not update the message subject/body references to the message attribute. You must manually update the message attribute references. See: Change the internal name of a workflow object: page 14 – 2.

Use the script as follows:

```
sqlplus <user/pwd> @wfchmsga <msg_type> <msg_name> <old_attr>
<new_attr>
```

Replace <msg_type> with the item type of the message attribute you wish to update, replace <msg_name> with the internal name of the message that the message attribute belongs to, replace <old_attr> with the current internal name of the message attribute, and replace <new_attr> with the new internal name of the message attribute.

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
```

Wfnlena.sql

If you define a new language in your Oracle installation, use wfnlena.sql to enable or disable that language in Oracle Workflow. See: WFNLADD.sql: page 14 – 4.

Use the script as follows:

```
sqlplus <user/pwd> @wfnlena <language_code> <enable_flag>
```

Replace *<language_code>* with a valid language code, and replace *<enable_flag>* with Y to enable and N to disable the specified language.

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.

Wfqclean.sql

Use wfqclean.sql to clean up Workflow queues in the system tables.

Attention: This script is only necessary if you are using a version of Oracle8 prior to 8.1.5 and you drop your user or tablespace without previously dropping the workflow queues using wfqued.sql. The wfqued.sql script is located in the Oracle Workflow *sql* subdirectory. The DROP USER CASCADE and DROP TABLESPACE INCLUDING CONTENTS commands in these prior versions of Oracle8 leave queue data in your system tables that result in an ORA–600 error when you recreate the queues. To avoid this case, you should always run wfqued.sql to drop the queues prior to dropping the user or tablespace.

Use the wfqclean.sql script as follows:

```
sqlplus system/manager @wfqclean <un>
```

Replace *<un>* with username of the schema that experiences the ORA–600 error.

Wfrefchk.sql

Use wfrefchk.sql to check for invalid workflow data that is missing primary key data for a foreign key.

```
sqlplus <user/pwd> @wferfchk
```

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 8 – 53.

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.

Wfrmita.sql

Use wfrmita.sql to delete all workflow data for a specified item type attribute. This script prompts you for the item type and the name of the attribute to delete. Alternatively, you can use Oracle Workflow

Builder to delete an item type attribute from a workflow definition stored in a file or a database.

Use the script as follows:

```
sqlplus <user/pwd> @wfrmita
```

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.

Wfrmtype.sql

Use wfrmtype.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> @wfrmtype
```

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>
```

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>
```

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>
```

Wfstdchk.sql

Use wfstdchk.sql to check and report any problems found in the Oracle Workflow data model. For example, this script will report any function activities that reference invalid functions and scan the tables of each workflow process definition object to verify that each row has a valid internal name and display name.

Use the script as follows:

```
sqlplus <user/pwd> @wfstdchk
```

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
```

Wfverchk.sql

Use wfverchk.sql if you suspect that problems arising in your workflow process are due to multiple versions of an activity being active simultaneously. This script identifies errors in versions of activities that cause multiple versions to appear to be active at once.

Use the script as follows:

```
sqlplus <user/pwd> @wfverchk
```

Wfverupd.sql

Use wfverupd.sql to correct problems arising in your workflow process that are due to multiple versions of an activity being active simultaneously. This script identifies and corrects errors in versions of activities that cause multiple versions to appear to be active at once.

Use the script as follows:

```
sqlplus <user/pwd> @wfverupd
```

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.

New	Creates a new workspace for you to define an item type.
Quick Start Wizard	Creates a framework from which you can begin designing a workflow process definition. See: Quick Start Wizard Overview: page 3 – 18.
Open...	Opens a data store by prompting you to connect to a database or a file. See: Opening and Saving Item Types: page 3 – 12.
Close Store	Closes the selected data store. This menu option is available only if the Navigator is the active window.
Save	Saves changes to the currently connected database or file. See: Opening and Saving Item Types: page 3 – 12.
Save As	Save changes to the file or database you specify with an optional effective date.
Create Shortcut	Creates a shortcut icon on your desktop of the current Oracle Workflow Builder session. Prompts for a shortcut name. The shortcut runs Oracle Workflow Builder and automatically connects to the data store that was selected at the time you created the shortcut, loading in the item types and opening the process windows that were loaded and open at the time. If the data store is a database, the shortcut prompts for the database password before starting Oracle Workflow Builder. This feature is available only when you run Oracle

	Workflow Builder in Microsoft Windows 95 or Windows NT 4.0 or higher. Earlier versions of Microsoft Windows NT do not support the concept of shortcuts. See: Creating a Shortcut Icon for a Workflow Process: page 5 – 17.
Verify	Validates all process definitions in the current data store. Use Refresh to display the latest verification report of the process. See: To Validate a Process: page 5 – 15
Print Diagram	Prints the process diagram displayed in the active process window. See: To Print a Process: page 5 – 15.
Show/Hide Item Types...	Displays the Show Item Types window to determine which item types in the current data store to show or hide in the navigator tree.
Load Roles from Database	Loads the Oracle Workflow directory service roles from the current database store into Oracle Workflow Builder and makes them viewable from the Directory Service branch in the navigator tree as well as from any property page poplist field that references roles. This menu option is available only if the current data store is a database. See: Roles: page 5 – 19.
Exit	Exits Oracle Workflow Builder.

Edit Menu

The Edit menu varies depending on whether you select the Navigator window or a process window. The following menu options appear only when you select the Navigator window and apply only to the Navigator window:

New	Creates a new item type, function activity, process activity, notification activity, message, lookup type, lookup code, or attribute by displaying its property page(s).
Copy	Copies the selected object in the navigator tree.
Paste	Pastes the object from the clipboard into the selected branch of the navigator tree.
Delete	Deletes the selected object from the navigator tree.
Find	Displays the Search window so you can enter search criteria to find an object in the navigator

tree. See: To Find an Object in the Navigator Tree: page 3 – 6.

Find Again Finds an object in the navigator tree using the same criteria defined previously in the Search window.

Properties Shows the property pages of the selected object.

Process Details Opens the process window of the selected process activity.

Move Attribute Reorders the attributes listed in the current branch of the navigator tree by moving the selected attribute up or down the list.

The following menu options appear only when you select a process window and apply only to the selected process window:

Delete Selection Deletes the selected object(s) from the process window.

Properties Shows the property pages of the selected activity node.

Copy Diagram Copies the process diagram displayed in the active process window to the clipboard. See: To Copy a Process Diagram to the Clipboard: page 5 – 15

View Menu

The View menu lets you alter the display of Oracle Workflow Builder.

Font Displays the Fonts property page. Use the property page to change the font settings of the text that appear in the Navigator and process windows. Changes apply to all future sessions of Oracle Workflow Builder. See: Modifying Fonts in Oracle Workflow Builder: 5 – 16.

Log –> Show Toggles between displaying and hiding the Log window. The Message Log window displays messages from the Workflow Builder that are not error–related.

