

# Oracle<sup>®</sup> Workflow Guide

RELEASE 2.6.2  
VOLUME 1  
March 2002

ORACLE<sup>®</sup>

Oracle Workflow Guide Volume 1, Release 2.6.2

The part number for this volume is A95276–03. To reorder this book, please use the set part number, A95265–03.

**Copyright © 1996, 2002 Oracle Corporation. All rights reserved.**

Primary Authors: Siu Chang, Clara Jaeckel

Major Contributors: George Buzsaki, John Cordes, Mark Craig, Kevin Hudson, George Kellner, David Lam, Jin Liu, Kenneth Ma, 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 laws. Reverse engineering, disassembly or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. 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 NOTICE**

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 Express, JInitiator, Oracle8, Oracle8i, Oracle9i, Oracle*MetaLink*, Oracle Press, Oracle Store, PL/SQL, and SQL\*Plus are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.



# Contents

## VOLUME 1

- Preface** ..... **xix**
- Audience for This Guide ..... xx
- How To Use This Guide ..... xx
- Documentation Accessibility ..... xxi
- Other Information Sources ..... xxii
- Online Documentation ..... xxii
- Related User’s Guides ..... xxiii
- Guides Related to All Products ..... xxiii
- User Guides Related to This Product ..... xxiv
- Installation and System Administration ..... xxv
- Other Implementation Documentation ..... xxvi
- Training and Support ..... xxviii
- Do Not Use Database Tools to Modify Oracle  
    Applications Data ..... xxix
- About Oracle ..... xxix
- Your Feedback ..... xxx

## Chapter 1

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

<b>Chapter 2</b>	<b>Setting Up Oracle Workflow</b> . . . . .	<b>2 – 1</b>
	Oracle Workflow Hardware and Software Requirements . . . . .	2 – 2
	Overview of Setting Up . . . . .	2 – 6
	Overview of Required Setup Steps for the Standalone Version of Oracle Workflow . . . . .	2 – 6
	Overview of Required Setup Steps for the Version of Oracle Workflow Embedded in Oracle Applications . . . . .	2 – 7
	Optional Setup Steps . . . . .	2 – 7
	Other Workflow Features . . . . .	2 – 8
	Identifying the Version of Your Oracle Workflow Server . . . . .	2 – 9
	Setup Flowchart . . . . .	2 – 10
	Setup Checklist . . . . .	2 – 11
	Setup Steps . . . . .	2 – 12
	Overview of Oracle Workflow Access Protection . . . . .	2 – 101
	Setting Up a Default Access Level . . . . .	2 – 105
	Using the Workflow Definitions Loader . . . . .	2 – 107
	Using the Workflow XML Loader . . . . .	2 – 112
<b>Chapter 3</b>	<b>Defining a Workflow Process</b> . . . . .	<b>3 – 1</b>
	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
	Using Oracle Workflow Builder with Different Server Versions . . . . .	3 – 21
	Item Type Definition Web Page . . . . .	3 – 24
<b>Chapter 4</b>	<b>Defining Workflow Process Components</b> . . . . .	<b>4 – 1</b>
	Workflow Process Components . . . . .	4 – 2
	Item Types . . . . .	4 – 2
	Allowing Access to an Object . . . . .	4 – 17
	Lookup Types . . . . .	4 – 19
	Messages . . . . .	4 – 23
	Activities . . . . .	4 – 42
	Voting Activity . . . . .	4 – 61
	Deleting Objects in Oracle Workflow Builder . . . . .	4 – 68
	Modifying Objects in Oracle Workflow Builder . . . . .	4 – 69
	Workflow Objects That Support Versioning . . . . .	4 – 70

Workflow Objects That Do Not Support Versioning . . . . .	4–71
---	------

<b>Chapter 5</b>	<b>Defining a Workflow Process Diagram . . . . .</b>	<b>5–1</b>
	Process Window . . . . .	5–2
	Modifying Fonts in Oracle Workflow Builder . . . . .	5–21
	Creating a Shortcut Icon for a Workflow Process . . . . .	5–22
	Roles . . . . .	5–24

<b>Chapter 6</b>	<b>Predefined Workflow Activities . . . . .</b>	<b>6–1</b>
	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 . . . . .	6–8
	End Activity . . . . .	6–8
	Role Resolution Activity . . . . .	6–9
	Notify Activity . . . . .	6–9
	Vote Yes/No Activity . . . . .	6–10
	Master/Detail Coordination Activities . . . . .	6–11
	Wait for Flow Activity . . . . .	6–12
	Continue Flow Activity . . . . .	6–12
	Assign Activity . . . . .	6–14
	Get Monitor URL Activity . . . . .	6–14
	Get Event Property Activity . . . . .	6–15
	Set Event Property Activity . . . . .	6–15
	Compare Event Property Activity . . . . .	6–16
	XML Get Tag Value Activity . . . . .	6–18
	XML Compare Tag Value Activities . . . . .	6–19
	XML Transform Activity . . . . .	6–20
	Concurrent Manager Standard Activities . . . . .	6–22
	Execute Concurrent Program Activity . . . . .	6–22
	Submit Concurrent Program Activity . . . . .	6–23
	Wait for Concurrent Program Activity . . . . .	6–24
	Default Error Process . . . . .	6–26

System: Error Item Type and Item Attributes . . . . .	6 – 27
Default Error Process . . . . .	6 – 28
Retry-only Process . . . . .	6 – 32
Default Event Error Process . . . . .	6 – 34

## Chapter 7

<b>Defining Procedures and Functions for Oracle Workflow . . . .</b>	<b>7 – 1</b>
Defining Procedures and Functions for Oracle Workflow . . . . .	7 – 2
Standard API for PL/SQL Procedures Called by Function Activities . . . . .	7 – 3
Standard API for Java Procedures Called by Function Activities . . . . .	7 – 8
Standard API for an Item Type Selector or Callback Function . .	7 – 13
Standard APIs for “PL/SQL” and “PL/SQL CLOB” Documents . . . . .	7 – 17
“PL/SQL” Documents . . . . .	7 – 17
“PL/SQL CLOB” Documents . . . . .	7 – 19
Standard API for an Event Data Generate Function . . . . .	7 – 21
Standard APIs for a Queue Handler . . . . .	7 – 23
Enqueue . . . . .	7 – 23
Dequeue . . . . .	7 – 24
Standard API for an Event Subscription Rule Function . . . . .	7 – 25

## Chapter 8

<b>Oracle Workflow APIs . . . . .</b>	<b>8 – 1</b>
Oracle Workflow Procedures and Functions . . . . .	8 – 2
Overview of the Workflow Engine . . . . .	8 – 3
Oracle Workflow Java Interface . . . . .	8 – 5
Additional Workflow Engine Features . . . . .	8 – 8
Workflow Engine APIs . . . . .	8 – 19
CreateProcess . . . . .	8 – 21
SetItemUserKey . . . . .	8 – 23
GetItemUserKey . . . . .	8 – 24
GetActivityLabel . . . . .	8 – 25
SetItemOwner . . . . .	8 – 26
StartProcess . . . . .	8 – 28
LaunchProcess . . . . .	8 – 30
SuspendProcess . . . . .	8 – 32
ResumeProcess . . . . .	8 – 34
AbortProcess . . . . .	8 – 36
CreateForkProcess . . . . .	8 – 38
StartForkProcess . . . . .	8 – 40

Background	8 – 41
AddItemAttribute	8 – 43
AddItemAttributeArray	8 – 46
SetItemAttribute	8 – 48
SetItemAttrDocument	8 – 51
SetItemAttributeArray	8 – 53
getItemTypes	8 – 56
GetItemAttribute	8 – 57
GetItemAttrDocument	8 – 59
GetItemAttrClob	8 – 60
getItemAttributes	8 – 61
GetItemAttrInfo	8 – 62
GetActivityAttrInfo	8 – 63
GetActivityAttribute	8 – 64
GetActivityAttrClob	8 – 66
BeginActivity	8 – 67
CompleteActivity	8 – 69
CompleteActivityInternalName	8 – 72
AssignActivity	8 – 74
Event	8 – 75
HandleError	8 – 77
SetItemParent	8 – 79
ItemStatus	8 – 80
getProcessStatus	8 – 81
Workflow Function APIs	8 – 82
loadItemAttributes	8 – 83
loadActivityAttributes	8 – 84
getActivityAttr	8 – 85
getItemAttr	8 – 87
setItemAttrValue	8 – 88
execute	8 – 89
Workflow Attribute APIs	8 – 90
WFAttribute	8 – 92
value	8 – 93
getName	8 – 94
getValue	8 – 95
getType	8 – 96
getFormat	8 – 97
getValueType	8 – 98
toString	8 – 99
compareTo	8 – 100
Workflow Core APIs	8 – 101

CLEAR	8 – 102
GET_ERROR	8 – 103
TOKEN	8 – 104
RAISE	8 – 105
CONTEXT	8 – 108
TRANSLATE	8 – 110
Workflow Purge APIs	8 – 111
Items	8 – 113
Activities	8 – 114
Notifications	8 – 115
Total	8 – 116
TotalPERM	8 – 117
AdHocDirectory	8 – 118
Purge Obsolete Workflow Runtime Data Concurrent Program	8 – 119
Workflow Directory Service APIs	8 – 121
GetRoleUsers	8 – 123
GetUserRoles	8 – 124
GetRoleInfo	8 – 125
GetRoleInfo2	8 – 126
IsPerformer	8 – 127
UserActive	8 – 128
GetUserName	8 – 129
GetRoleName	8 – 130
GetRoleDisplayName	8 – 131
SetAdHocUserStatus	8 – 132
SetAdHocRoleStatus	8 – 133
CreateAdHocUser	8 – 134
CreateAdHocRole	8 – 136
AddUsersToAdHocRole	8 – 138
SetAdHocUserExpiration	8 – 139
SetAdHocRoleExpiration	8 – 140
SetAdHocUserAttr	8 – 141
SetAdHocRoleAttr	8 – 142
RemoveUsersFromAdHocRole	8 – 143
Workflow LDAP APIs	8 – 144
Synch_changes	8 – 145
Synch_all	8 – 146
Schedule_changes	8 – 147
Workflow Preferences API	8 – 148
get_pref	8 – 148
Workflow Monitor APIs	8 – 149



GetAccessKey .....	8 – 150
GetDiagramURL .....	8 – 151
GetEnvelopeURL .....	8 – 153
GetAdvancedEnvelopeURL .....	8 – 155
Oracle Workflow Views .....	8 – 157
WF_ITEM_ACTIVITY_STATUSES_V .....	8 – 157
WF_NOTIFICATION_ATTR_RESP_V .....	8 – 159
WF_RUNNABLE_PROCESSES_V .....	8 – 160
WF_ITEMS_V .....	8 – 161
Workflow Queue APIs .....	8 – 162
EnqueueInbound .....	8 – 165
DequeueOutbound .....	8 – 167
DequeueEventDetail .....	8 – 170
PurgeEvent .....	8 – 172
PurgeItemType .....	8 – 173
ProcessInboundQueue .....	8 – 174
GetMessageHandle .....	8 – 175
DequeueException .....	8 – 176
DeferredQueue .....	8 – 177
InboundQueue .....	8 – 178
OutboundQueue .....	8 – 179
ClearMsgStack .....	8 – 180
CreateMsg .....	8 – 181
WriteMsg .....	8 – 182
SetMsgAttr .....	8 – 183
SetMsgResult .....	8 – 184
Document Management APIs .....	8 – 185
get_launch_document_url .....	8 – 186
get_launch_attach_url .....	8 – 187
get_open_dm_display_window .....	8 – 188
get_open_dm_attach_window .....	8 – 189
set_document_id_html .....	8 – 190
Overview of the Oracle Workflow Notification System .....	8 – 192
Notification Model .....	8 – 192
Notification APIs .....	8 – 197
Send .....	8 – 199
Custom Callback Function .....	8 – 200
SendGroup .....	8 – 203
Forward .....	8 – 205
Transfer .....	8 – 207
Cancel .....	8 – 209

CancelGroup .....	8 – 210
Respond .....	8 – 211
Responder .....	8 – 212
VoteCount .....	8 – 213
OpenNotificationsExist .....	8 – 214
Close .....	8 – 215
AddAttr .....	8 – 216
SetAttribute .....	8 – 217
GetAttrInfo .....	8 – 219
GetInfo .....	8 – 220
GetText .....	8 – 221
GetShortText .....	8 – 222
GetAttribute .....	8 – 223
GetAttrDoc .....	8 – 225
GetSubject .....	8 – 226
GetBody .....	8 – 227
GetShortBody .....	8 – 228
TestContext .....	8 – 229
AccessCheck .....	8 – 230
WorkCount .....	8 – 231
getNotifications .....	8 – 232
getNotificationAttributes .....	8 – 233
WriteToClob .....	8 – 234
Overview of the Oracle Workflow Business Event System .....	8 – 235
Business Event System Datatypes .....	8 – 236
Agent Structure .....	8 – 237
getName .....	8 – 237
getSystem .....	8 – 237
setName .....	8 – 238
setSystem .....	8 – 238
Parameter Structure .....	8 – 239
getName .....	8 – 239
getValue .....	8 – 239
setName .....	8 – 240
setValue .....	8 – 240
Parameter List Structure .....	8 – 241
Event Message Structure .....	8 – 242
Initialize .....	8 – 245
getPriority .....	8 – 245
getSendDate .....	8 – 245
getReceiveDate .....	8 – 246
getCorrelationID .....	8 – 246

getParameterList	8 – 246
getEventName	8 – 246
getEventKey	8 – 247
getEventData	8 – 247
getFromAgent	8 – 247
getToAgent	8 – 247
getErrorSubscription	8 – 247
getErrorMessage	8 – 248
getErrorStack	8 – 248
setPriority	8 – 248
setSendDate	8 – 248
setReceiveDate	8 – 249
setCorrelationID	8 – 249
setParameterList	8 – 249
setEventName	8 – 250
setEventKey	8 – 250
setEventData	8 – 250
setFromAgent	8 – 251
setToAgent	8 – 251
setErrorSubscription	8 – 251
setErrorMessage	8 – 251
setErrorStack	8 – 252
Content	8 – 252
Address	8 – 253
AddParameterToList	8 – 253
GetValueForParameter	8 – 253
Example for Using Abstract Datatypes	8 – 255
Mapping Between WF_EVENT_T and OMBAQ_TEXT_MSG	8 – 257
Event APIs	8 – 260
Raise	8 – 261
Send	8 – 265
NewAgent	8 – 267
Test	8 – 268
Enqueue	8 – 269
Listen	8 – 270
Workflow Agent Listener Concurrent Program	8 – 272
SetErrorInfo	8 – 273
SetDispatchMode	8 – 274
AddParameterToList	8 – 275
AddParameterToListPos	8 – 276
GetValueForParameter	8 – 277