Log –> Detailed Toggles the debug mode of Oracle Workflow Builder on and off. When you check Detailed, you turn the debug mode on and cause Oracle Workflow Builder to write more extensive messages to the Log window. You should not check Detailed unless instructed to do so by your Oracle customer support representative, as this

	mode significantly slows down the Oracle Workflow Builder.
Log –>To File	Writes all future content of the Message Log window to a file. Select Log Show from the View menu to determine the location and name of the log file.
Log –> Bring to Front	Brings the Message Log window to the front as the active window.
Grid Snap	Toggles grid snap on or off for all process windows.
Show Label in Designer submenu	A submenu of options that let you control the information displayed in an activity's label. Choose either Instance Label, Internal Name, Display Name, Performer, or Comment.
Show Label in Designer –> Activity Label	Uses the node label as the label for each activity node in a process diagram. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.
Show Label in Designer –> Internal Name	Uses the internal name of an activity as the label for each activity node in a process diagram. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.
Show Label in Designer –> Display Name	Uses the display name of an activity as the label for each activity node in a process diagram. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.
Show Label in Designer –> Performer	Uses the activity's performer as the label for each activity node in a process diagram. Function and process activities that do not have performers do not have a label. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.
Show Label –> Comment	Uses the activity's comment as the label for each activity node in a process diagram. Activities that do not have a comment do not have a label. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.

Developer Mode Toggles the display between standard presentation mode and developer mode. In developer mode, all icons revert to the default icon for the specific object type/subtype, subprocess icons are distinct from top level process icons, and in the navigator tree, objects are shown and sorted by internal name. Note that attributes are shown by internal name but are not sorted.

If the Navigator window is the active window, then the following menu option also appears:

Split Window Splits the Navigator window horizontally or vertically.

If a process window is the active window, then the following menu options also appear:

Overview Displays the process Overview window. See: To Display a Process Overview: page 5 – 14.

Show Process in Navigator For the current process displayed in the process diagram window, this menu option locates its corresponding process activity in the Navigator window.

Show Overlay Image Toggles the display to either show or hide the overlay image for an icon, if it has one. For example, the Start and End activities in a process have a green arrow and red arrow overlay image, respectively.

Window Menu

The 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—Copies the selected object.

Paste—Pastes the copied object into the current object branch.

Verify—Validates the process definition.

Developer Mode—Toggles between Developer and Presentation display.

Find—Displays the Search window to specify the search criteria to locate an object in the navigator tree.

Quick Start Wizard—Runs the Quick Start Wizard to begin creating a workflow process definition.

Help—Displays help on how to use Oracle Workflow.

New Object—Creates a new object depending on the object branch you select (item type, Processes, Notifications, Functions, Messages, or Lookup Types) by displaying the property page for that object type.

Process Window Toolbar

The process window toolbar includes the following buttons which apply only to objects selected the current process window:

Open—Displays the Open window to open stored item types from a file or database.

Save—Saves any changes in the selected data store to the currently connected database or file. Displays the Open window to let you connect to a database or file if the selected data store is not connected to a database or file.

Print Diagram—Prints the current process diagram.

New Process—Displays the process activity node property page for you to create a new process activity.

New Notification—Displays the notification activity node property page for you to create a new notification activity.

New Function—Displays the function activity node property page for you to create a new function activity.

Delete Selection—Deletes the selected object.

Properties—Shows the property pages of the selected object.

Developer Mode—Toggles between Developer and Presentation display.

Find—Displays the process Overview window.

Show Instance Labels—Displays the instance label of the node as the node activity label in the Process window.

Show Internal Names—Displays the internal name of the node as the node activity label in the Process window.

Show Display Names—Displays the display name of the node as the node activity label in the Process window.

Show Comments—Displays the comments of the node as the node activity label in the Process window.

Show Performers—Displays the performer of the node as the node activity label in the Process window.

Help—Displays help on how to use Oracle Workflow.

APPENDIX

B

Oracle Applications Embedded Workflows

This appendix lists the workflows that are embedded in Oracle Applications and Oracle Self–Service Web Applications as well as Oracle's support policy towards the customization of these workflows.

Predefined Workflows Embedded in Oracle Applications and Oracle Self–Service Web Applications

You can use Oracle Workflow to customize the predefined workflow processes listed below. A full description of each workflow is documented in its respective product's User's Guide or Configuration Guide, if one is available.

> **Note:** Some Oracle Applications products use the Account Generator feature to dynamically create accounting flexfield combinations. The Account Generator has generic predefined workflow functions that each Oracle Application product uses in its own predefined Account Generator process. The Account Generator processes for each product are not listed in this section, but are documented in more detail in each respective product's User's Guide. A general discussion of the Account Generator feature is also available in the Oracle Applications Flexfields Guide.

See Also

Oracle Support Policy for Predefined Workflows: page B – 11

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 Application Object Library

Oracle Application Object Library provides a set of standard function activities that you can use to incorporate concurrent manager processing into any Oracle Applications workflow process. The

standard function activities are associated with the Concurrent Manager Functions item type. See: Concurrent Manager Standard Activities: page 6 – 16.

Oracle Business Intelligence System

BIS Management by Exceptions Process—This generic workflow process is a template for BIS customers to use as part of their Performance Management Framework. When actual performance does not meet expected performance, this process sends a basic corrective action notification with an embedded report URL. All other processes under the OBIS Corrective Action item type are similar to this generic process.

Oracle Self–Service Expenses

Expense Reporting Process—Oracle Self–Service Expenses uses the Expense Reporting workflow process to process the manager approval and accounting review of expense reports entered in Self–Service Expenses. The Expense Reporting process begins when a user submits an expense report, and finishes when an expense report is rejected or approved. If approved, the Expense Reporting process makes the expense report available for the Payables Invoice Import process. The Expense Report process notifies employees at key event points during the manager approval and accounting review process.

Oracle Self–Service Human Resources

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 workflow 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 workflow completes. The workflow keeps the originating manager informed of the offer status throughout the process.

Career Management Reviews Process—Sends notifications to reviews for Appraisals and Assessments.

360 Appraisals Process—Sends a notification to a set of people informing them that they should perform an appraisal as part of a group.

Other Processes—Oracle Self–Service HR includes processes that allow employees and mangers to view, update, and display approved personal details on the Web, including:

- Basic Details (Name, Marital status, and so on)
- Addresses
- Phone Numbers
- Contact Persons
- Resume
- Qualifications
- Personal Competence Profile
- School and College Attendances
- Work Choices

Oracle Self–Service HR also includes processes to allow employees to enroll in a class and apply for a job.

Oracle Self–Service Purchasing

Receipt Confirmation Process—Sends receipt notifications to requestors, informing them that they should have received their order. This process is also known as the PO Confirm Receipt workflow.

Requisition Approval Process—Submits a requisition created from Web Requisitions to the appropriate managers for approval and updates the status of the requisition.

Oracle 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 Process—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.

Step–Down AutoAllocations Process—Initiates the GL Allocation process and directs batches to the GL Mass Allocation process or the GL Recurring Journals process. These processes validate batches and determine if approvals are required for a batch, submit the batch(s) to approvers if required, then notify individuals of the approval results. If errors occur, the contact or responsibility can choose to rollback the Step–Down AutoAllocation process which reverses any posted journals.

Global InterCompany System—The Global Intercompany System (formerly CENTRA) is an enhanced feature for Release 11i. It provides an environment within which multiple companies can exchange intercompany transactions. When a sender company initiates an intercompany transaction, the workflow process notifies the receiver company of the transaction. The workflow process notifies the sender company if the receiver company approves or rejects the transaction. In addition, a threshold amount can be set to limit the volume of notifications or to notify approvers of only large transactions.

Oracle Federal Human Resources

GHR Personnel Action Process—Enables the routing of the Request for Personnel Action (RPA) Form for data entry, signature, and review

before the final approval and update to the database. Based on the agency's practices, the user can route the RPA to an individual, groupbox, or routing list within the routing group. As the RPA is routed, the system maintains a history of actions. By referring to the history, users can learn what action was taken, by whom, and on what date.

GHR Position Description Process—Enables the routing of the Position Description form for data entry, signature, review and classification. Based on the agency's practices, the user can route the Position Description form to an individual, groupbox, or routing list within the routing group. As the Position Description form is routed, the system maintains a history of actions. By referring to the history, users can learn what action was taken, by whom, and on what date.

GHR Within Grade Increase Process—Enables the automatic processing of Within Grade Increase(WGI) actions without any manual intervention. The default WGI process automatically notifies the Personnel Office of the WGI approval and requires no response. WGI process can be configured during implementation in many ways based on the agency's practices.

Oracle Human Resources

Task Flow Item Type—Oracle Human Resources provides a predefined workflow item type called HR Task Flow that you can use to set up your task flows. The HR Task Flow item type includes a function activity for every HR application window that is allowed to be incorporated into a task flow. You can use these predefined function activities to model a workflow process for each task flow. Moreover, each function activity includes activity attributes that you can set to create button labels and position buttons on its corresponding application window.

The HR Task Flow item type provides you with an alternative to using forms to set up and maintain your task flows. By integrating with Oracle Workflow, you can use the graphical Oracle Workflow Builder to help you design and diagram the sequence of your windows.

Oracle Order Entry

ATO Change Order Management Process—This process is an Oracle Order Entry/Shipping process navigator flow. This process provides you with the ability to streamline the change order process for assemble–to–order (ATO) customers through an easy–to–use, graphical user interface. In addition, it enables you to streamline the change order process across application products and includes complete business cycles.

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 Process—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 Process—Notifies and confirms credit card transactions with a card holder's manager. Oracle Payables initiates this process when the Credit Card Transaction Employee workflow process executes. The Credit Card Transaction Manager workflow process notifies a manager of transactions created by the employee's credit card, and determines if the manager needs to approve the transactions.

Expense Reporting Process—Oracle Payables uses the Expense Reporting workflow process to process the manager approval and accounting review of expense reports entered in Self–Service Expenses. The Expense Reporting process begins when a user submits an expense report, and finishes when an expense report is rejected or approved. If approved, the Expense Reporting process makes the expense report available for the Payables Invoice Import process. The Expense Report process notifies employees at key event points during the manager approval and accounting review process.

Oracle Planning

Planning Exception Message Process—Sends notifications to suppliers, customer contacts, or internal personnel that inform them of planning exceptions and lets the recipients initiate appropriate action to correct the planning exception.

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 Project Manufacturing

Indirect/Capital Project Definition Process—This process is part of the Project Manufacturing Project Definition process navigator flow. This process guides users through all the necessary sequence of steps for setting up an Indirect– or Capital–type project for use in Oracle Project Manufacturing.

Contract Project Definition Process—This process is part of the Project Manufacturing Project Definition process navigator flow. This process guides users through all the necessary sequence of steps for setting up a Contract–type project for use in Oracle Project Manufacturing.

Oracle Process Manufacturing

Quality Control Sample Creation Notification Process—Notifies and prompts a valid user who is associated with certain parameters of transactions such as Organization, Warehouse, or Item, to create samples for quality assurance in the Product Development Module of Oracle Process Manufacturing. Specific inventory transactions in Oracle Process Manufacturing initiate this workflow process. The user can create a quality control sample by invoking the Sample Creation form directly from the notification.

Quality Control Sample Acceptance Process—Spawns detail processes that notify quality control analysts to perform tests on a newly created sample and manages the testing results for final sample acceptance. This workflow is initiated when a sample is created in the Product Development Module of Oracle Process Manufacturing. This workflow is a master process that determines the number of tests to be performed on the sample based on predefined specifications and spawns a matching number of Quality Control Assay Testing detail processes to notify the analysts to perform the tests. The master process waits until all the detail processes complete before sending a notification with the sample disposition to the sample approver. The notification allows the approver to view the results directly from the Result form and enter a final disposition on the sample. The process then completes by notifying the inventory approver of the final sample disposition.

Quality Control Assay Testing Process—Notifies quality control analysts to perform tests on a newly created sample. This process is initiated by the Quality Control Sample Acceptance process. It sends a notification to the analyst who needs to perform the tests. The analyst can respond to the notification by directly opening the Result form from the notification to enter the results of the tests.

Item Activation Process—Notifies an approver to approve an item once it is created in Oracle Process Manufacturing. The item is made inactive until the approver approves it.

Lot Expire and Lot Retest Process—Notifies appropriate roles associated with an item when a lot or sublot of that item expires or is ready for retesting.

Process Manufacturing Intelligence

Process Manufacturing Inventory Turns Process—Sends notifications to the designated responsibilities whenever the actual values of the inventory turn do not fall within the targeted values defined in the Inventory Turn Report. The Inventory Turn Report is part of Process Manufacturing BIS.

Oracle Purchasing

Procurement Workflow—The Procurement Workflow is a lights–out, hands–off transaction processing system that is truly flexible and extensible to all members of your supply chain. It is one of the key enablers in the shift towards more strategic sourcing and procurement activities. It consists of the Document Approval, Automatic Document Creation, Change Orders, Account Generation, Send Notifications, Price/Sales Catalog Notification, and Receipt Confirmation (used only by Self–Service Purchasing) workflow processes.

Document Approval Process—Performs all approval related activities in Oracle Purchasing. These include, but are not limited to, document submission, approval, forwarding, approval notifications, and rejection. This includes the PO Approval workflow process for approving purchase orders and the PO Requisition Approval workflow process for approving requisitions.

Automatic Document Creation Process—Automatically creates standard purchase orders or releases against blanket agreements using approved purchase requisition lines, if the requisition lines have the required sourcing information. This process is also known as the PO Create Documents workflow.

Change Orders Process—Allows you to control which changes require a manual reapproval and which will be automatically reapproved. All

reapproved documents, either manual or automatic, will result in the document revision being incremented. This process is part of the PO Approval workflow.

Send Notifications Process—Looks for documents that are incomplete, rejected, or in need of reapproval, and sends notifications regarding the document's status to the appropriate people. This is also known as the PO Send Notifications for Purchasing Documents workflow.

Price/Sales Catalog Notification Process—Sends a notification to the buyer when the price/sales catalog information sent through the Purchasing Documents Open Interface includes price increases that exceed a price tolerance that you set. This process is also known as the PO Catalog Price Tolerance Exceeded Notifications workflow.

Oracle Receivables

Credit Memo Request Approval Process – Routes a credit memo request for approval using an organization's internal management hierarchy or approval limits defined in Oracle Receivables. If the request is approved, a credit memo is created in Oracle Receivables. Otherwise, the process notifies the requestor with an explanation of why it was rejected.

Oracle Service

Service Request Process—Routes a service request to individuals in the organization for resolution. Customize the process to select and notify service personnel, as well as to transfer and escalate service requests automatically based on your organization's service rules and guidelines.

Service Request Actions and Dispatch Process—Routes a service request action to individuals in the organization for resolution and in addition, notify with instructions, appropriate service personnel who need to be dispatched to a field site. Customize the process to manage, transfer or escalate dispatch requests.

Field Service Dispatch Process—Inserts or updates service request data into the interface table and sends a notification to the field service engineer with dispatch information. This process is used by Oracle Mobile Field Service.