GetValueForParameterPos .....	8 – 278
Event Subscription Rule Function APIs .....	8 – 279
Default_Rule .....	8 – 281
Log .....	8 – 283
Error .....	8 – 284
Warning .....	8 – 285
Success .....	8 – 286
Workflow_Protocol .....	8 – 287
Error_Rule .....	8 – 288
SetParametersIntoParameterList .....	8 – 289
Event Function APIs .....	8 – 290
Parameters .....	8 – 291
SubscriptionParameters .....	8 – 293
AddCorrelation .....	8 – 294
Generate .....	8 – 296
Receive .....	8 – 298
Business Event System Replication APIs .....	8 – 300
WF_EVENTS Document Type Definition .....	8 – 302
WF_EVENTS_PKG.Generate .....	8 – 303
WF_EVENTS_PKG.Receive .....	8 – 304
WF_EVENT_GROUPS Document Type Definition .....	8 – 305
WF_EVENT_GROUPS_PKG.Generate .....	8 – 306
WF_EVENT_GROUPS_PKG.Receive .....	8 – 307
WF_SYSTEMS Document Type Definition .....	8 – 308
WF_SYSTEMS_PKG.Generate .....	8 – 309
WF_SYSTEMS_PKG.Receive .....	8 – 310
WF_AGENTS Document Type Definition .....	8 – 311
WF_AGENTS_PKG.Generate .....	8 – 312
WF_AGENTS_PKG.Receive .....	8 – 313
WF_EVENT_SUBSCRIPTIONS Document Type Definition .....	8 – 314
WF_EVENT_SUBSCRIPTIONS_PKG.Generate .....	8 – 315
WF_EVENT_SUBSCRIPTIONS_PKG.Receive .....	8 – 316

## Index

## VOLUME 2

<b>Chapter 9</b>	<b>Oracle Workflow Home Page</b> . . . . .	<b>9 – 1</b>
	Accessing the Oracle Workflow Home Page . . . . .	9 – 2
	Setting User Preferences . . . . .	9 – 6
<b>Chapter 10</b>	<b>Viewing Notifications and Processing Responses</b> . . . . .	<b>10 – 1</b>
	Overview of Notification Handling . . . . .	10 – 2
	Reviewing Notifications via Electronic Mail . . . . .	10 – 2
	Viewing Notifications from a Web Browser . . . . .	10 – 12
	Reviewing a Summary of Your Notifications via Electronic Mail . . . . .	10 – 24
	Defining Rules for Automatic Notification Processing . . . . .	10 – 25
<b>Chapter 11</b>	<b>Monitoring Workflow Processes</b> . . . . .	<b>11 – 1</b>
	Overview of Workflow Monitoring . . . . .	11 – 2
	Workflow Monitor . . . . .	11 – 2
	Workflow Monitor Access . . . . .	11 – 7
<b>Chapter 12</b>	<b>Testing a Workflow Definition</b> . . . . .	<b>12 – 1</b>
	Testing Workflow Definitions . . . . .	12 – 2
<b>Chapter 13</b>	<b>Managing Business Events</b> . . . . .	<b>13 – 1</b>
	Managing Business Events . . . . .	13 – 2
	Events . . . . .	13 – 4
	Systems . . . . .	13 – 17
	Agents . . . . .	13 – 22
	Event Subscriptions . . . . .	13 – 34
	Setting Up Message Propagation . . . . .	13 – 53
	Raising Events . . . . .	13 – 65
	Signing Up Systems . . . . .	13 – 67
	Synchronizing Systems . . . . .	13 – 70
	Reviewing Local Queues . . . . .	13 – 72
	Workflow Agent Ping/Acknowledge . . . . .	13 – 77
	The Workflow Agent Ping/Acknowledge Item Type . . . . .	13 – 78
	Summary of the Master Ping Process . . . . .	13 – 79
	Master Ping Process Activities . . . . .	13 – 80
	Summary of the Detail Ping Process . . . . .	13 – 81

Detail Ping Process Activities .....	13 – 82
--------------------------------------	---------

## Chapter 14

<b>Predefined Workflow Events .....</b>	<b>14 – 1</b>
Predefined Workflow Events .....	14 – 2
Event Definition Events .....	14 – 2
Event Group Definition Events .....	14 – 3
System Definition Events .....	14 – 4
Agent Definition Events .....	14 – 4
Event Subscription Definition Events .....	14 – 5
Synchronize Event Systems Event .....	14 – 5
Seed Event Group .....	14 – 6
Ping Agent Events .....	14 – 8
System Signup Event .....	14 – 9
Any Event .....	14 – 10
Unexpected Event .....	14 – 12
User Entry Has Changed Event .....	14 – 15
Workflow Send Protocol .....	14 – 17
The Workflow Send Protocol Item Type .....	14 – 18
Summary of the Workflow Event Protocol Process .....	14 – 20
Workflow Event Protocol Process Activities .....	14 – 21
Workflow Send Protocol Events .....	14 – 24

## Chapter 15

<b>Demonstration Workflow Processes .....</b>	<b>15 – 1</b>
Sample Workflow Processes .....	15 – 2
Displaying the Process Diagram of a Sample Workflow .....	15 – 3
Requisition Process .....	15 – 5
Installing the Requisition Data Model .....	15 – 6
Initiating the Requisition Workflow .....	15 – 8
The Requisition Item Type .....	15 – 12
Summary of the Requisition Approval Process .....	15 – 13
Requisition Process Activities .....	15 – 15
Summary of the Notify Approver Subprocess .....	15 – 19
Notify Approver Subprocess Activities .....	15 – 21
Sample StartProcess Function .....	15 – 23
Example Function Activities .....	15 – 26
Example: Select Approver .....	15 – 26
Example: Verify Authority .....	15 – 29
Example Notification Activity .....	15 – 31
Example: Notify Requisition Approval Required .....	15 – 31
Product Survey Process .....	15 – 34

Installing the Product Survey Data Model .....	15 – 35
Initiating the Product Survey Workflow .....	15 – 36
The Product Survey Item Type .....	15 – 38
Summary of the Survey – Single Process .....	15 – 39
Survey – Single Process Activities .....	15 – 41
Summary of the Survey – Master/Detail Process .....	15 – 42
Survey – Master/Detail Process Activities .....	15 – 44
Summary of the Detail Survey Process .....	15 – 46
Detail Survey Process Activities .....	15 – 47
Document Review Process .....	15 – 49
The Document Management Item Type .....	15 – 49
Summary of the Document Review Process .....	15 – 50
Document Review Process Activities .....	15 – 52
Error Check Process .....	15 – 54
The Periodic Alert Item Type .....	15 – 54
Summary of the Error Check Process .....	15 – 56
Error Check Process Activities .....	15 – 57
Summary of the User Defined Alert Action Process .....	15 – 60
User Defined Alert Action Process Activities .....	15 – 61
Event System Demonstration .....	15 – 63
Installing the Event System Demonstration Data Model ...	15 – 64
Initiating the Event System Demonstration Workflow .....	15 – 66
The Event System Demonstration Item Type .....	15 – 71
Summary of the Buyer: Top Level PO Process .....	15 – 73
Buyer: Top Level PO Process Activities .....	15 – 75
Summary of the Buyer: Send PO to Supplier Subprocess ...	15 – 78
Buyer: Send PO to Supplier Subprocess Activities .....	15 – 78
Summary of the Buyer: Receive Supplier PO Acknowledgement Subprocess .....	15 – 80
Buyer: Receive Supplier PO Acknowledgement Subprocess Activities .....	15 – 81
Summary of the Buyer: Advanced Shipment Notice Subprocess .....	15 – 83
Buyer: Advanced Shipment Notice Subprocess Activities ..	15 – 84
Summary of the Buyer: Receive Supplier Invoicing Subprocess .....	15 – 85
Buyer: Receive Supplier Invoicing Subprocess Activities ...	15 – 86
Summary of the Supplier: Top Level Order Process .....	15 – 87
Supplier: Top Level Order Process Activities .....	15 – 88
Summary of the Supplier: Get Order Details Subprocess ...	15 – 91
Supplier: Get Order Details Subprocess Activities .....	15 – 92
Summary of the Supplier: Credit Check Subprocess .....	15 – 94

Supplier: Credit Check Subprocess Activities . . . . .	15 – 95
Summary of the Supplier: Stock Check Subprocess . . . . .	15 – 96
Supplier: Stock Check Subprocess Activities . . . . .	15 – 97
Summary of the Supplier: Advanced Shipment Notice Subprocess . . . . .	15 – 98
Supplier: Advanced Shipment Notice Subprocess Activities . . . . .	15 – 99
Summary of the Supplier: Send Supplier Invoice Subprocess . . . . .	15 – 100
Supplier: Send Supplier Invoice Subprocess Activities . . . . .	15 – 101
B2B Purchase Order Event . . . . .	15 – 102
B2B Purchase Order Acknowledgement Event . . . . .	15 – 105
B2B Advanced Shipment Notice Event . . . . .	15 – 106
B2B Invoice Event . . . . .	15 – 107

## Chapter 16

<b>Workflow Administration Scripts . . . . .</b>	<b>16 – 1</b>
Miscellaneous SQL Scripts . . . . .	16 – 2
FNDWFLST . . . . .	16 – 4
FNDWFPR . . . . .	16 – 5
WFNLADD.sql . . . . .	16 – 5
Wfagtlst.sql . . . . .	16 – 6
Wfbkg.sql . . . . .	16 – 6
Wfbkgchk.sql . . . . .	16 – 7
Wfchact.sql . . . . .	16 – 7
Wfchacta.sql . . . . .	16 – 7
Wfchita.sql . . . . .	16 – 8
Wfchitt.sql . . . . .	16 – 8
Wfchluc.sql . . . . .	16 – 8
Wfchlut.sql . . . . .	16 – 9
Wfchmsg.sql . . . . .	16 – 9
Wfchmsga.sql . . . . .	16 – 9
Wfdirchk.sql . . . . .	16 – 10
Wfevtenq.sql . . . . .	16 – 10
Wfjvstop.sql . . . . .	16 – 11
Wfmqupd.sql . . . . .	16 – 12
Wfnlena.sql . . . . .	16 – 12
Wfntfsh.sql . . . . .	16 – 12
Wfprot.sql . . . . .	16 – 12
Wfqclean.sql . . . . .	16 – 13
Wfrefchk.sql . . . . .	16 – 13
Wfretry.sql . . . . .	16 – 13
Wfrmall.sql . . . . .	16 – 14



Wfrmita.sql	16 – 14
Wfrmitms.sql	16 – 15
Wfrmitt.sql	16 – 15
Wfrmtime.sql	16 – 15
Wfrun.sql	16 – 15
Wfstat.sql	16 – 16
Wfstatus.sql	16 – 16
Wfstdchk.sql	16 – 16
Wfver.sql	16 – 16
Wfverchk.sql	16 – 17
Wfverupd.sql	16 – 17

<b>Appendix A</b>	<b>Oracle Workflow Builder Menus and Toolbars</b>	<b>A – 1</b>
	Oracle Workflow Builder Menus	A – 2
	Oracle Workflow Builder Toolbars	A – 7

<b>Appendix B</b>	<b>Oracle Workflow Implementation in Other Oracle Products</b>	<b>B – 1</b>
	Predefined Workflows Embedded in Oracle E–Business Suite	B – 2
	Oracle Workflow Business Event System Implementation in Oracle E–Business Suite	B – 16
	Oracle Workflow Implementation in the Oracle9i Platform	B – 18
	Oracle Support Policy for Predefined Workflows, Events, and Subscriptions	B – 20
	Customization Guidelines	B – 20
	Resolving Customization Issues	B – 21
	What Is NOT Supported	B – 21
	What Is Supported	B – 21

<b>Appendix C</b>	<b>Oracle Workflow Performance Concepts</b>	<b>C – 1</b>
	Oracle Workflow Performance Concepts	C – 2
	Designing Workflow Processes for Performance	C – 2
	Managing Runtime Data for Performance	C – 8

**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, Oracle suggests 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 Oracle database server, PL/SQL, and Oracle9i Application Server technology. If you have not yet been introduced to any of these systems, Oracle suggests 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.

- 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 and Java 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 how to manage business events.
- Chapter 14 describes the standard events provided with Oracle Workflow.
- Chapter 15 describes the demonstration workflow processes included with Oracle Workflow.
- Chapter 16 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, the Oracle Applications features that leverage the Business Event System, and the Oracle9i platform features that leverage Oracle Workflow. This appendix also includes the Oracle Workflow support policy.
- Appendix C describes concepts and techniques that you can use for performance gain when running Oracle Workflow.

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

## Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle Corporation is actively engaged with other market-leading technology vendors to address technical

obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at <http://www.oracle.com/accessibility/>.

### **Accessibility of Code Examples in Documentation**

---

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

### **Accessibility of Links to External Web Sites in Documentation**

---

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

---

## **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.

## **Online Documentation**

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

- **Online Help** – The new features section in the HTML help describes new features in 11*i*. This information is updated for each new release of Oracle Workflow. The new features section also includes information about any features that were not yet available when this guide was printed. For example, if your administrator has installed software from a mini-pack or an upgrade, this document describes the new features. Online help patches are available on *OracleMetaLink*.