Oracle Support Policy for Predefined Workflows

Oracle Workflow is embedded in Oracle Applications and is used by its modules to automate and streamline business processes. You can use Oracle Workflow Builder to easily modify an existing business process without changing its application's code. Oracle Workflow also allows you to extend your workflow processes as your business rules change and mature.

Before you use Oracle Workflow to customize any predefined workflow process, you should familiarize yourself with the following customization guidelines to ensure standard and safe design and development practices. By following these guidelines, you will be able to supply important information to Oracle Support Services in helping you resolve any issues that arise from your customizations.

Customization Guidelines

1. Verify that all setups have been completed as documented in the Oracle Workflow Guide, and the product–specific User's Guides.
2. Test the unmodified seeded workflow on a test database and ensure that it runs successfully with the setup and data specific to your environment.
3. Refer to the product–specific User's Guide and any documentation update, available on MetaLink, for the specific workflow of interest. These documentation sources specifically mention what should NOT be modified. Oracle Support Services will not support modifications to any object that is specifically documented as not modifiable.
4. Gradually build in customizations step–by–step, and test the customized workflow after each step.
5. When creating PL/SQL procedures, conform to the standard PL/SQL API templates documented in the Oracle Workflow Guide. Be sure to handle exceptions in the event of an error so you can track down the procedure where the error has occurred.
6. Do not implement the customized workflow in production without fully ensuring that it works successfully on a test database, which is a replica of your production setup.

Resolving Customization Issues

If you encounter a problem when customizing a seeded workflow, you should:

- Provide the Support analyst with the modified Workflow definition file, and where possible, identify the exact step where the problem occurred.
- Provide the Support analyst with results of running the unmodified seeded Workflow.

What Is NOT Supported

The following types of customizations are not supported:

1. Modifying a workflow object that has a protection level that is less than 100.
2. Altering a workflow object's protection level if its original protection level is less than 100.
3. Modifying your access level to an unauthorized level of less than 100 for the purpose of modifying workflow objects that are protected at levels less than 100.
4. Customizations that are explicitly documented as being UNSUPPORTED in the seeded workflow's product–specific User's Guide or documentation update notes. This includes modifying processes, attributes, function activities, notifications, lookup types, or messages that are specifically documented as not to be modified.
5. Manual modifications of Workflow tables with a prefix of WF_ or FND_ unless it is documented in the Oracle Workflow Guide or is required by Oracle Support Services.

What Is Supported

The following types of customizations are supported:

1. Any customization that is stated as Required in the seeded workflow's product–specific User's Guide or documentation update notes.
2. Customization examples documented in the product–specific User's Guide or documentation update notes. Any issues that arise are fully supported to resolution, to the extent that the customization example was followed as documented. Any deviation from what is documented amounts to a custom development issue that needs further evaluation. See number 3 below.

3. Customizations that are not explicitly stated as unsupported customizations, required customizations or supported customization examples are supported to the extent that the customer must first isolate the problem following the Customization Guidelines discussed earlier. If upon evaluation, Oracle Support Services deems that the isolated problem stems from an Oracle product, Oracle will supply a solution. Otherwise, it is the responsibility of the customer to correct the custom development issue.

Glossary

Access Level A numeric value ranging from 0 to 1000. Every workflow user operates at a specific access level. The access level defines whether the user can modify certain workflow data. You can only modify data that is protected at a level equal to or higher than your access level.

Activity A unit of work performed during a business process.

Activity Attribute A parameter that has been externalized for a function activity that controls how the function activity operates. You define an activity attribute by displaying the activity's Attributes properties page in the Activities window. You assign a value to an activity attribute by displaying the activity node's Attribute Values properties page in the Process window.

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.

External Functions Programs that are executed outside of Oracle8.

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 predefine item type attribute as a message attribute. Defined as a 'Send' source, a message attribute gets replaced with a runtime value when the message is sent. Defined as a 'Respond' source, a message attribute prompts a user for a response when the message is sent.

Node An instance of an activity in a process diagram as shown in the Process window.

Notification An instance of a message delivered to a user.

Notification Activity A unit of work that requires human intervention. A notification activity sends a message to a user containing the information necessary to complete the work.

Notification Mailer A 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

Symbols

&#NID, 4 – 12, 4 – 13, 4 – 14, 4 – 27

A

AbortProcess( ), 8 – 28
Access Level, 2 – 71
  default, 2 – 74
Access level indicator, 4 – 16
Access property page, 4 – 16
Access protection
  See also Access level; Protection level
  preserving customizations, 4 – 17
AccessCheck( ), 8 – 176
ACCOUNT parameter, 2 – 48
ACTID, 7 – 4, 7 – 9
Activities, 3 – 9, 4 – 37
  accessing from different data stores, 5 – 6, 6 – 2
  Concurrent Manager, 6 – 16
  copy, 4 – 49
  cost, 4 – 40
  create, 4 – 41, 4 – 43, 4 – 46
  deferred, 4 – 40
  effective date, 4 – 48
  error process, 4 – 48
  for an error process, 6 – 20
  function, 4 – 37, 4 – 39
  icons, 2 – 69, 4 – 42, 4 – 44, 4 – 47, 4 – 51
  in a loop, 4 – 48
  in the Notify Approver subprocess, 13 – 20
  in the Requisition process, 13 – 14
  joining branches, 5 – 4
  notification, 4 – 37, 4 – 38
  optional details, 4 – 48
  process, 4 – 37, 4 – 39
  processing cost, 8 – 6
  result type, 4 – 41, 4 – 44, 4 – 47
  Standard, 4 – 37, 6 – 2
  statuses, 8 – 3
  System: Error, 4 – 37
  timing out, 5 – 9
  version number, 4 – 49
Activities( ), 8 – 71
Activity attributes
  See also Function activity attributes
  setting values for, 5 – 11
Activity nodes
  in the Notify Approver subprocess, 13 – 20
  in the Requisition process, 13 – 14
Ad hoc users and roles, 5 – 19
  APIs, 8 – 77
AddAttr( ), 8 – 163
AddItemAttr( ), 8 – 35
AddUsersToAdHocRole( ), 8 – 94
AdHocDirectory( ), 8 – 75
Administrator privileges, 2 – 14
Advanced Queues integration, 8 – 113
ALLOW_FORWARDED_RESPONSE parameter, 2 – 49
And activity, 6 – 2
Any transitions, 5 – 2
APIs, 8 – 3
AQ message payload, 8 – 114
Arrows, 5 – 2

Assign activity, 6 – 14
AssignActivity( ), 8 – 52
Attachments, DM documents, 10 – 31
Attribute, token substitution, 4 – 36
Attribute types
  attribute, 4 – 10
  date, 4 – 10, 4 – 30
  document, 4 – 10, 4 – 13, 4 – 31
  form, 4 – 10, 4 – 12, 4 – 30
  lookup, 4 – 10, 4 – 30
  number, 4 – 9, 4 – 30
  role, 4 – 10, 4 – 31
  text, 4 – 9, 4 – 30
  URL, 4 – 10, 4 – 11, 4 – 30
Attribute–type attributes, 4 – 4
Attributes
  copy, 4 – 15
  type, 4 – 3, 4 – 9, 4 – 30
Automatic Notification Handler, 10 – 24
Automatic responses, 10 – 24
Automatic routing, 10 – 24

B

Background engine, scripts, 14 – 5
Background Engines
  about, 2 – 34
  scripts, 14 – 4
  starting, 2 – 34
  submitting, 2 – 35
Background( ), 8 – 33
BeginActivity( ), 8 – 47
Block activity, 6 – 5

C

Callback functions, 7 – 7
  command, 7 – 9
  for item types, 4 – 5
Cancel( ), 8 – 156
CancelGroup( ), 8 – 157
Checking
  activity versions, 14 – 14
  background engines, 14 – 5
  directory service data model, 14 – 8
  foreign/primary key references, 14 – 10
  workflow data model, 14 – 13
CLEAR( ), 8 – 59
ClearMsgStack( ), 8 – 129
Close( ), 8 – 162
Compare Date activity, 6 – 3
Compare Execution Time activity, 6 – 3
Compare Number activity, 6 – 3
Compare Text activity, 6 – 3
Comparison activities, 6 – 3
CompleteActivity( ), 8 – 49
Concurrent Manager activities, 6 – 16
Concurrent Manager Functions item type, 6 – 16
Concurrent program, FNDWFPR, 14 – 4
Concurrent programs
  Notification Mailer, 2 – 38, 2 – 46
  Purge Obsolete Workflow Runtime Data, 8 – 76
  Workflow Background Process, 2 – 35
  Workflow Definitions Loader, 2 – 78
  Workflow Resource Generator, 8 – 62
CONNECT parameter, 2 – 48
Content–attached checkbox, 4 – 32
CONTEXT( ), 8 – 65
Continue Flow activity, 6 – 13
Coordinating master/detail activities, 6 – 12
Cost threshold, 4 – 40
CreateAdHocRole( ), 8 – 92
CreateAdHocUser( ), 8 – 90
CreateForkProcess(), 8 – 30
CreateMsg( ), 8 – 130
CreateProcess( ), 8 – 16
CurrentUser( ), 8 – 83
Custom logos, in the Notification Web page, 2 – 68
Customization Level, 2 – 74
  for activities, 4 – 7, 4 – 11, 4 – 19, 4 – 27, 4 – 43, 4 – 45, 4 – 47, 4 – 51

D

Date–type attributes, 4 – 3
DEBUG parameter, 2 – 51
Default Error Process, 6 – 22
Default transitions, 5 – 2
DEFAULT_ERROR, 6 – 22
Defer Thread activity, 6 – 6
Deferred activities, 2 – 34, 4 – 40
Deferred processing, 2 – 34, 8 – 6
DeferredQueue function, 8 – 126
Delete
  all workflow data, 14 – 11
  data for an item type, 14 – 12
  item type attributes, 14 – 11
  runtime data for an item type, 14 – 12
  workflow status information, 14 – 12
Demonstration, directory service, 13 – 6
DequeueEventDetail( ), 8 – 120
DequeueOutbound( ), 8 – 117
Detail Notification web page, 10 – 18
Detail process, 6 – 12
Detail Survey process
  activities, 13 – 46
  summary, 13 – 45
Diagram arrows, 5 – 2
Direct Response E–mail, 10 – 3
DIRECT_RESPONSE parameter, 2 – 49
Directory repository, 2 – 17
Directory Service
  in Navigator tree, 3 – 3
  view from Builder, 5 – 21
Directory services, 2 – 17
  checking the data model, 2 – 22, 14 – 8
  integrating with local workflow users, 2 – 24
  integrating with native Oracle users, 2 – 23
  integrating with Oracle HR, 2 – 22
Directory Services APIs, 8 – 77
DISCARD parameter, 2 – 51
DM documents, 10 – 31
Document integration, 4 – 3, 4 – 10, 4 – 31, 7 – 11
Document Management, item type, 13 – 48
Document management, 10 – 31
  home, 10 – 35
  nodes, 2 – 31, 10 – 35
  notification integration, 10 – 31
  Oracle Workflow toolbar, 10 – 31
  repositories, 2 – 31
  supported functions, 10 – 33
Document Management APIs, 8 – 134
Document management integration, 4 – 3, 4 – 5
Document Management Transport Window, 10 – 35
Document Nodes, 2 – 31
  web page, 2 – 31
Document Review process, 13 – 48
  activities, 13 – 51
  summary, 13 – 49
Document–type attributes, 4 – 3
Documents, 4 – 5
Dynamic priority, 5 – 10
Dynamic timeouts, 5 – 10

E

E–mail notifications, 1 – 4, 2 – 38
  and HTML attachments, 2 – 2
  example direct response instructions, 10 – 7
  modifying mail templates, 2 – 56
  requirements, 2 – 2
  summaries, 10 – 23
  templates for, 2 – 38, 10 – 3
  with HTML attachments, 10 – 2
Edit menu, A – 3
Effective date, 3 – 16
Effective dates, 3 – 13, 3 – 16, 4 – 48, 8 – 9
Effectivity, dates of, 3 – 7
END activities, 5 – 4
End Activity, 6 – 9
Engine thresholds, 2 – 36
EnqueueInbound( ), 8 – 116
Environment variables
  WF_ACCESS_LEVEL, 2 – 71, 2 – 75
  WF_RESOURCES, 2 – 30
Error activities, 6 – 20
Error Check process, 13 – 53
  activities, 13 – 56

summary, 13 – 55
Error handling, 8 – 53
Error process, 4 – 48, 6 – 20
Error Processing, 8 – 7
Errored activities, retrying, 14 – 11
Example function activity
Select Approver, 13 – 25
Verify Authority, 13 – 28
Example process, Requisition, 13 – 4
Execute Concurrent Program activity, 6 – 16
External document integration, 4 – 5

F

FAILCOMMAND parameter, 2 – 51
File menu, A – 2
Find Notifications web page, 10 – 14
FNDWFPR, 8 – 76
concurrent program, 14 – 4
Fonts
modifying, 5 – 16
setting, 5 – 16
Forced synchronous processes, 8 – 11
Form–type attributes, 4 – 3
FORWARD mode, 8 – 10
Forward( ), 8 – 143, 8 – 154
Frame target, URL attributes, 4 – 31
FROM parameter, 2 – 48
FUNCMODE, 7 – 4, 7 – 5
Function activities, 4 – 39
create, 4 – 43
standard PL/SQL API, 7 – 2
Function activity attributes, 4 – 8, 4 – 45
Functions, 3 – 10
See also PL/SQL procedures

G

Get Monitor URL activity, 6 – 15
GET_ERROR( ), 8 – 60
get_launch_attach_url( ), 8 – 136
get_launch_document_url( ), 8 – 135
get_open_dm_select_window( ), 8 – 137, 8 – 138
get_pref( ), 8 – 100
GetActivityAttrDate( ), 8 – 46
GetActivityAttrInfo( ), 8 – 45
GetActivityAttrNumber( ), 8 – 46
GetActivityAttrText( ), 8 – 46
GetActivityLabel( ), 8 – 20
GetAdvancedEnvelopeURL( ), 8 – 106
GetAttrDate( ), 8 – 170
GetAttrDoc( ), 8 – 171
GetAttrInfo( ), 8 – 166
GetAttrNumber( ), 8 – 170
GetAttrText( ), 8 – 170
GetBody( ), 8 – 173
GetDiagramURL( ), 8 – 102
GetEnvelopeURL( ), 8 – 104
GetInfo( ), 8 – 167
GetItemAttrDate( ), 8 – 41
GetItemAttrDocument( ), 8 – 42
getItemAttributes( ), 8 – 43
GetItemAttrInfo( ), 8 – 44
GetItemAttrNumber( ), 8 – 41
GetItemAttrText( ), 8 – 41
getItemTypes( ), 8 – 40
GetItemUserKey( ), 8 – 19
GetMessageHandle( ), 8 – 125
GetNotificationAttributes( ), 8 – 179
GetNotifications( ), 8 – 178
getProcessStatus( ), 8 – 57
GetRoleDisplayName( ), 8 – 87
GetRoleInfo( ), 8 – 80
GetRoleInfo2( ), 8 – 81
GetRoleName( ), 8 – 86
GetRoleUsers( ), 8 – 78
GetShortBody( ), 8 – 174
GetShortText( ), 8 – 169
GetSubject( ), 8 – 172
GetText( ), 8 – 168
GetUserName( ), 8 – 85
GetUserRoles( ), 8 – 79