- **11i Features Matrix** – This document lists new features available by patch and identifies any associated new documentation. The new features matrix document is available on *OracleMetaLink*.
- **Readme File** – Refer to the readme file for patches that you have installed to learn about new documentation or documentation patches that you can download.

Portions of this guide are also available online in Windows Help format. 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 HTML format, and portions of the guide are available in Windows Help format as well. 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 or from the help icon in the Oracle Workflow web pages.

## Related User's 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's guides when you set up and use Oracle Workflow 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 Document Library CD included in your media pack, or by using a Web browser with a URL that your system administrator provides.

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

## Guides Related to All Products

### Oracle Applications User's Guide

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

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

## **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's 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 11i User Guide Online Help**

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

## **Oracle Financials Open Interface Reference**

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

## **Oracle XML Gateway User's Guide**

This guide explains how to implement the production and consumption of valid, well-formed XML messages between Oracle Applications and trading partners.

# **Installation and System Administration**

## **Oracle Applications Concepts**

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

## **Installing Oracle Applications**

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

## **Upgrading Oracle Applications**

Refer to this guide if you are upgrading your Oracle Applications Release 10.7 or Release 11.0 products to Release 11*i*. This guide describes the upgrade process and lists database and product-specific

upgrade tasks. You must be either at Release 10.7 (NCA, SmartClient, or character mode) or Release 11.0, to upgrade to Release 11*i*. You cannot upgrade to Release 11*i* directly from releases prior to 10.7.

### **Maintaining Oracle Applications**

---

Use this guide to help you run the various AD utilities, such as AutoUpgrade, AutoPatch, AD Administration, AD Controller, AD Relink, License Manager, and others. It contains how-to steps, screenshots, and other information that you need to run the AD utilities. This guide also provides information on maintaining the Oracle applications file system and database.

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

---

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

### **Oracle Alert User's Guide**

---

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

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

---

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

## **Other Implementation Documentation**

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

---

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

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

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

## **Multiple Organizations in Oracle Applications**

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

## **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 manual also provides information on creating custom reports on flexfields data.

## **Oracle eTechnical Reference Manuals**

Each eTechnical Reference Manual (eTRM) contains database diagrams and a detailed description of database tables, forms, reports, and programs for a specific Oracle Applications product. This information helps you convert data from your existing applications, integrate Oracle Applications data with non-Oracle applications, and write custom reports for Oracle Applications products. Oracle eTRM is available on *OracleMetaLink*.

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

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

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

This manual contains up-to-date information about integrating with other Oracle Manufacturing applications and with your other systems.

This documentation includes APIs and open interfaces found in Oracle Manufacturing.

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

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

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

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

## **Training and Support**

### **Training**

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

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

### **Support**

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

---

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

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

Oracle provides powerful tools you can use to create, store, change, retrieve, and maintain information in an Oracle database. But if you use Oracle tools such as SQL\*Plus to modify Oracle Applications data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

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

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

---

## About Oracle

Oracle Corporation develops and markets an integrated line of software products for database management, applications development, decision support, and office automation, as well as Oracle Applications, an integrated suite of more than 160 software modules for financial management, supply chain management, manufacturing, project systems, human resources, and customer relationship management.

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

Oracle is the world's leading supplier of software for information management, and the world's second largest software company. Oracle offers its database, tools, and applications products, along with related consulting, education, and support services, in over 145 countries around the world.

---

## Your Feedback

Thank you for using Oracle Workflow and this guide.

Oracle values your comments and feedback. At the end of this guide is a Reader's Comment Form 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](mailto:appsdoc_us@oracle.com).

# Overview of Oracle Workflow

**T**his 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.
- Workflow Monitor, which allows you to track your workflow process using a web browser.
- The Business Event System, which communicates business events between systems.

---

## 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 Oracle database server, to express any business rule that affects a workflow process. See: Workflow Processes: page 1 – 6.

### **Delivering Electronic Notifications**

---

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



## **Integrating Systems**

---

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

---

## **Major Features and Definitions**

### **Oracle Workflow Builder**

---

Oracle Workflow Builder lets you create, view, or modify a business process with simple drag and drop operations. Using the Workflow Builder, you can create and modify all workflow objects, including activities, item types, and messages. See: Workflow Processes: page 1 – 6.

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

### **Workflow Engine**

---

The Workflow Engine embedded in the Oracle database 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.

### **Business Event System**

---

The Business Event System is an application service that uses the Oracle Advanced Queuing (AQ) infrastructure to communicate business events

between systems. The Business Event System consists of the Event Manager, which lets you register subscriptions to significant events, and event activities, which let you model business events within workflow processes.

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

### **Workflow Definitions Loader**

---

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

### **Complete Programmatic Extensibility**

---

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

### **Electronic Notifications**

---

Oracle Workflow lets you include users in your workflows to handle activities that cannot be automated, such as approvals for requisitions or sales orders. 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.

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

Also, if you are using the standalone version of Oracle Workflow available with Oracle<sup>9i</sup> Release 2, you can use the standalone Oracle Workflow Manager component available through Oracle Enterprise Manager as an additional administration tool for Oracle Workflow. For more information, please refer to the Oracle Workflow Manager online help.

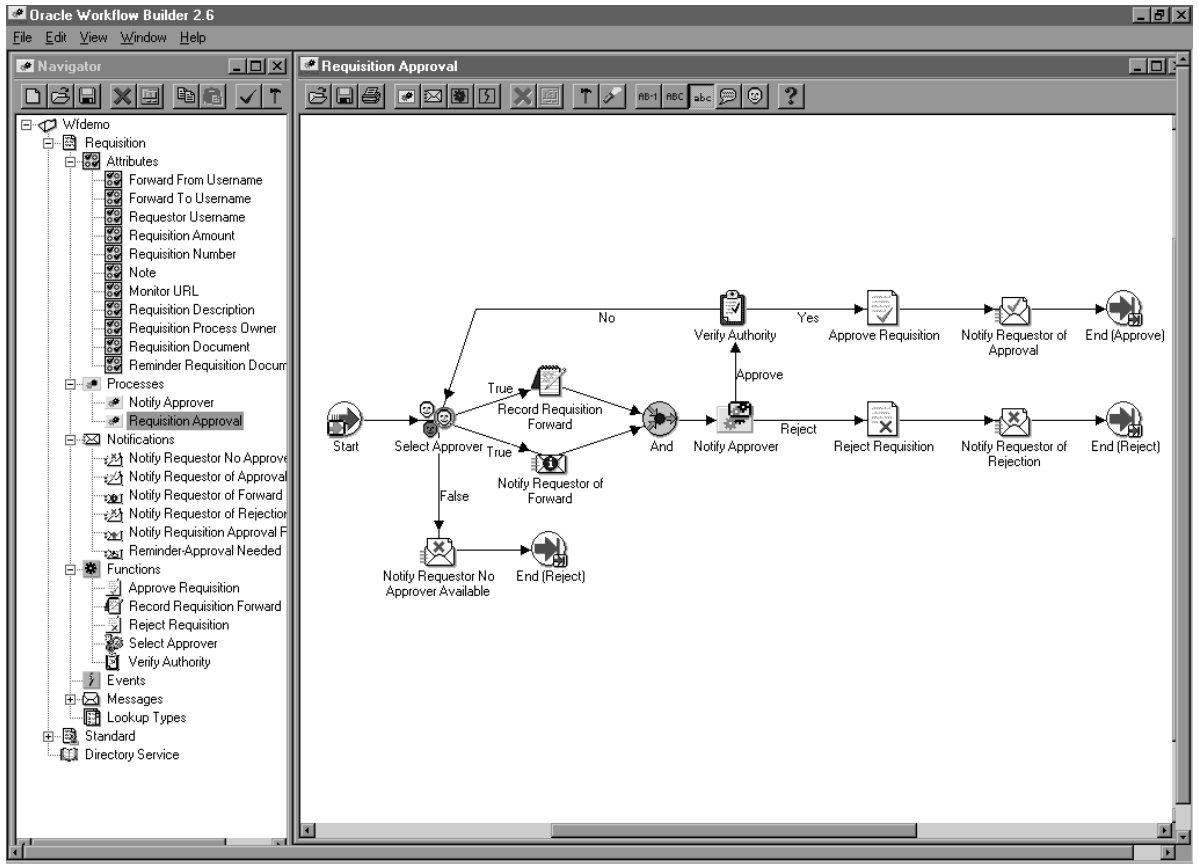
---

## Workflow Processes

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

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

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



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

**T**his 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 Universal Installer and requires the installation of Oracle Net Services (version 8.1.6 or higher for Oracle8i, or version 9.0.1 or higher for Oracle9i) and Required Support Files (version 8.1.6 or higher for Oracle8i, or version 9.0.1 or higher for Oracle9i). 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.
  - Remote access and control software to be used by Customer Support for dial-in access through a modem to your PC. The preferred software is Symantec's Norton pcANYWHERE, or Microcom's Carbon Copy. Without some form of remote access and control software, Oracle Worldwide Customer Support will not be able to dial in to your site to diagnose problems, nor will they be able to supply patches directly to your client PC.



**Warning:** Please follow the necessary security precautions against viruses and unauthorized access when installing any software that allows remote access.

- Dual speed, ISO 9660 format CD-ROM available as a logical drive
- Microsoft Windows 95, Windows 98, Windows 2000, or Windows NT 4.0 or higher
- At least 60 Mb of available disk space to install Oracle Workflow Builder, Oracle Net Services, and Required Support Files.
- At least 32 Mb of memory, 64 Mb recommended





**Attention:** Oracle Net Services require and only support the use of Microsoft's TCP/IP drivers.

- The Oracle Workflow Server requires the following:
  - Oracle8*i* Enterprise or Standard Edition database version 8.1.6 or higher, or Oracle9*i* Enterprise or Standard Edition database version 9.0.1 or higher, along with the Oracle Objects and JServer Options, installed on a supported server machine
  - At least 40 Mb of available disk space for Oracle Workflow Server once it is installed in your Oracle Home
  - At least 128 Mb of memory, 256 Mb recommended
  - Oracle Net Services version 8.1.6 or higher for Oracle8*i*, or version 9.0.1 or higher for Oracle9*i*
  - SQL\*Plus version 8.1 or higher for Oracle8*i*, or version 9.0.1 or higher for Oracle9*i*

If you are installing Oracle Workflow Server on Microsoft Windows NT, the following additional hardware and software configurations are required:

- ISO 9660 format CD-ROM available as a logical drive
- Microsoft Windows NT 4.0 or higher



**Attention:** While the version of Oracle Workflow offered to Oracle8*i* Standard Edition customers is exactly the same as the version offered to Oracle8*i* Enterprise Edition customers, it is important to note that Oracle Workflow leverages Oracle8*i* functionality. Consequently, using an Oracle8*i* Standard Edition database limits some of the features available for use by the Oracle Workflow Business Event System.

For example:

- You cannot create any additional queues in Oracle8*i* Standard Edition beyond the default queues provided by Oracle Workflow. If you require additional queues, you should choose Oracle8*i* Enterprise Edition.
- Oracle Advanced Queuing propagation in Oracle8*i* Standard Edition does not support propagating messages outside the local database. If you require messages to be propagated to other systems, you should choose Oracle8*i* Enterprise Edition.

In Oracle9i, however, these restrictions no longer apply. Exactly the same functionality is available with Oracle Workflow in an Oracle9i Standard Edition database as in an Oracle9i Enterprise Edition database.

- The e-mail notifications component contains a Notification Mailer program that can send mail through 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 UNIX gateway product to communicate with your e-mail application of choice.



**Attention:** The Microsoft Outlook E-mail Security Update that was released on June 7, 2000 desupports the MAPI Common Messaging Calls (CMC) interface used by the Oracle Workflow MAPI Mailer. (See: *OL2000: Developer Information About the Outlook E-mail Security Update*, <http://support.microsoft.com/support/kb/articles/Q262/7/01.ASP>.) As a result, the Oracle Workflow MAPI Mailer is not certified on any Microsoft Windows platforms where this Microsoft Outlook E-mail Security Update or above has been applied. The Oracle Workflow MAPI Mailer is not certified on Windows XP.

Workflow customers running on NT/2000 are certified to install the UNIX version of the Oracle Workflow Notification Mailer (on UNIX) and connect to a Workflow Server database running on NT/2000.

- 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, Workflow Monitor, and Event Manager components require Oracle HTTP Server and mod\_plsql to be installed on a server machine. The Oracle HTTP Server and mod\_plsql components are included with Oracle8i Database version 8.1.7 or higher and with Oracle9i Database version 9.0.1 or higher, as well as with Oracle9i Application Server version 1.0.1 or higher.

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 1.1.8 or higher and Abstract Windowing Toolkit (AWT), such as Netscape Communicator version 4.76 or a higher version of 4.7x, or Microsoft Internet Explorer version 5.0x or 5.5x.

- To run external Java function activities and to use the Workflow XML Loader, you must have Java Runtime Environment (JRE) version 1.1.8, or a higher 1.1.x version, installed.
- To extract the HTML help for the standalone version of Oracle Workflow, you need an unzip utility.
- To implement Oracle Internet Directory integration, you must have Oracle Internet Directory Release 2 (9.0.2) installed. To implement Single Sign-On integration, you must implement Oracle Internet Directory integration, and you must have Oracle*9i*AS Single Sign-On Server Release 2 (9.0.2) and Oracle*9i*AS Portal Release 2 (9.0.2) installed and have mod\_osso installed with Oracle HTTP Server.

---

## Overview of Setting Up

After you install Oracle Workflow, you implement it for your site by setting up the preferences and components appropriate for your enterprise.

---

## Overview of Required Setup 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 – 14.
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 individual users or a group of users. Oracle Workflow provides example directory services views that you can modify and reload. See: Setting Up an Oracle Workflow Directory Service: page 2 – 21.
3. Create a view called WF\_LANGUAGES that identifies the languages defined in your Oracle database server 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 – 38.
4. 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 – 42.
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 – 43.

---

## Overview of Required Setup 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 – 14.
2. Set the system profile options called Socket Listener Activated and Socket Listener Port. See: Setting the Socket Listener Profile Options: page 2 – 40.
3. Set up background Workflow Engines to control the load and throughput of the primary Workflow Engine on your system. You can specify the cost threshold level of your primary and background engines to determine the activities an engine processes and the activities an engine defers. See: Setting Up Background Workflow Engines: page 2 – 43.



**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 – 21
- WF\_LANGUAGES view: page 2 – 38
- Path to the language-dependent resources file: page 2 – 42

---

## Optional Setup Steps

1. 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 – 12.
2. If you are using the standalone version of Oracle Workflow with Oracle9i Database Server Release 2 or Oracle9i Application Server (Oracle9iAS) Release 2 or higher, you can synchronize the user information in your Workflow directory service with Oracle Internet Directory (OID). Additionally, if you have installed Oracle9iAS Release 2 or higher, you can also use OID integration to implement single sign-on integration. See: Synchronizing Workflow Directory Services with Oracle Internet Directory: page 2 – 30.

3. 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 – 48.
4. You can modify the templates for your electronic mail notifications. See: Modifying Your Message Templates: page 2 – 69.
5. Customize the company logo that appears in Oracle Workflow's web pages. See: Customizing the Logo on Oracle Workflow's Web Pages: page 2 – 84.
6. 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 – 85.
7. Set up the Java Function Activity Agent if you are using the standalone version of Oracle Workflow and you want to run external Java function activities. See: Setting Up the Java Function Activity Agent: page 2 – 86.
8. Set up the Business Event System if you want to communicate business events between systems using event subscription processing and Workflow process event activities. See: Setting Up the Business Event System: page 2 – 96.
9. Set up the WF\_EVENT\_OMB\_QH queue handler if you are using the Business Event System with Oracle8i and you want to use Oracle Message Broker to propagate event messages between systems. See: Setting Up the WF\_EVENT\_OMB\_QH Queue Handler: page 2 – 100.

---

## 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 – 101.

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 – 107.

If you are using the Business Event System, you can use the Workflow XML Loader to load XML definitions for Business Event System objects

between a database and a flat file. See: Using the Workflow XML Loader: page 2 – 112.

---

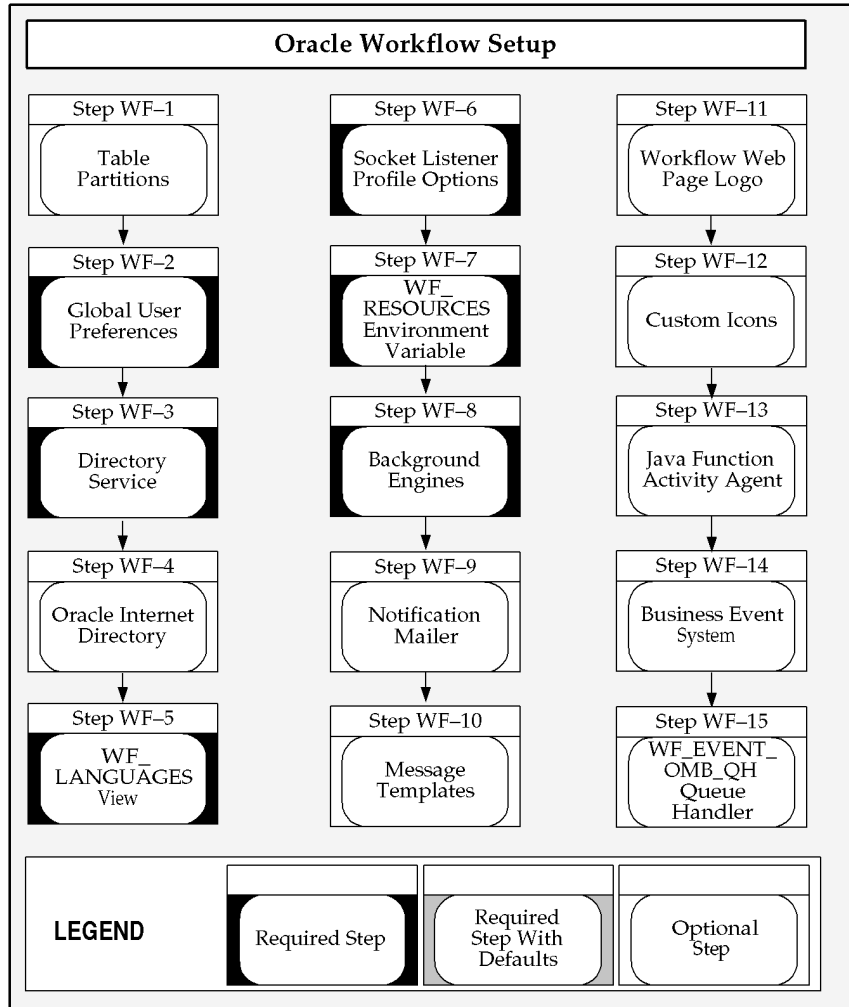
## Identifying the Version of Your Oracle Workflow Server

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 16 – 16.

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 2.6 against an Oracle Workflow 2.0.3 database.

## Setup Flowchart

The following flowchart shows the Oracle Workflow setup steps. Some of the steps 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 – 12	Both
Step 2	Required	Setting Global User Preferences: page 2 – 14	Both
Step 3	Required	Setting Up an Oracle Workflow Directory Service: page 2 – 21	Standalone
Step 4	Optional	Synchronizing Workflow Directory Services with Oracle Internet Directory: page 2 – 30	Standalone
Step 5	Required	Creating the WF_LANGUAGES View: page 2 – 38	Standalone
Step 6	Required	Setting the Socket Listener Profile Options: page 2 – 40	Embedded
Step 7	Required	Setting the WF_RESOURCES Environment Variable: page 2 – 42	Standalone
Step 8	Required	Setting Up Background Workflow Engines: page 2 – 43	Both
Step 9	Optional	Implementing the Notification Mailer: page 2 – 48	Both
Step 10	Optional	Modifying Your Message Templates: page 2 – 69	Both
Step 11	Optional	Customizing the Logo on Oracle Workflow's Web Pages: page 2 – 84	Both
Step 12	Optional	Adding Custom Icons to Oracle Workflow: page 2 – 85	Both
Step 13	Optional	Setting Up the Java Function Activity Agent: page 2 – 86	Standalone
Step 14	Optional	Setting Up the Business Event System: page 2 – 96	Both
Step 15	Optional	Setting Up the WF_EVENT_OMB_QH Queue Handler: page 2 – 100	Both

## Setup Steps

### Step 1 Partitioning Workflow Tables

Partitioning addresses key issues in supporting very large tables and indexes by letting you decompose them into smaller and more manageable pieces called partitions. SQL queries and DML statements do not need to be modified in order to access partitioned tables. However, once partitions are defined, DDL statements can access and manipulate individual partitions rather than entire tables or indexes. In this way, partitioning can simplify the manageability of large database objects. Also, partitioning is entirely transparent to applications.

You can optionally run a script to partition certain Workflow tables that store runtime status data. For the version of Oracle Workflow embedded in Oracle Applications, the script is called `wfupartb.sql`; for the standalone version of Oracle Workflow, the script is called `wfupart.sql`. This step is highly recommended for performance gain.

The script partitions four Workflow tables and recreates the associated indexes. The following table shows the Workflow tables and indexes on which the script runs.

Table	Index
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 the partitioning script, 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 disk space 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 the partitioning script through Oracle Net Services, then you must set the `TWO_TASK` variable before you begin.

For Oracle Workflow embedded in Oracle Applications, the `wfupartb.sql` 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
```

For standalone Oracle Workflow, the `wfupart.sql` script is located in the `wf/admin/sql` subdirectory in your Oracle Home. Use the script as follows:

```
sqlplus <wf_user>/<wf_passwd> @wfupart <wf_user> <wf_passwd>
```

For example:

```
sqlplus owf_mgr/owf_mgr @wfupart owf_mgr owf_mgr
```

If the partitioning 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.

## See Also

Partitioning for Performance: page C – 8

## 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 hard-coded 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 – 21.

**Context:** You need to perform this step only once.

See: Setting User Preferences: page 9 – 6.

### ► 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
```

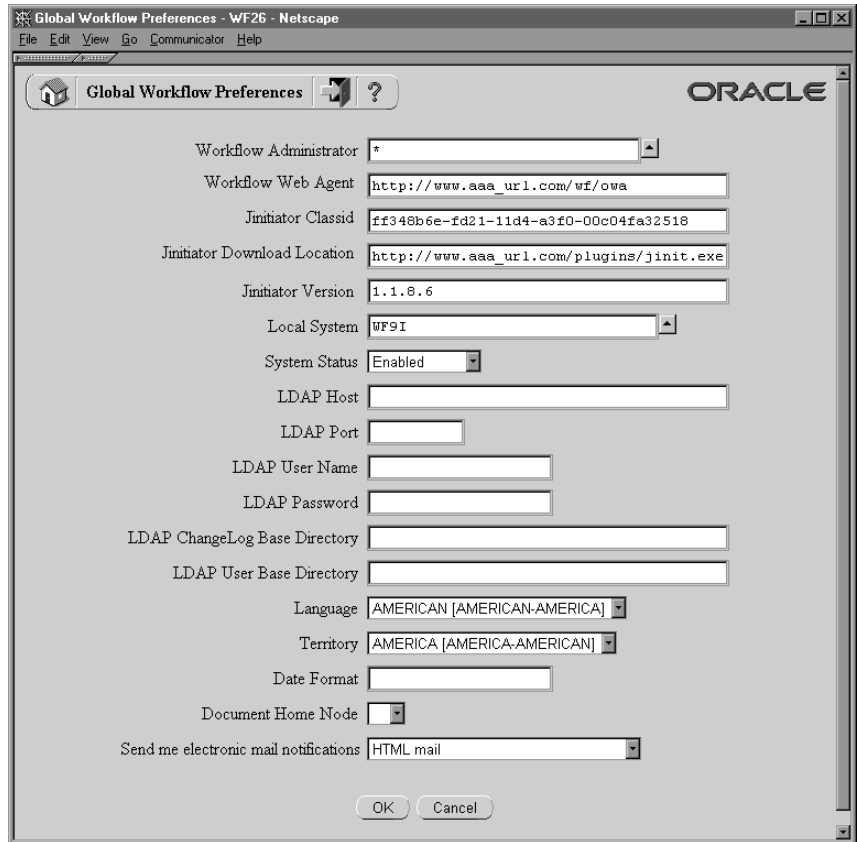
<webagent> represents the base URL of the web agent you configured for Oracle Workflow in your Web server.



**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 and access the Event Manager web pages.

If you want all users and roles to have workflow administrator 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 – 21.

**Note:** To find out which role currently has workflow administrator privileges, without accessing the Global Workflow Preferences page, you can use the following command:

```
select text
from wf_resources
where name = 'WF_ADMIN_ROLE';
```

After installing Oracle Workflow, you should change the Workflow Administrator preference from the default setting to the role that you want to have administrator privileges.

- For the standalone version of Oracle Workflow, the default setting after installation is an asterisk (\*). You can log in as any user to access the Global Workflow Preferences page and specify the preferences you want.
- For the version of Oracle Workflow embedded in Oracle Applications, the default setting after installation is SYSADMIN. You must log in as the SYSADMIN user to access the Global Workflow Preferences page and specify the preferences you want.

**Note:** The SYSADMIN role is different than the role associated with the System Administrator responsibility in Oracle Applications. If you want to assign workflow administrator privileges to this or any other Oracle Applications responsibility, you must set the Workflow Administrator preference to the internal name of the Workflow role associated with that responsibility.

You can query the WF\_ROLES view to find the role name for a responsibility. For example, to find the role names for various administrator responsibilities in Oracle Applications, use the following command:

```
select name, display_name
from wf_roles
where display_name like '%Admin%';
```

If you set the Workflow Administrator preference to the role name of a responsibility, then any Oracle Applications user with that responsibility will have workflow administrator privileges.

4. In the Workflow Web Agent field, enter the base URL of the Oracle web agent you defined for Oracle Workflow in Oracle HTTP 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 HTTP Server as your Web server:

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

See your Oracle HTTP 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. If you are using the version of Oracle Workflow embedded in Oracle Applications, enter the Jinitiator plugin class ID, download location, and version number. This information is required for Oracle Workflow to launch Oracle Applications forms linked to notifications and to display the Workflow Monitor. See: Setting the Socket Listener Profile Options: page 2 – 40.

You can find the class ID and version number for the version of JInitiator you have installed in the jinit-version.txt file (<drive>:\Program Files\Oracle\jinit<version>\doc\jinit-version.txt). The download location is the location where you have staged the JInitiator executable for download to users' client machines. For more information, refer to Complete Guide to JInitiator for Oracle's E-Business Suite (Note 162488.1) and Upgrading Oracle JInitiator with Oracle Applications 11i (Note 124606.1), available on MetaLink.

6. The Local System field displays the system name for the database where this installation of Oracle Workflow is located. Oracle Workflow automatically creates the system definition for this database in the Event Manager during installation. The Business Event System treats this system as the local system and all others as external systems. See: Systems: page 13 – 17.

**Note:** The Local System setting is specific to this installation of Oracle Workflow and is not included when Business Event System data is replicated to other systems.

7. In the System Status field, use the list of values to select the Business Event System status that you want to assign to the local system.
  - Enabled—Subscriptions are executed on all events.



- Local Only—Subscriptions are executed only on events raised on the local system.
  - External Only—Subscriptions are executed only on events received from external systems.
  - Disabled—No subscriptions are executed on any events.
 

**Note:** Oracle Workflow sets the system status to Enabled by default. After you finish setting up the Business Event System, you can change the setting to the status you want for event processing.