Global Preferences, web page, 2 – 12
Global variables, 4 – 2

H

HandleError( ), 8 – 53
Hardware requirements, 2 – 2
Help menu, A – 6
Hidden item types, 3 – 4
Home page, 9 – 2
HTMLAGENT parameter, 2 – 51
HTMLDESC parameter, 2 – 51
HTMLFILE parameter, 2 – 51
HTMLTYPE parameter, 2 – 51

I

Icons, 2 – 69
  viewing, 4 – 42, 4 – 45, 4 – 47
IDLE parameter, 2 – 50
InboundQueue function, 8 – 127
Initiating a workflow process, 13 – 7, 13 – 35
Internal names
  updating activity, 14 – 5
  updating activity attributes, 14 – 6
  updating item attributes, 14 – 6
  updating item types, 14 – 6
  updating lookup codes, 14 – 7
  updating lookup types, 14 – 7
  updating message attributes, 14 – 8
  updating messages, 14 – 7
Introduction, features of manual, xii
IsPerformer( ), 8 – 82
Item attributes, external document integration, 4 – 5
Item type attributes, 4 – 2, 4 – 8, 8 – 9
  Requisition, 13 – 11
Item types, 3 – 9, 4 – 2
  callback function, 4 – 5
  Concurrent Manager Functions, 6 – 16
  context reset, 7 – 7
  copy, 4 – 14
  creation, 4 – 6
  loading, 3 – 12
  persistence type, 4 – 4
  Requisition, 13 – 11
  saving, 3 – 12
  selector functions, 4 – 4, 7 – 7
  Standard, 6 – 2
  System: Error, 6 – 20
  System: Mailer, 2 – 56
ITEMKEY, 7 – 3, 7 – 9
Items( ), 8 – 70
ItemStatus( ), 8 – 56
ITEMTYPE, 7 – 3, 7 – 8

J

Java APIs, 8 – 4
Java interface, 8 – 4
Java monitor tool, 11 – 2
JavaScript, support in a Web browser, 2 – 2
Joining activities, 5 – 4

L

Launch Process activity, 6 – 6
LaunchProcess( ), 8 – 24
List of values, in a web interface, 10 – 22
Load balancing, 6 – 9
Loading item types, 3 – 12
LOG parameter, 2 – 50
Lookup codes, copy, 4 – 21
Lookup types, 3 – 9, 4 – 18
  copy, 4 – 21
  creation, 4 – 19
Lookup–type attributes, 4 – 3
Loop Counter activity, 6 – 7
Loop Reset, 5 – 3
Loops, 4 – 48, 7 – 5, 8 – 7

M

Master process, 6 – 12

Master/Detail coordination activities, 6 – 12
notes on usage, 6 – 14
Menus, Oracle Workflow Builder, A – 2
Message attributes, 4 – 22, 4 – 23, 4 – 28, 4 – 29, 4 – 45, 13 – 31
for Workflow Cancelled Mail message, 2 – 61
for Workflow Closed Mail message, 2 – 64
for Workflow Invalid Mail message, 2 – 62
for Workflow Open FYI Mail message, 2 – 61
for Workflow Open Mail (Direct) message, 2 – 59
for Workflow Open Mail (Templated) message, 2 – 57
for Workflow Summary Mail message, 2 – 65
for Workflow Warning Mail message, 2 – 66
Respond, 4 – 23, 4 – 29, 4 – 33
Send, 4 – 23, 4 – 29
source, 4 – 23, 4 – 29
Message templates, for E–mail notifications, 2 – 56
Messages, 3 – 9
body, 4 – 26, 13 – 31
copy, 4 – 36
creation, 4 – 24
overriding default priority, 5 – 10
subject, 4 – 25, 13 – 31
viewing, 13 – 32
Messages window, 4 – 22
MIME support, 2 – 39
Monitoring
Workflow Monitor, 11 – 2
workitems, 1 – 4
Multilingual support, 14 – 4, 14 – 9

N

Naming conventions, PL/SQL stored procedures, 13 – 14
Navigator Toolbar, A – 7
Navigator tree, finding objects in, 3 – 6
NLS support
in a web session, 2 – 26
in E–mail notifications, 2 – 26
in Oracle Workflow Builder, 2 – 25
Node activities, dynamic priority, 5 – 10
NODE parameter, 2 – 48
Nodes
adding to a process, 5 – 5
start and end, 5 – 7
NOOP activity, 6 – 7
Notification, status, 14 – 9
Notification access keys, 10 – 3
Notification activities, 4 – 38
coupling with custom functions, 4 – 42, 8 – 10
create, 4 – 41
Notify Requisition Approval Required, 13 – 31
Notification APIs, 8 – 141, 8 – 146
Notification functions, 4 – 42, 8 – 10
Notification ID token, 4 – 12, 4 – 13, 4 – 14, 4 – 27
Notification IDs, 10 – 3
Notification Mailer
about, 2 – 38
configuration file, 2 – 47
MIME support, 2 – 39
notification preference, 2 – 39
response processing, 2 – 54
script to restart, 2 – 54
shutdown, 2 – 38
starting, 2 – 46
starting for MAPI–compliant applications, 2 – 46
starting for Oracle Office, 2 – 45
starting for UNIX Sendmail, 2 – 45
Notification method, 10 – 2
Notification preference, 9 – 6
Notification preferences, 2 – 16, 2 – 39
Notification summaries, via E–mail, 10 – 23
Notification System, 2 – 38, 8 – 141
Notification templates, for E–mail notifications, 2 – 56
Notification Web page, 1 – 4
reassigning notifications, 10 – 21
Notifications, 10 – 2
attaching a document, 10 – 34
dependence on directory services, 10 – 2
forwarding, 8 – 143

HTML–formatted E–mail, 10 – 10
identifying the responder, 8 – 158
load balancing, 6 – 9
plain text E–mail using direct response, 10 – 6
plain text E–mail using templated response, 10 – 4
plain text E–mail with attachments, 10 – 11
reassign in Notification Web page, 10 – 21
reassign via E–mail, 10 – 12
responding with Notification Web page, 10 – 21
timed out, 8 – 144
transferring, 8 – 144
via E–mail, 2 – 38, 10 – 2
via Notification Web page, 10 – 13
viewing attached documents, 10 – 31
Notifications Worklist. *See* Worklist web page
Notifications( ), 8 – 72
Notify activity, 6 – 10
Notify Approver, example notification activities, 13 – 30
Notify Approver subprocess, summary, 13 – 19
Notify Requisition Approval Required, 13 – 31
Number–type attributes, 4 – 3