**Note:** The System Status setting is specific to this installation of Oracle Workflow and is not included when Business Event System data is replicated to other systems.
8. If you are implementing Oracle Internet Directory (OID) synchronization, specify the Lightweight Directory Access Protocol (LDAP) server information for the LDAP directory to which you want to connect.
    - LDAP Host—The host on which the LDAP directory resides.
    - LDAP Port—The port on the host.
  9. If you are implementing OID synchronization, specify the LDAP user account used to connect to the LDAP server. This LDAP user account must have write privileges.
    - LDAP User Name—The LDAP user. This user name is required to bind to the LDAP directory. For example:
 

```
cn=orcladmin
```
    - LDAP Password—The LDAP password. The password is stored in encrypted form.
  10. If you are implementing OID synchronization, specify the directories for the change log and the user records.
    - LDAP Changelog Base Directory—The LDAP node under which change logs are located. For example:
 

```
cn=changelog
```
    - LDAP User Base Directory—The LDAP node under which user records can be found. For example:
 

```
cn=Base, cn=OracleSchemaVersion
```
  11. 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.

12. In the Date Format field, specify an Oracle8i–compliant date format that defines the default date format for the workflow database sessions of all users. An example of an Oracle8i–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.

13. Leave the Document Home Node field blank. This functionality is reserved for future use.

14. In the ‘Send me electronic mail notifications’ field, use the list of values to select 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.

15. Click 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, workflow web agent, local system, and LDAP 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 – 121.



**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.

**Note:** If you are using the standalone version of Oracle Workflow, and you have installed Oracle9i Release 2 or higher or Oracle9iAS Release 2 or higher, you can integrate your Workflow directory service with Oracle Internet Directory (OID) as your directory repository. In this case, you must map your directory service views only to the WF\_LOCAL tables to enable synchronization of Workflow user information with OID. See: Integrating Oracle Workflow Directory Services with Local Workflow Users: page 2 – 28 and Synchronizing Workflow Directory Services with Oracle Internet Directory: page 2 – 30.

**Context:** You need to perform this step only once.

See: Predefined Directory Services: page 2 – 26

See: Ad Hoc Users and Roles: page 5 – 24

## 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 – 49.

**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 – 14, User Preferences: page 9 – 6 and get\_pref: page 8 – 148.

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

- **Language**—The value of the database NLS\_LANGUAGE initialization parameter that specifies the default language-dependent behavior of the user's notification session. Refer to your Oracle Database 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: Global Preferences: page 2 – 14, User Preferences: page 9 – 6 and get\_pref: page 8 – 148.

**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 – 14 and User Preferences: page 9 – 6.

```
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 have 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 – 69.

- **Territory**—The value of the database NLS\_TERRITORY initialization parameter that specifies the default territory-dependant formatting used in the user's notification session. Refer to your Oracle Database 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 – 14, User Preferences: page 9 – 6 and get\_pref: page 8 – 148.

```
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
- \*Status
- \*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. The Oracle database server 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 – 121.

## 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 16 – 10.

### ► **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 *wfdirowv.sql* script in the Oracle Workflow *sql* subdirectory on your server. This script is automatically run by the Oracle Universal Installer when you install the standalone version of Oracle Workflow.

You can customize and rerun this script if necessary. The 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`.

**Note:** The `wfdirouv.sql` script sets each native Oracle user's e-mail address to the user's respective username. As a minimal setup step, you should edit the `wfdirouv.sql` script to either link your native Oracle users to an existing mail directory store through the `WF_ROLES` view definition, or, if the usernames and e-mail account names match, then simply add the domain for your organization, such as '@oracle.com', to the usernames in the `WF_USERS` view definition. Typically, the columns that you change are `EMAIL_ADDRESS` in `WF_USERS` and `EMAIL_ADDRESS` in `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 set up 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.



**Attention:** If you want to implement Oracle Internet Directory (OID) integration for the standalone version of Oracle

Workflow, you must run the *wfdircsv.sql* script to ensure that your directory service views are mapped only to the WF\_LOCAL tables, because only the WF\_LOCAL\_USERS table will be synchronized with OID. (Only users are maintained through OID, not Workflow roles.) In this case, you must not customize the script to incorporate any other tables. After implementing OID integration, you maintain your user information only through OID. See: Synchronizing Workflow Directory Services with Oracle Internet Directory: page 2 – 30.

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 Synchronizing Workflow Directory Services with Oracle Internet Directory

If you are using the standalone version of Oracle Workflow, and you have installed Oracle9i Release 2 or higher or Oracle9iAS Release 2 or higher, you can synchronize the user information in your Workflow directory service with Oracle Internet Directory (OID) using Lightweight Directory Access Protocol (LDAP). This integration is recommended because it enables you to manage and publish user information in a central location which various systems can reference.

Synchronization with OID enables Oracle Workflow to do the following:

- Assign ownership of work items and send notifications to users defined in OID.
- Synchronize with other external user directories that are synchronized with OID.
- Participate in single sign-on through LDAP external authentication with Oracle Portal and Oracle9iAS Single Sign-On Server, if you have installed Oracle9iAS Release 2 or higher. With single sign-on, a user who is logged into any participating Oracle9iAS component is automatically authenticated when accessing any other participating component and does not need to log in again.

**Context:** You need to perform this step only once.

### **Oracle Internet Directory**

---

Oracle Internet Directory is a general purpose directory service that enables fast retrieval and centralized management of information about dispersed users and network resources. It combines Lightweight Directory Access Protocol (LDAP) Version 3 with the high performance, scalability, robustness, and availability of Oracle9i.

LDAP is a standard, extensible directory access protocol. It is a common language that LDAP clients and servers use to communicate. LDAP was conceived as an internet-ready, lightweight implementation of the International Standardization Organization (ISO) X.500 standard for directory services. It requires a minimal amount of networking software on the client side, which makes it particularly attractive for internet-based, thin client applications.

The advantages of OID include:

- Scalability – Oracle Internet Directory exploits the strengths of Oracle9i, enabling support for terabytes of directory information. In addition, such technologies as multithreaded LDAP servers and database connection pooling allow it to support thousands of concurrent clients with subsecond search response times.

Oracle Internet Directory also provides data management tools, such as Oracle Directory Manager and a variety of command-line tools, for manipulating large volumes of LDAP data.

- High availability – Oracle Internet Directory is designed to meet the needs of a variety of important applications. For example, it supports full, multimaster replication between directory servers: If one server in a replication community becomes unavailable, then a user can access the data from another server. Information about changes made to directory data on a server is stored in special tables on the Oracle9i database. These are replicated throughout the directory environment by Oracle9i Replication, a robust replication mechanism.

Oracle Internet Directory also takes advantage of all the availability features of Oracle9i. Because directory information is stored securely in the Oracle9i database, it is protected by Oracle's backup capabilities. Additionally, the Oracle9i database, running with large datastores and heavy loads, can recover from system failures quickly.

- Security – Oracle Internet Directory offers comprehensive and flexible access control. An administrator can grant or restrict access to a specific directory object or to an entire directory subtree. Moreover, Oracle Internet Directory implements three levels of user authentication: anonymous, password-based, and certificate-based using Secure Socket Layer (SSL) Version 3 for authenticated access and data privacy.
- Synchronization with other directories – Oracle Internet Directory includes the Oracle Directory Integration platform that enables you to synchronize with other enterprise repositories, including third-party LDAP directories.

Oracle9iAS Single Sign-On uses Oracle Internet Directory to store user entries. It maps users for any partner application to user entries in OID entries, and authenticates them by using LDAP mechanisms.

## See Also

*Oracle Internet Directory Administrator's Guide*

### **Oracle9iAS Single Sign-On**

---

Oracle9iAS Single Sign-On is a component of Oracle9i Application Server that provides a framework for secure single sign-on, allowing users to log in to multiple Web-based applications by entering a user name and password only once.

Oracle9iAS Single Sign-On provides the following benefits:

- Ease of administration and reduced administrative costs, because user names and passwords can be stored and maintained outside of any particular application and shared across the enterprise
- Convenient login experience, because users do not need to maintain a separate username and password for each application they access
- Increased security, because when the password is only required once, users are less likely to use simple, easy-to-remember passwords or write them down

The core of the Oracle9iAS Single Sign-On technology is the Login Server. The Login Server authenticates users and passes their identities to the partner applications that are integrated with it.

Partner applications support a single sign-on mechanism that enables them to accept a user's username and password as validated by the Login Server. A partner application delegates its authentication to the Login Server. If a partner application is registered with the Login Server, users can log into it using the single sign-on mechanism.

With `mod_osso`, an Oracle module that enables single sign-on, Oracle HTTP Server becomes a partner application of the Login Server. Oracle Workflow uses Oracle HTTP Server as its Web server. If you implement Oracle Internet Directory/Single Sign-On integration, Oracle Workflow participates in single sign-on by using `mod_osso` to authenticate access to its secured web pages.

When a user first tries to access a secured Workflow web page, the Workflow security package `WFA_SEC` checks the CGI environment variable `REMOTE_USER` for user information. If the user is not already logged in to Oracle Workflow or another Oracle9iAS Single Sign-On participating application, the user will be prompted to log in before the page appears.

**Note:** The WFA\_SEC package must be loaded as a post-install step if you choose to implement Oracle Internet Directory/Single Sign-On integration. For more information, see your installation documentation.

To set the variable REMOTE\_USER, Oracle HTTP Server internally calls to mod\_osso. Acting as an Oracle*9i*AS Single Sign-On partner application, mod\_osso transparently redirects the user to the Login Server to obtain authentication credentials, if no application cookie is present.

The Login Server performs the following steps:

- Prompts the user for the user name and password, if no login cookie is present.
- Authenticates the user by means of the user name and password, using external repository authentication that relies on an LDAP-compliant directory, specifically Oracle Internet Directory. The Login Server binds to OID and then looks up the user credentials stored in the directory.
- Stores an encrypted login cookie on the authenticated client.
- Transparently redirects the user to the partner application, mod\_osso, by using a URL with an encrypted parameter containing the user's identity.

Oracle HTTP Server with mod\_osso then performs the following steps:

- Decrypts the parameter.
- Identifies the user.
- Establishes its own session management (for example, determining what, if any, access privileges to grant to the user).
- Sets a partner application cookie so that subsequent user access does not require a redirect to the Login Server.
- Presents the requested application page to the user.

If, during the same session, the user again seeks access to the same or to a different partner application, the Login Server does not prompt the user for a username and password. Instead, the Login Server obtains the information from the login cookie that is already on the client browser. The login cookie provides the Login Server with the user's identity and indicates that authentication has already been performed. If there is no login cookie, the Login Server presents the user with a login page.

To guard against eavesdropping, the Login Server can send the login cookie to the client browser over an encrypted SSL channel.

The login cookie expires with the session, either at the end of a time interval specified by the administrator, or when the user exits the browser. The login cookie is never written to disk.

**Note:** To log out of a partner application and log in as another user, the user must also log out of the Login Server session. Otherwise, the authentication request returns the partner application to the logged in state of the previous user.

## See Also

*Oracle9iAS Single Sign-On Administration Guide*

*Oracle9iAS Single Sign-On Application Developer's Guide*

### **Oracle Internet Directory Synchronization**

---

Oracle Workflow provides APIs to synchronize the user information in your Workflow directory service with OID. These APIs are defined in a PL/SQL package called WF\_LDAP. See: Workflow LDAP APIs: page 8 – 144.

**Note:** OID integration includes only individual users, not user groups. Workflow roles are not maintained through OID.

#### ► **To Synchronize Workflow Directory Services with OID**

1. Ensure that the following PL/SQL packages required for LDAP synchronization are loaded in your database:
  - DBMS\_LDAP—This package contains the functions and procedures which can be used to access data from LDAP servers.
  - WFA\_SEC—This package contains Workflow security functions and procedures.

For the standalone version of Oracle Workflow, installation of these packages is completed as part of the post-installation steps after you install the Oracle Workflow server.

2. For single sign-on integration, ensure that the Database Access Descriptor for Oracle Workflow is protected in the mod\_osso configuration file. For the standalone version of Oracle Workflow,



mod\_osso configuration is completed as part of the post-installation steps after you install the Oracle Workflow server.

3. Ensure that the following global Workflow preferences are set to the appropriate information for your OID installation. See: To Set Global User Preferences: page 2 – 14.
  - LDAP Host
  - LDAP Port
  - LDAP User Name
  - LDAP Password
  - LDAP Changelog Base Directory
  - LDAP User Base Directory
4. Run the *wfdircsv.sql* script to map your directory service views only to the WF\_LOCAL tables. See: Integrating Oracle Workflow Directory Services with Local Workflow Users: page 2 – 28.
5. Enable the predefined subscription to the oracle.apps.wf.public.user.change event with the rule function *WF\_SSO.user\_change*. See: User Entry Has Changed Event: page 14 – 15 and To Update or Delete an Event Subscription: page 13 – 52.
6. To begin the synchronization, run the *WF\_LDAP.Synch\_all( )* API. This function retrieves all the existing user information from OID, based on the LDAP directory information specified in the Global Workflow Preferences, and raises the oracle.apps.wf.public.user.change event. The predefined subscription to this event that you enabled loads the user information into the WF\_LOCAL\_USERS table.

Because *Synch\_all( )* retrieves information for all users stored in OID, you should use this function only once during setup. If necessary, however, you can also run *Synch\_all( )* as required for recovery or cleanup.

Use the following commands to run *Synch\_all( )*:

```
declare
    res boolean := FALSE;
begin
    wf_log_pkg.WF_DEBUG_FLAG := TRUE;

    res := wf_ldap.synch_all();
    if (res) then
```

```

        dbms_output.put_line('succeeded');
    else
        dbms_output.put_line('failed ');
    end if;
end;
/

```

7. Subsequently, you must maintain the synchronization between your Workflow directory service and OID by retrieving and loading only changed OID user information. It is recommended that you update the user information every ten minutes.

You can use either *WF\_LDAP.Synch\_changes()* or *WF\_LDAP.Schedule\_changes()* to retrieve changed user information from OID. *WF\_LDAP.Synch\_changes()* identifies LDAP user changes in OID, including creation, modification, and deletion, by querying the LDAP change log records. The function connects to OID based on the LDAP directory information specified in the Global Workflow Preferences. If there is a change, the function retrieves the user information from OID and raises the oracle.apps.wf.public.user.change event. The predefined subscription to this event that you enabled loads the changed user information into the WF\_LOCAL\_USERS table. You can use *WF\_LDAP.Synch\_changes()* to perform a single update.

To continue updating user information periodically, use *WF\_LDAP.Schedule\_changes()*. This procedure submits a database job using the DBMS\_JOB utility to run *WF\_LDAP.Synch\_changes()* repeatedly at an interval that you specify. The default interval, which is also the recommended frequency to check for updates, is ten minutes.

You can create a script to run *WF\_LDAP.Schedule\_changes()*. For example, to run the API at an interval of ten minutes, create a SQL file with the following commands:

```

declare
begin
    wf_log_pkg.WF_DEBUG_FLAG := TRUE;
    wf_ldap.schedule_changes(0,0,10);
end;
/

```

Then run SQL\*Plus and load your new script to the database.

**Note:** You must terminate the running of any WF\_LDAP APIs before changing your LDAP setup, such as by migrating to a different LDAP server.



**Attention:** If you implement OID integration, you must maintain your users only through OID. You must not create ad hoc users in the WF\_LOCAL\_USERS table, because you risk discrepancies in your user information and unpredictable results if you use any tool other than OID to maintain users after integrating with OID. Consequently, if you implement OID integration, you must not use the *CreateAdHocUser()*, *SetAdHocUserStatus()*, *SetAdHocUserExpiration()*, or *SetAdHocUserAttr()* APIs in the WF\_DIRECTORY package.

You can still use ad hoc roles, however, since Workflow roles are not maintained through OID.

## See Also

Setting Global User Preferences: page 2 – 14  
Workflow LDAP APIs: page 8 – 144  
User Entry Has Changed Event: page 14 – 15  
Managing Job Queues, *Oracle Administrator's Guide*  
Workflow Directory Service APIs: page 8 – 121

## Step 5 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: Oracle Database National Language Support Guide

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

1. Install Oracle Workflow.
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 16 – 12.
4. Run the script WFNLADD.sql to create rows for the enabled language in each workflow object translation table. See: WFNLADD.sql: page 16 – 5.
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:** 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 – 14 and Setting User Preferences: page 9 – 6.

► **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 – 69.
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 – 14.

## Step 6 Setting the Socket Listener Profile Options

The Notification Details web page can 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 choose 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 1.1.7.27.)
```

Run the Workflow Resource Generator to load the contents of wfcfg.msg into the WF\_RESOURCES table.

You can also set these three values in the Global Preferences page. See: To Set Global User Preferences: page 2 – 14.

You must also set the Socket Listener Port profile option to the port at which Oracle Workflow should launch attached forms. This profile option can be set to different ports for different users.

If the socket listener port is not set at user level, Oracle Workflow launches attached forms at the default port set for the site. However, if users have set different ports, Oracle Workflow launches the forms for each user at the specified port. By using different socket listener ports, two different users logged into Oracle Applications on the same machine can both launch attached forms at the same time without interference from each other.

**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 – 105.

## 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 this 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 environment variable 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 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, and 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.

Additionally, a background engine must be set up to handle stuck processes. A process is identified as stuck when it has a status of ACTIVE, but cannot progress any further. For example, a process could become stuck in the following situations:

- A thread within a process leads to an activity that is not defined as an End activity but has no other activity modeled after it, and no other activity is active.
- A process with only one thread loops back, but the pivot activity of the loop has the On Revisit property set to Ignore.
- An activity returns a result for which no eligible transition exists. For instance, if the function for a function activity returns an unexpected result value, and no default transition is modeled after that activity, the process cannot continue.

The background engine sets the status of a stuck process to ERROR:#STUCK and executes the error process defined for it.

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).

Ensure that you have at least one background engine that can check for timed out activities, one that can process deferred activities, and one that can handle stuck processes. At a minimum, you need to set up one background engine that can handle both timed out and deferred activities as well as stuck processes.

Generally, you should run a separate background engine to check for stuck processes at less frequent intervals than the background engine that you run for deferred activities, normally not more often than once a day. Run the background engine to check for stuck processes when the load on the system is low.

**Context:** You need to perform this step only once.

See: Activity Cost: page 4 – 47

See: Timeout Transitions: page 5 – 3

See: Deferring Activities: page C – 7

### ► **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. You can use the procedures in the `DBMS_JOB` package to schedule and manage the background engine as a database job. See: Background: page 8 – 41 and *Managing Job Queues, Oracle Administrator's Guide*.

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 – 45

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager to submit and manage the Workflow Background

Process concurrent program. For more information, please refer to the Oracle Applications Manager online help.

If you are using the standalone version of Oracle Workflow available with Oracle9i Release 2, you can use the standalone Oracle Workflow Manager component available through Oracle Enterprise Manager to submit and manage Workflow background engine database jobs. For more information, please refer to the Oracle Workflow Manager online help.

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

### **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.

**Note:** If you require a larger rollback segment for the Workflow Background Process than the default, you can use the Concurrent Programs window in the System Administrator responsibility to specify the rollback segment that you want. This rollback segment will be used instead of the default and will be used up until the first commit.

Query the Workflow Background Process concurrent program (FNDWFBG) in the Concurrent Programs window, and choose the Session Control button. Then in the Session Control window, enter the rollback segment you want in the Rollback Segment field, and save your work. See: Concurrent Programs Window, *Oracle Applications System Administrator's Guide*.

### **► 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:

<b>Item Type</b>	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.
<b>Minimum Threshold</b>	Specify the minimum cost that an activity must have for this background engine to execute it, in hundredths of a second.
<b>Maximum Threshold</b>	Specify the maximum cost that 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.
<b>Process Deferred</b>	Specify whether this background engine checks for deferred activities. Setting this parameter to 'Yes' allows the engine to check for deferred activities.
<b>Process Timeout</b>	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.
<b>Process Stuck</b>	Specify whether this background engine checks for stuck processes. Setting this parameter to 'Yes' allows the engine to check for stuck processes.

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

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

## See Also

See: Overview of Concurrent Programs and Requests, *Oracle Applications System Administrator'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 Workflow Engine threshold is set to 50 as a default. 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 9 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, dequeues these messages from the SMTP advanced queue, 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 or any MAPI-compliant mail application on Windows NT.

**Note:** A standard agent named WF\_SMTP\_O\_1\_QUEUE is defined for the Notification Mailer SMTP queue in the Event Manager. This agent appears in the Check Setup page and the Event System Local Queues page of the Event Manager, enabling you to use these pages to check the number of notification messages on the Notification Mailer queue. The WF\_SMTP\_O\_1\_QUEUE agent is not used by the Business Event System, however. See: Standard Agents: page 13 – 25, Checking the Business Event System Setup: page 13 – 53, and Reviewing Local Queues: page 13 – 72.

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 – 69.

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 – 67. 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 – 6.

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 – 50.
- 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 – 51.

 **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 – 53.
- 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 – 12

See: Reviewing a Summary of Your Notifications via Electronic Mail: page 10 – 24

### **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 and PL/SQL CLOB document attributes are token replaced with a plain text version of a PL/SQL document.
- 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 – 12.

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 and PL/SQL CLOB document attributes are token replaced with HTML text or plain text between `<pre>...</pre>` HTML tags.

- 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, an attachment called Notification References is appended to the message. This attachment includes a link to each URL attribute for the message that has Attach Content checked. You can navigate to a URL by choosing its link. The Notification Mailer does not have any special handling of image, video, or audio URL content.
- If the message attribute is a PL/SQL or PL/SQL CLOB document attribute, the fully generated PL/SQL document is fetched and attached to the message.

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.

**Note:** If the SEND\_ACCESS\_KEY parameter in the Notification Mailer configuration file is set to N, and you are not already logged in, you will be prompted to log in when you select the Notification Detail Link before you can access the Notification Details web page.

**Note:** The file name of the Notification Detail Link attachment is determined by the text value for the WF\_URL\_NOTIFICATION resource token, or by the token name if no text value is defined. Similarly, the file name of the Notification References attachment is determined by the text value for the WF\_URLLIST\_ATTACHMENT resource token, or by the token name if no text value is defined. The default file names are "Notification Detail Link.html" and "Notification References.html", respectively. If you want to specify different file names for these attachments, you must first create a .msg source file specifying the new file names as the text values for the WF\_URL\_NOTIFICATION and WF\_URLLIST\_ATTACHMENT resource tokens. Then use the Workflow Resource Generator program to generate a new

Oracle Workflow resource file (*wf<language>.res*) from the source file and to upload the new seed data from the source file to the database table WF\_RESOURCES. See: To Run the Workflow Resource Generator: page 8 – 105 and Setting the WF\_RESOURCES Environment Variable: page 2 – 42.

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 – 9.

### **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 – 50 and Sending HTML-Formatted E-mail: page 2 – 51.

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, an attachment called Notification References is appended to the message. This attachment includes a link to each URL attribute for

the message that has Attach Content checked. You can navigate to a URL by choosing its link.

- If the message attribute is a PL/SQL or PL/SQL CLOB document attribute, the fully generated PL/SQL document is fetched and attached to the message.

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.

**Note:** If the SEND\_ACCESS\_KEY parameter in the Notification Mailer configuration file is set to N, and you are not already logged in, you will be prompted to log in when you select the Notification Detail Link before you can access the Notification Details web page.

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.
- Select the HTML Message Body attachment to display the HTML-formatted version of the e-mail message, 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.

**Note:** The file name of the HTML Message Body attachment is determined by the text value for the WF\_HTML\_MESSAGE resource token, or by the token name if no text value is defined. Similarly, the file name of the Notification Detail Link attachment is determined by the text value for the WF\_URL\_NOTIFICATION resource token, or by the token name if no text value is defined; and the file name of the Notification References attachment is determined by the text value for the WF\_URLLIST\_ATTACHMENT resource token, or by the token name if no text value is defined. The default file

names are "HTML Message Body.html", "Notification Detail Link.html", and "Notification References.html", respectively. If you want to specify different file names for these attachments, you must first create a .msg source file specifying the new file names as the text values for the WF\_HTML\_MESSAGE, WF\_URL\_NOTIFICATION, and WF\_URLLIST\_ATTACHMENT resource tokens. Then use the Workflow Resource Generator program to generate a new Oracle Workflow resource file (wf<language>.res) from the source file and to upload the new seed data from the source file to the database table WF\_RESOURCES. See: To Run the Workflow Resource Generator: page 8 – 105 and Setting the WF\_RESOURCES Environment Variable: page 2 – 42.

## Starting the Notification Mailer

---

You can install and set up the Notification Mailer to run against UNIX Sendmail 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 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 UNIX Sendmail can provide a gateway to.

**Note:** If you are using the standalone version of Oracle Workflow available with Oracle9i Release 2, you can use the standalone Oracle Workflow Manager component available through Oracle Enterprise Manager to monitor the throughput of the Notification Mailer. For more information, please refer to the Oracle Workflow Manager online help.

### ► To Start the Notification Mailer for UNIX Sendmail

#### 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 `snd` to use the UNIX Sendmail version of the Notification Mailer. Replace `<config_file>` with the full path and

name of the configuration file that contains the parameters you want to run with the Notification Mailer.

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

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

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

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

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

### **For the Version of Oracle Workflow Embedded in Oracle Applications:**

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

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager to configure and run the Notification Mailer. For more information, please refer to the Oracle Applications Manager online help.

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 Universal 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\wfmlr.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 – *<SID NAME>* 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:

```
wfmlr.exe <arg1> <arg2> ...
```

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

```
wfmlr.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.

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager to configure and run the Notification Mailer. For more information, please refer to the Oracle Applications Manager online help.

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*).



**Attention:** The CONNECT parameter is not required if you submit the Notification Mailer as a concurrent program from the version of Oracle Workflow embedded in Oracle Applications. In this case, the concurrent manager passes in the database connection information for the Notification Mailer to use.



- **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 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. The node name is included with the outgoing notification ID. The default name is `wf`.

If you have multiple workflow databases on one server, you must define a unique node name for each Notification Mailer configuration file to ensure that the Notification Mailer for each instance uses a different `TMP` file for processing. For example, if you already have a Notification Mailer on the server with the node name `wf`, you could assign the node name `wf2` to the Notification Mailer for the next instance you set up.

**Note:** You can only run one Notification Mailer per instance, and each Notification Mailer must use a separate e-mail account to process responses.

- **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`.



**Attention:** The `FROM` parameter is only used to set the `From:` field value by the UNIX Sendmail version of the Notification Mailer. The MAPI-compliant version of the Notification Mailer uses the display value that is set up for the mail account specified in the `ACCOUNT` parameter.

- **SUMMARYONLY**—(Required) Indicate whether this Notification Mailer processes only notifications assigned to users/roles with a notification preference of 'SUMMARY' or

whether it only processes notifications for users/roles with a notification preference of 'MAILTEXT', '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 – 21.

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 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.



**Attention:** If you include the HTML\_MAIL\_TEMPLATE parameter in your configuration file, the Notification Mailer will ignore the DIRECT\_RESPONSE parameter when sending messages to users with a notification preference of 'MAILATTH'. Instead, all users with this notification preference will receive the plain text message body defined for the Workflow Open Mail for Outlook Express template, which uses the templated response method. Consequently, if you do not need to use the Workflow Open Mail for Outlook Express template, you should comment out the

HTML\_MAIL\_TEMPLATE parameter to let the DIRECT\_RESPONSE parameter take effect.