O

On Revisit, 8 – 8
OpenNotificationsExist( ), 8 – 161
Or activity, 6 – 2
Oracle Net8, 2 – 2
Oracle Office, 2 – 44
required folders, 2 – 52
Oracle WebServer
identifying the workflow web agent, 2 – 14
modifying workflow web templates, 2 – 68
setting up the Workflow Monitor, 2 – 68
Oracle Workflow, implementation issues, 2 – 4
Oracle Workflow Builder, 1 – 3
Loader functionality, 3 – 15
overview, 3 – 2
requirements, 2 – 2
save modes, 3 – 15, 4 – 16
starting from command line, 3 – 17
Oracle Workflow home page, 9 – 2
Oracle Workflow views, 8 – 108
Oracle8 Advanced Queues integration, 8 – 113
OutboundQueue function, 8 – 128

P

Partitioning Workflow tables, 2 – 10
Payload, for Advanced Queues messages, 8 – 114
Periodic Alert, item type, 13 – 53
Persistence, 4 – 4
PL/SQL, 1 – 3
document, 7 – 11
PL/SQL APIs
for a 'PL/SQL' document, 7 – 11
for a selector or callback function, 7 – 7
for function activities, 7 – 2
PL/SQL documents, 4 – 5
PL/SQL stored procedures
creating, 13 – 14
naming conventions, 13 – 14
scripts, 13 – 14
Post–notification functions, 4 – 38
Preferred notification method, 10 – 2
Preserving customizations, for an activity, 4 – 17
Process activities, 4 – 39
create, 4 – 46
Process definition, modifying, 3 – 11
Process diagram
adding nodes, 5 – 5
drawing, 5 – 2, 5 – 5
Process rollback, 8 – 53
Process window, 5 – 2
editing, 5 – 2
Process Window Toolbar, A – 8
Processes
activity transitions, 5 – 2
copying to clipboard, 5 – 15
creation, 3 – 7
editing, 3 – 10, 3 – 12

loops, 7 – 5, 8 – 7
overview, 5 – 14
printing, 5 – 15
starting, 5 – 4
verify, 5 – 15
ProcessInboundQueue( ), 8 – 124
Product Survey, web page, 13 – 35
Product Survey item type, 13 – 37
Product Survey process, 13 – 33
initiating, 13 – 35
installing, 13 – 34
Protection level, 2 – 72
reset, 14 – 9
Protection level locking. *See* Access protection
Purge, runtime data, 14 – 4
Purge Obsolete Workflow Runtime Data concurrent program, 8 – 76
PurgeEvent( ), 8 – 122
PurgeItemType( ), 8 – 123

R

RAISE( ), 8 – 62
Reassign notifications
in Notification Web page, 10 – 21
via E–mail, 10 – 12
Reassign web page, 10 – 21
RemoveUsersFromAdHocRole, 8 – 99
REPLYTO parameter, 2 – 51
Requirements, hardware and software, 2 – 2
Requisition, data model, 13 – 5
Requisition Demonstration, web page, 13 – 7
Requisition process, 13 – 4
example function activities, 13 – 25
initiating, 13 – 7
installing, 13 – 5
summary, 13 – 12
Reset process. *See* Rollback
Respond attributes, 2 – 57, 2 – 59
RESPOND mode, 8 – 10
Respond to notification
HTML–formatted E–mail, 10 – 10
plain text E–mail using direct response, 10 – 6
plain text E–mail using templated response, 10 – 4
plain text E–mail with attachments, 10 – 11
via Notification Web page, 10 – 13
Respond( ), 8 – 143, 8 – 158
Responder, 8 – 158
Responder( ), 8 – 159
Response methods, direct vs. templated, 2 – 49
Response processing, by Notification Mailer, 2 – 54
Responses, processing, 8 – 143
RESULT, 7 – 4, 7 – 9
Result type
for activities, 4 – 41, 4 – 44, 4 – 47
for voting activities, 4 – 51
ResumeProcess( ), 8 – 27
Retry Error, 6 – 25
RETRY_ONLY, 6 – 25
Role
administrator, 2 – 14
property page, 5 – 21
Role Resolution activity, 6 – 9
Role–type attributes, 4 – 4
Roles, 5 – 19
ad hoc, 5 – 19
loading into the Workflow Builder, 5 – 20
tab page, 5 – 19
view from Builder, 5 – 21
Rollback, of process, 8 – 53
Routing, automatic, 10 – 24
Routing rules
deleting, 10 – 30
for a role, 10 – 25
listing, 10 – 25
overriding, 10 – 29
updating, 10 – 30

S

Sample workflow processes, 13 – 2
Savepoints, 7 – 2, 8 – 4
Select Approver function activity, 13 – 25

Selector functions, 4 – 4, 7 – 7
Send( ), 8 – 141, 8 – 148
SendGroup( ), 8 – 141, 8 – 152
set_document_id_html( ), 8 – 139
SetAdHocRoleAttr( ), 8 – 98
SetAdHocRoleExpiration( ), 8 – 96
SetAdHocRoleStatus( ), 8 – 89
SetAdHocUserAttr( ), 8 – 97
SetAdHocUserExpiration( ), 8 – 95
SetAdHocUserStatus( ), 8 – 88
SetAttrDate( ), 8 – 164
SetAttrNumber( ), 8 – 164
SetAttrText( ), 8 – 164
SetItemAttrDate( ), 8 – 36
SetItemAttrDocument( ), 8 – 38
SetItemAttrNumber( ), 8 – 36
SetItemAttrText( ), 8 – 36
SetItemOwner( ), 8 – 21
SetItemParent API, 6 – 12
SetItemParent( ), 8 – 55
SetItemUserKey( ), 8 – 18
SetMsgAttr( ), 8 – 132
SetMsgResult( ), 8 – 133
Shortcuts, 5 – 17
Shutdown files, 2 – 50
SHUTDOWN parameter, 2 – 50
Software requirements, 2 – 2
Standard activities, 6 – 2
Standard APIs
  for "PL/SQL documents", 7 – 11
  for function activities, 7 – 2
  for selector/callback functions, 7 – 7
Standard error process, 6 – 20
Standard item type, 6 – 2
START activities, 5 – 4
Start activity, 6 – 9
StartForkProcess(), 8 – 32
StartProcess function, for sample Requisition process, 13 – 22
StartProcess( ), 8 – 22
Status report
  developer, 14 – 13
  end user, 14 – 13
Submit Concurrent Program activity, 6 – 17
Subprocesses, timing out, 5 – 9
SUMMARYONLY parameter, 2 – 49
Survey – Master/Detail process
  activities, 13 – 43
  summary, 13 – 41
Survey – Single Process, activities, 13 – 40
Survey – Single process, summary, 13 – 38
SuspendProcess( ), 8 – 26
Synchronous processing, 8 – 11
System: Error item type, 6 – 20
System: Mailer item type, 2 – 56

T

Tag files, 2 – 52
TAGFILE parameter, 2 – 52
TCP/IP drivers, 2 – 2
Templated Response E–mail, 10 – 3
Test harness, 12 – 2
TEST_ADDRESS parameter, 2 – 51
TestContext( ), 8 – 175
Text–type attributes, 4 – 3
Timed out processes, 2 – 34
Timeout transitions, 5 – 2, 5 – 3
Timeouts, 5 – 9
  dynamic, 5 – 10
Token substitution
  attributes, 4 – 36
  of document–type message attributes, 4 – 13
TOKEN( ), 8 – 61
Toolbars, Oracle Workflow Builder, A – 7
Total( ), 8 – 73
TotalPERM( ), 8 – 74
TRANSFER mode, 8 – 10
Transfer( ), 8 – 144, 8 – 155
Transitions, 5 – 2
  Any, 5 – 2
  creating, 5 – 12

Default, 5 – 2
editing, 5 – 12
Timeout, 5 – 2
TRANSLATE( ), 8 – 67
Translation, 2 – 25

U

UNIX Sendmail, 2 – 44
UNPROCESS parameter, 2 – 52
Upgrading workflow definitions, 8 – 9
URL attributes, frame target, 4 – 31
URL message attributes, attached vs embedded, 4 – 32
URL–type attributes, 4 – 3
URLs
for Find Notifications Routing Rules web page, 10 – 25
for Find Notifications web page, 10 – 14
for Find Processes web page, 11 – 8
for Notifications Routing Rules web page, 10 – 25
for Oracle Workflow home page, 9 – 2
for Product Survey web page, 13 – 35
for Requisition Demonstration web page, 13 – 8
for the Workflow Monitor, 11 – 7
for Worklist web page, 10 – 13
User Defined Alert Action process
activities, 13 – 60
summary, 13 – 59
User Preferences, web page, 9 – 4
User preferences, 2 – 12
document management home, 2 – 16, 9 – 6
Language and Territory, 2 – 15
language and territory, 9 – 5
notification preference, 9 – 6
notification preferences, 2 – 16
UserActive( ), 8 – 84
Users, ad hoc, 5 – 19

V

Vacation forwarding, 10 – 24
Verify Authority function activity, 13 – 28
Version, 8 – 9, 14 – 14
of Oracle Workflow, 2 – 6
Version compatibility, 2 – 6
Version number, for activities, 4 – 49
Versioning, 3 – 7
View menu, A – 4
View notifications
E–mail summary, 10 – 23
electronic mail, 10 – 2
Notification Web page, 10 – 13
web browser, 10 – 13
Views, Oracle Workflow, 8 – 108
Vote Yes/No activity, 6 – 11
VoteCount( ), 8 – 160
Voting activities
processing, 8 – 144
result type, 4 – 51
Voting activity, 4 – 50

W

Wait activity, 6 – 4
Wait for Concurrent Program activity, 6 – 18
Wait for Flow activity, 6 – 12
Web agent, for Oracle Workflow, 2 – 14
Web home page, 9 – 2
Web notifications, requirements, 2 – 3
WF_ACCESS_LEVEL, 2 – 71, 2 – 75
WF_ENGINE.AbortProcess, 7 – 5
WF_ENGINE.BACKROUND, 2 – 34
WF_ITEM_ACTIVITY_STATUSES_V, 8 – 108
WF_ITEMS_V, 8 – 112
WF_LANGUAGES view, 2 – 25
WF_LOCAL_* tables, 2 – 17
WF_NOTIFICATION_ATTR_RESP_V, 8 – 110
WF_REQDEMO.SelectApprover, 13 – 25
WF_REQDEMO.StartProcess, 13 – 7
WF_REQDEMO.VerifyAuthority, 13 – 17, 13 – 28

WF_RESOURCES, environment variable, 2 – 30
WF_ROLES, view, 2 – 20
WF_RUNNABLE_PROCESSES_V, 8 – 111
WF_USER_ROLES, view, 2 – 21
WF_USERS, view, 2 – 17
Wfbkg.sql, 14 – 4
Wfbkgchk.sql, 14 – 5
Wfchact.sql, 14 – 5
Wfchacta.sql, 14 – 6
Wfchita.sql, 14 – 6
Wfchitt.sql, 14 – 6
Wfchluc.sql, 14 – 7
Wfchlut.sql, 14 – 7
Wfchmsg.sql, 14 – 7
Wfchmsga.sql, 14 – 8
Wfdirchk.sql, 14 – 8
wfdircsv.sql, 2 – 24
wfdirhrv.sql, 2 – 22
wfdirouv.sql, 2 – 23
WFLOAD, 2 – 78
wfload, 2 – 76
wfmail.cfg, 2 – 47
WFNLADD.sql, 14 – 4
WFNLENA.sql, 14 – 9
Wfntfsh.sql, 14 – 9
Wfprot.sql, 14 – 9
Wfqclean.sql, 14 – 10
Wfrefchk.sql, 14 – 10
Wfretry.sql, 14 – 11
Wfrmall.sql, 14 – 11
Wfrmita.sql, 14 – 11
Wfrmitms.sql, 14 – 12
Wfrmitt.sql, 14 – 12
Wfrmtype.sql, 14 – 12
Wfrun.sql, 14 – 13
WFRUND.SQL, 13 – 7
Wfstat.sql, 14 – 13
Wfstatus.sql, 14 – 13
Wfstdchk.sql, 14 – 13
Wfupartb.sql, 2 – 10
Wfver.sql, 14 – 14
Wfverchk.sql, 14 – 14
Wfverupd.sql, 14 – 14
Windows menu, A – 6
WorkCount( ), 8 – 177
Workflow administrator, 2 – 14
Workflow Builder menus, A – 2
Workflow Cancelled Mail message template, 2 – 61
Workflow Closed Mail message template, 2 – 64
Workflow Core APIs, 8 – 58
Workflow definitions
  loading, 1 – 3
  source control, 3 – 12
  testing, 12 – 2
  transferring, 2 – 76
Workflow Definitions Loader, 1 – 3, 2 – 76
  concurrent program, 2 – 78
Workflow Demonstrations home page, 13 – 2
Workflow Designer. *See* Oracle Workflow Builder
Workflow diagrams, displaying, 13 – 3
Workflow Directory Service APIs, 8 – 77
Workflow Engine, 1 – 3
  calling after activity completion, 8 – 6
  calling for activity initiation, 8 – 3
  CANCEL mode, 8 – 8
  core APIs, 8 – 58, 8 – 68
  cost threshold, 4 – 40
  deferred activities, 8 – 6
  directory services, 8 – 77
  error processing, 8 – 7
  Java APIs, 8 – 4, 8 – 14
  looping, 8 – 7
  master/detail processes, 8 – 55
  PL/SQL APIs, 8 – 14
  requirements, 2 – 2
  RUN mode, 8 – 8
  threshold cost, 2 – 36, 8 – 6
Workflow Engine APIs, 8 – 3
Workflow Invalid Mail message template, 2 – 62
Workflow Monitor, 11 – 2
  Administration buttons, 11 – 6

Detail Tab window, 11 – 4
Process Diagram window, 11 – 3
Process title, 11 – 3
setup, 2 – 68
Workflow Monitor APIs, 8 – 101
Workflow Notification APIs. *See* Notification APIs
Workflow Open Mail (Direct) message template, 2 – 58
Workflow Open Mail (Templated) message template, 2 – 56
Workflow Open Mail message template, 2 – 60
Workflow Preferences API, 8 – 100
Workflow processes
creating and starting, 14 – 13
monitoring, 11 – 2
samples, 13 – 2
Workflow Purge APIs, 8 – 68
Workflow Queue APIs, 8 – 113
Workflow queues, cleaning, 14 – 10
Workflow Resource Generator, 8 – 62
concurrent program, 8 – 63
Workflow roles, 2 – 17
Workflow Summary Mail message template, 2 – 65
Workflow users, 2 – 17
Workflow Views, 8 – 108
Workflow Warning Mail message template, 2 – 66
Workflow web pages, modifying template, 2 – 68
Workitems. *See* Items
Worklist web page, 10 – 17
WriteMsg( ), 8 – 131

Reader's Comment Form

Oracle Workflow Guide
A75397–01

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.