- **AUTOCLOSE\_FYI**—Indicate whether this Notification Mailer automatically closes notifications that do not require a response, such as FYI (For Your Information) notifications, after sending the notifications by electronic mail. Valid values include Y or N. The default is Y.

If the value is N, all FYI notifications will remain open in the Notifications Worklist until users manually close these notifications.

- **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.



**Attention:** The `REPLYTO` parameter is only used to set the reply-to address by the UNIX Sendmail version of the Notification Mailer. The MAPI-compliant version of the Notification Mailer uses the display value that is set up for the mail account specified in the `ACCOUNT` parameter.

- **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 – 14.

- **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`.

**Note:** For the UNIX Sendmail version of the Notification Mailer, the DISCARD value must always be the full path name of a mail file.

- **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`.

**Note:** For the UNIX Sendmail version of the Notification Mailer, the PROCESS value must always be the full path name of a mail file.

- **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`.

**Note:** For the UNIX Sendmail version of the Notification Mailer, the UNPROCESS value must always be the full path name of a mail file.

- **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 – 82.

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 status and activity status are updated to ERROR. Ideally, the workflow administrator corrects the problem and resends the notification by updating its mail status to MAIL.
- 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. This 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 – 79.



**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`.

- **RESET\_FAILED**—Indicate whether this Notification Mailer should reset all notifications with a mail status of FAILED to a mail status of MAIL when the Notification Mailer is started. Valid values include Y or N. The default is N.

If the value is Y, the Notification Mailer will reset all FAILED notifications to a mail status of MAIL on startup and then attempt to process these notifications as usual.

- **RESET-NLS**—Indicate whether the Notification Mailer should convert the NLS codeset for a notification message according to the notification recipient’s preferences before composing the message. Valid values include Y or N. The default is N.

If the value is Y, the Notification Mailer will convert the message to the codeset listed in the `WF_LANGUAGES` table for the language and territory specified in the recipient’s Workflow user preferences. If no preferred territory is specified, the Notification Mailer will use the codeset associated with the first entry it encounters for the user’s preferred language. If neither a language nor a territory is specified in the user preferences, the Notification Mailer will use the codeset seeded in `WF_LANGUAGES` for the language AMERICAN and territory AMERICA. See: Setting User Preferences: page 9 – 6.

- **HTML\_MAIL\_TEMPLATE**—Specify `OPEN_MAIL_OUTLOOK` to use the Workflow Open Mail for Outlook Express message as the

template for e-mail notifications that require a response, for users with a notification preference of 'MAILHTML' or 'MAILATTH'. You can select this message template if you use an e-mail application such as Microsoft Outlook Express as your e-mail client, in order to include a link to the Notification Details web page which lets users respond to the notification there. This template is provided to accommodate e-mail applications that cannot process the response links included in the Workflow Open Mail (Templated) and Workflow Open Mail (Direct) templates.

If you want to use the normal Workflow Open Mail (Templated) and Workflow Open Mail (Direct) templates, you can comment out this parameter in your configuration file and let the Notification Mailer use these templates by default.



**Attention:** If you include the `HTML_MAIL_TEMPLATE` parameter in your configuration file, the Notification Mailer will ignore the `DIRECT_RESPONSE` parameter when sending messages to users with a notification preference of 'MAILATTH'. Instead, all users with this notification preference will receive the plain text message body defined for the Workflow Open Mail for Outlook Express template, which uses the templated response method. Consequently, if you do not need to use the Workflow Open Mail for Outlook Express template, you should comment out the `HTML_MAIL_TEMPLATE` parameter to let the `DIRECT_RESPONSE` parameter take effect.

**Note:** The `HTML_MAIL_TEMPLATE` parameter does not apply to users with a notification preference of 'MAILTEXT'.

- **SEND\_ACCESS\_KEY**—Specify `Y` to include an access key in the Notification Detail Link attachment that is sent with HTML e-mail notifications and with plain text e-mail notifications with HTML attachments. The access key allows users to access the Notification Details web page directly by clicking the Notification Detail Link, whether they are currently logged in or not. However, if users are not already logged in, they cannot access any other notifications except the notification with which the attachment was sent. Specify `N` to exclude the access key from the Notification Detail Link. When users click the link without the access key, they are prompted to log in, if they have not already done so, before they can access the Notification Details web page. The default is `Y`.

See: Defining Rules for Automatic Notification Processing: page 10 – 25



## ► 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 Notification Mailer. The script is called *wfmail.csh* and it is located in the Oracle Home *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 10 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 – 58.

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.

<b>START_DATE</b>	The date the message is sent.
<b>TO</b>	The role the notification is sent to; the performer.
<b>SUBJECT</b>	The subject line defined in the message.
<b>BODY</b>	The text of the body defined in the message.
<b>COMMENT</b>	Comments added by the sender or the forwarder.
<b>PRIORITY</b>	The priority of the notification message.
<b>DUE_DATE</b>	The date by which a response is required, specified in the notification activity.
<b>NOTIFICATION</b>	Required notification code used to identify the information in the notification.
<b>RESPONSE</b>	The user response section as defined by the Respond message attributes in the actual notification message definition.
<b>MAILTO</b>	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.
<b>CLICK_HERE_RESPONSE</b>	The content of the HTML tag that a recipient would click on to access the Notification Details page to respond to a notification. This attribute is not currently used.

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
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 – 58.*

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.

<b>START_DATE</b>	The date the message is sent.
<b>TO</b>	The role the notification is sent to; the performer.
<b>SUBJECT</b>	The subject line defined in the message.
<b>BODY</b>	The text of the body defined in the message.
<b>COMMENT</b>	Comments added by the sender or the forwarder.
<b>PRIORITY</b>	The priority of the notification message.
<b>DUE_DATE</b>	The date by which a response is required, specified in the notification activity.
<b>NOTIFICATION</b>	Required notification code used to identify the information in the notification.
<b>RESPONSE</b>	The user response section as defined by the Respond message attributes in the actual notification message definition.
<b>MAILTO</b>	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.
<b>CLICK_HERE_RESPONSE</b>	The content of the HTML tag that a recipient would click on to access the Notification Details page to respond to a notification. This attribute is not currently used.

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
quotes.</B>
<P>&MAILTO
</BODY>
</HTML>
```

## Workflow Open Mail for Outlook Express Message

---

If you use an e-mail application such as Microsoft Outlook Express as your e-mail client, you should select the Workflow Open Mail for Outlook Express message as a template for e-mail notifications that require a response, for users with a notification preference of MAILHTML or MAILATTH. This message includes a link to the Notification Details web page to let users respond to the notification there. This template is provided to accommodate e-mail applications that cannot process the response links included in the Workflow Open Mail (Templated) and Workflow Open Mail (Direct) templates.

**Note:** To select the Workflow Open Mail for Outlook Express message as the message template for users with a notification preference of MAILHTML or MAILATTH, set `HTML_MAIL_TEMPLATE=OPEN_MAIL_OUTLOOK` in the configuration file for the Notification Mailer. See: *To Create a Configuration File for the Notification Mailer: page 2 – 58.*



**Attention:** If you include the `HTML_MAIL_TEMPLATE` parameter in your configuration file, the Notification Mailer will ignore the `DIRECT_RESPONSE` parameter when sending messages to users with a notification preference of 'MAILATTH'. Instead, all users with this notification preference will receive the plain text message body defined for the Workflow Open Mail for Outlook Express template, which uses the templated response method. Consequently, if you do not need to use the Workflow Open Mail for Outlook Express template, you should comment out the `HTML_MAIL_TEMPLATE` parameter to let the `DIRECT_RESPONSE` parameter take effect.

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 MAILATTH. The HTML-formatted message body includes a link called "Click here to respond" which lets users access the notification in the Notification Details web page to respond to the notification. 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:** When users choose the "Click here to respond" link, it automatically attempts to establish a web session with the web server. Users must be able to connect to the web server to use this link to respond to a notification. See: *Reviewing Notifications via Electronic Mail: page 10 – 2.*



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

<b>START_DATE</b>	The date the message is sent.
<b>TO</b>	The role the notification is sent to; the performer.
<b>SUBJECT</b>	The subject line defined in the message.
<b>BODY</b>	The text of the body defined in the message.
<b>COMMENT</b>	Comments added by the sender or the forwarder.
<b>PRIORITY</b>	The priority of the notification message.
<b>DUE_DATE</b>	The date by which a response is required, specified in the notification activity.
<b>NOTIFICATION</b>	Required notification code used to identify the information in the notification.
<b>RESPONSE</b>	The user response section as defined by the Respond message attributes in the actual notification message definition.
<b>MAILTO</b>	The content of the HTML tag that a recipient would click on to respond to a notification. This attribute is not currently used.
<b>CLICK_HERE_RESPONSE</b>	The content of the HTML tag that a recipient would click on to access the Notification Details page 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 for Outlook Express 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 in your reply. 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>&CLICK_HERE_RESPONSE
</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.

<b>START_DATE</b>	The date the message is sent.
<b>TO</b>	The role the notification is sent to; the performer.
<b>SUBJECT</b>	The subject line defined in the message.
<b>BODY</b>	The text of the body defined in the message.
<b>COMMENT</b>	Comments added by the sender or the forwarder.
<b>PRIORITY</b>	The priority of the notification message.
<b>DUE_DATE</b>	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 URL Attachment Message**

---

The Notification system uses the Workflow URL Attachment message as a template to create the Notification References attachment for HTML-formatted notification messages that include URL attributes with Attach Content checked. The template includes a list with links to each URL.

The Workflow URL Attachment message has the following message attribute. The value is drawn from the message definition associated with a notification activity.

**URLLIST** The list of URLs to be included in the attachment.

You can customize the text that appears in the body of the Workflow URL Attachment template, where an attribute preceded by an ampersand (&) is token substituted with a runtime value when the notification is sent. The boilerplate text for the HTML-formatted message body is as follows:

```
<HTML> <HEAD> <TITLE> Oracle Workflow Notification
References </TITLE> </HEAD>
<BODY BGCOLOR="#FFFFFF" >
```

```

<P><B><FONT SIZE=+1>Notification References</FONT> </B>

<HR WIDTH="100%">
<BR>
&URLLIST
<BR>
<HR WIDTH="100%">
<BR>&nbsp;
</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:

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

<b>START_DATE</b>	The date the original message was sent.
<b>TO</b>	The role the notification is sent to; the performer.
<b>SUBJECT</b>	The subject line of the original message.
<b>BODY</b>	The text of the original message.
<b>COMMENT</b>	Comments added by the sender or the forwarder.
<b>PRIORITY</b>	The priority of the notification message.
<b>DUE_DATE</b>	The date by which a response is required, specified by the notification activity.
<b>NOTIFICATION</b>	Required notification code used to identify the information in the notification.
<b>RESPONSE</b>	The user response section as defined by the Respond message attributes in the original message definition.
<b>MAIL_ERROR_MESSAGE</b>	An error message that the mail program generates if an error occurs upon processing the response.
<b>MAIL_ERROR_STACK</b>	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.
<b>CLICK_HERE_RESPONSE</b>	The content of the HTML tag that a recipient would click on to access the Notification Details page to respond to a notification. This attribute is not currently used.

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 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:

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

<b>SUMMARY</b>	Summary report.
<b>USER_NAME</b>	The user/role the notification summary is sent to; the performer.
<b>SYSDATE</b>	The current date.

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

```
Summary of Notifications for '&USER_NAME'
(Please use the Notifications web page 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 Notifications web page 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:

<b>UBODY</b>	The text of the unsolicited mail message body.
<b>USUBJECT</b>	The text of the unsolicited mail subject line.
<b>UFROM</b>	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 11 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 HTTP 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 1.1.8 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 **FNDLOGOS.gif** if you are using Oracle Workflow embedded in Oracle Applications or **WFLOGO.gif** if you are using the standalone version of Oracle Workflow.
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 12 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 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 the physical directory that your web server's */OA\_MEDIA/* virtual directory points to, so that the Workflow Monitor can access them:

**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 **Setting Up the Java Function Activity Agent**

To execute external Java function activities, you must set up the Java Function Activity Agent. This functionality is currently only available for the standalone version of Oracle Workflow. The Java Function Activity Agent dequeues the messages related to external Java activities from the 'Outbound' queue for external function processing, calls the appropriate Java functions, and places the results on the 'Inbound' queue for external function processing.

**Note:** These 'Outbound' and 'Inbound' queues are separate from the queues used for the Business Event System. See: Workflow Queue APIs: page 8 – 162.

After a Java function completes, you must run a background engine to process the 'Inbound' queue and complete the function activity. See: Setting Up Background Engines: page 2 – 43.

Some standard Workflow activities are external Java function activities and require the Java Function Activity Agent. You can also define your own external Java function activities. See: Standard Activities: page 6 – 2, To Create a Function Activity: page 4 – 50, and Standard API for Java Procedures Called by Function Activities: page 7 – 8.

To run the Java Function Activity Agent, you must have Java Runtime Environment (JRE) Version 1.1.8, or a higher 1.1.x version, installed.

**Note:** The Java Runtime Environment is available for download from <http://www.javasoft.com>.

**Context:** You need to perform this step only once.

### **Starting the Java Function Activity Agent**

---

If you are using the standalone version of Oracle Workflow, you can run scripts provided by Oracle Workflow to start the Java Function Activity agent. You can also start the agent manually.

When you start the Java Function Activity Agent, you must specify the database connection details. You can also optionally specify the character set and the JDBC driver type that you want to use.

You use different commands to start the agent depending on whether you are running it from a script or manually, which platform you are running it on, and which options you want to specify.

## Starting the Java Function Activity Agent From a Script

---

If you are using the standalone version of Oracle Workflow, you can run scripts called `wfjvlsnr.csh` or `wfjvlsnr.bat` to start the Java Function Activity Agent on UNIX or on Windows NT, respectively. These scripts are located on your server in the Oracle Workflow *admin* subdirectory.

If you define your own external Java function activities, you must edit the scripts to include the path to the JAR files containing your custom Java classes. The custom class files should reside on the same platform where the Java Function Activity Agent is run. The Java Function Activity Agent does not need to reside on the same tier as the database, however.

You can use commands with different syntax to run the scripts, depending on your platform and the options you want to specify.

### Running the `wfjvlsnr.csh` Script on UNIX

For example, you can use the following command to run the `wfjvlsnr.csh` script on UNIX, if you want to use the default JDBC OCI8 driver:

```
wfjvlsnr.csh <user_name>/<password>@<connect_string>  
[<character_set>]
```

Replace the parameters in the command as follows:

- `<user_name>`—The user name of your Oracle Workflow database account.
- `<password>`—The password for your Oracle Workflow database account.
- `<connect_string>`—The connect string for the database. Since this command uses the default JDBC OCI8 driver, the connect string should be the database name as specified in its TNSNAMES entry.
- `<character_set>`—The character set to use for the database session. If you do not specify a character set, Oracle Workflow uses UTF8 by default.

You can also use the following command to run the `wfjvlsnr.csh` script on UNIX, if you want to specify the JDBC driver type to use:

```
wfjvlsnr.csh "<user_name> <password> <connect_string>  
[<JDBC_driver>]" [<character_set>]
```

Replace the parameters in the command as follows:

- *<user\_name>*—The user name of your Oracle Workflow database account.
- *<password>*—The password for your Oracle Workflow database account.
- *<connect\_string>*—The connect string for the database. The format of the connect string depends on the JDBC driver type.
  - For a JDBC OCI8 driver, the connect string should be the database name as specified in its TNSNAMES entry, in the following format:  
*<database\_name>*
  - For a JDBC THIN driver, you can use two different types of connect string. For the first type, the connect string should include the host name, port number, and database system identifier (SID) in the following format:  
*<host\_name>:<port\_number>:<database\_SID>*  
  
For the second type, the connect string should include an Oracle Net name–value pair with the host name, protocol, port number, and SID in the following format:  
*(description=(address=(host=<host\_name>)(protocol=<protocol>)(port=<port\_number>))(connect\_data=(sid=<database\_SID>)))*
- *<JDBC\_driver>*—The JDBC driver type you want to use to connect to the database. The JDBC driver type can be either *oci8* or *thin*. If you do not specify a character set, Oracle Workflow uses the JDBC OCI8 driver by default.
 

**Note:** The connection details, including the user name, password, connect string, and JDBC driver type, must be enclosed in double quotes to separate them from the character set parameter.
- *<character\_set>*—The character set to use for the database session. If you do not specify a character set, Oracle Workflow uses UTF8 by default.

### Running the wfjvlsnr.bat Script on Windows NT

On Windows NT, you can use the following command to run the wfjvlsnr.bat script, if you want to use the default JDBC OCI8 driver:

```
wfjvlsnr.bat <user_name>/<password>@<connect_string>
[<character_set>]
```

Replace the parameters in the command as follows:

- *<user\_name>*—The user name of your Oracle Workflow database account.
- *<password>*—The password for your Oracle Workflow database account.
- *<connect\_string>*—The connect string for the database. Since this command uses the default JDBC OCI8 driver, the connect string should be the database name as specified in its TNSNAMES entry.
- *<character\_set>*—The character set to use for the database session. If you do not specify a character set, Oracle Workflow uses UTF8 by default.

You can also use the following command to run the wfjvlsnr.bat script on Windows NT, if you want to specify the JDBC driver type to use:

```
wfjvlsnr.bat "<user_name> <password> <connect_string>
[<JDBC_driver>]" [<character_set>]
```

Replace the parameters in the command as follows:

- *<user\_name>*—The user name of your Oracle Workflow database account.
- *<password>*—The password for your Oracle Workflow database account.
- *<connect\_string>*—The connect string for the database. The format of the connect string depends on the JDBC driver type.
  - For a JDBC OCI8 driver, the connect string should be the database name as specified in its TNSNAMES entry, in the following format:  
*<database\_name>*
  - For a JDBC THIN driver, you can use two different types of connect string. For the first type, the connect string should include the host name, port number, and database system identifier (SID) in the following format:  
*<host\_name>:<port\_number>:<database\_SID>*

For the second type, the connect string should include an Oracle Net name–value pair with the host name, protocol, port number, and SID in the following format:

```
(description=(address=(host=<host_name>) (protocol=<protocol>) (port=<port_number>)) (connect_data=(sid=<database_SID>)))
```

- `<JDBC_driver>`—The JDBC driver type you want to use to connect to the database. The JDBC driver type can be either `oci8` or `thin`. If you do not specify a character set, Oracle Workflow uses the JDBC OCI8 driver by default.

**Note:** The connection details, including the user name, password, connect string, and JDBC driver type, must be enclosed in double quotes to separate them from the character set parameter.

- `<character_set>`—The character set to use for the database session. If you do not specify a character set, Oracle Workflow uses UTF8 by default.

### **Starting the Java Function Activity Agent Manually**

---

To start the Java Function Activity Agent manually, run JRE against `oracle.apps.fnd.wf.WFFALsnr`, specifying your CLASSPATH, the user name and password of your Oracle Workflow database account, and the database connect string. You can also optionally specify the character set and the JDBC driver type that you want to use.

The CLASSPATH must point to the Java Runtime Environment, the directory containing the Workflow JAR files, the Oracle XML parser, the Oracle JDBC implementation, and the following Workflow JAR files:

- `wfjava.jar`—The Java Function Activity Agent.
- `wfapi.jar`—Workflow Java APIs.
- The Share JAR file—Utilities referenced by the Workflow Java APIs. In the standalone version of Oracle Workflow with Oracle9i, this file is named `share-<version>.jar`, such as `share-1_1_9.jar`, or whichever version is current. In the version of Oracle Workflow embedded in Oracle Applications, this file is named `fndbalishare.jar`.
- The Ewt JAR file—Utilities referenced by the Workflow Java APIs. In the standalone version of Oracle Workflow with Oracle9i, this file is named `ewt-<version>.jar`, such as `ewt-3_3_18.jar`, or whichever version is current. In the version of Oracle Workflow embedded in Oracle Applications, this file is named `fndewt.jar`.
- The Swing JAR file—Optional additional utilities. In the standalone version of Oracle Workflow with Oracle9i, this file is named `swingall-<version>.jar`, such as `swingall-1_1_1.jar`, or whichever version is current. In the version of Oracle



Workflow embedded in Oracle Applications, this file is named `fndswing.jar`.

**Note:** In the standalone version of Oracle Workflow with Oracle9i, the Workflow JAR files are located in the `<ORACLE_HOME>/jlib` directory. In the version of Oracle Workflow embedded in Oracle Applications, the Workflow JAR files are located in the `<ORACLE_HOME>/wf/java/oracle/apps/fnd/wf/jar/` directory.

If you define your own external Java function activities, you must also include the JAR files containing your custom Java classes in the CLASSPATH. The custom class files should reside on the same platform where the Java Function Activity Agent is run. The Java Function Activity Agent does not need to reside on the same tier as the database, however.

You can use commands with different syntax to start the Java Function Activity Agent manually, depending on your platform and the options you want to specify.

### Starting the Java Function Activity Agent on UNIX

For example, you can use the following command to start the Java Function Activity Agent on UNIX, if you want to use the default JDBC OCI8 driver:

```
jre -classpath
"$<JREPATH>/rt.jar:$<Workflow_JAR_file_directory>:
$<Workflow_JAR_file_directory>/wfjava.jar:$<ORACLE_HOME>/wf/
xml/java/lib/xmlparserv2.jar:$<Workflow_JAR_file_directory>/
wfapi.jar:$<ORACLE_HOME>/jdbc/lib/classes111.zip:
$<Workflow_JAR_file_directory>/<Share_JAR_file>:
$<Workflow_JAR_file_directory>/<Ewt_JAR_file>:
$<Workflow_JAR_file_directory>/<Swing_JAR_file>:"
[-DCHARSET=<character_set>] oracle.apps.fnd.wf.WFFALsnr
<user_name>/<password>@<connect_string>
```

In this command, you can optionally use the `-DCHARSET` option to specify the character set that you want to use. If you do not specify a character set, Oracle Workflow uses UTF8 by default.

Replace the parameters in the command as follows:

- `<character_set>`—The character set to use for the database session.
- `<user_name>`—The user name of your Oracle Workflow database account.

- *<password>*—The password for your Oracle Workflow database account.
- *<connect\_string>*—The connect string for the database. Since this command uses the default JDBC OCI8 driver, the connect string should be the database name as specified in its TNSNAMES entry.

You can also use the following command to start the Java Function Activity Agent on UNIX, if you want to specify the JDBC driver type to use:

```
jre -classpath
"$<JREPATH>/rt.jar:$<Workflow_JAR_file_directory>:
$<Workflow_JAR_file_directory>/wfjava.jar:$<ORACLE_HOME>/wf/
xml/java/lib/xmlparserv2.jar:$<Workflow_JAR_file_directory>/
wfapi.jar:$<ORACLE_HOME>/jdbc/lib/classes111.zip:
$<Workflow_JAR_file_directory>/<Share_JAR_file>:
$<Workflow_JAR_file_directory>/<Ewt_JAR_file>:
$<Workflow_JAR_file_directory>/<Swing_JAR_file>:"
[-DCHARSET=<character_set>] oracle.apps.fnd.wf.WFFALSnr
<user_name> <password> <connect_string> [<JDBC_driver>]
```

In this command, you can optionally use the `-DCHARSET` option to specify the character set that you want to use. If you do not specify a character set, Oracle Workflow uses UTF8 by default.

Replace the parameters in the command as follows:

- *<character\_set>*—The character set to use for the database session.
- *<user\_name>*—The user name of your Oracle Workflow database account.
- *<password>*—The password for your Oracle Workflow database account.
- *<connect\_string>*—The connect string for the database. The format of the connect string depends on the JDBC driver type.
  - For a JDBC OCI8 driver, the connect string should be the database name as specified in its TNSNAMES entry, in the following format:  
*<database\_name>*
  - For a JDBC THIN driver, you can use two different types of connect string. For the first type, the connect string should include the host name, port number, and database system

identifier (SID) in the following format:

```
<host_name>:<port_number>:<database_SID>
```

For the second type, the connect string should include an Oracle Net name–value pair with the host name, protocol, port number, and SID in the following format:

```
(description=(address=(host=<host_name>) (protocol=
<protocol>) (port=<port_number>)) (connect_data=(sid=
<database_SID>)))
```

- *<JDBC\_driver>*—The JDBC driver type you want to use to connect to the database. The JDBC driver type can be either `oci8` or `thin`. If you do not specify a character set, Oracle Workflow uses the JDBC OCI8 driver by default.

### Starting the Java Function Activity Agent on Windows NT

On Windows NT, you can use the following command to start the Java Function Activity Agent, if you want to use the default JDBC OCI8 driver:

```
jre -classpath
";<JREPATH>\rt.jar;<Workflow_JAR_file_directory>;
<Workflow_JAR_file_directory>\wfjava.jar;<ORACLE_HOME>\wf\
xml\java\lib\xmlparserv2.jar;<Workflow_JAR_file_directory>\
wfapi.jar;<ORACLE_HOME>\jdbc\lib\classes11.zip;
<Workflow_JAR_file_directory>\<Share_JAR_file>;
<Workflow_JAR_file_directory>\<Ewt_JAR_file>;
<Workflow_JAR_file_directory>\<Swing_JAR_file>;"
-nojit [-DCHARSET=<character_set>]
oracle.apps.fnd.wf.WFFALSnr
<user_name>/<password>@<connect_string>
```

In this command, you can optionally use the `-DCHARSET` option to specify the character set that you want to use. If you do not specify a character set, Oracle Workflow uses UTF8 by default.

Replace the parameters in the command as follows:

- *<character\_set>*—The character set to use for the database session.
- *<user\_name>*—The user name of your Oracle Workflow database account.
- *<password>*—The password for your Oracle Workflow database account.
- *<connect\_string>*—The connect string for the database. Since this command uses the default JDBC OCI8 driver, the connect

string should be the database name as specified in its TNSNAMES entry.

You can also use the following command to start the Java Function Activity Agent on Windows NT, if you want to specify the JDBC driver type to use:

```
jre -classpath
";<JREPATH>\rt.jar;<Workflow_JAR_file_directory>;
<Workflow_JAR_file_directory>\wfjava.jar;<ORACLE_HOME>\wf\
xml\java\lib\xmlparserv2.jar;<Workflow_JAR_file_directory>\
wfapi.jar;<ORACLE_HOME>\jdbc\lib\classes111.zip;
<Workflow_JAR_file_directory>\<Share_JAR_file>;
<Workflow_JAR_file_directory>\<Ewt_JAR_file>;
<Workflow_JAR_file_directory>\<Swing_JAR_file>;"
-nojit [-DCHARSET=<character_set>]
oracle.apps.fnd.wf.WFFALsnr <user_name> <password>
<connect_string> [<JDBC_driver>]
```

In this command, you can optionally use the `-DCHARSET` option to specify the character set that you want to use. If you do not specify a character set, Oracle Workflow uses UTF8 by default.

Replace the parameters in the command as follows:

- `<character_set>`—The character set to use for the database session.
- `<user_name>`—The user name of your Oracle Workflow database account.
- `<password>`—The password for your Oracle Workflow database account.
- `<connect_string>`—The connect string for the database. The format of the connect string depends on the JDBC driver type.
  - For a JDBC OCI8 driver, the connect string should be the database name as specified in its TNSNAMES entry, in the following format:  
`<database_name>`
  - For a JDBC THIN driver, you can use two different types of connect string. For the first type, the connect string should include the host name, port number, and database system identifier (SID) in the following format:  
`<host_name>:<port_number>:<database_SID>`  
For the second type, the connect string should include an Oracle Net name–value pair with the host name, protocol,

port number, and SID in the following format:

```
(description=(address=(host=<host_name>)(protocol=
<protocol>)(port=<port_number>))(connect_data=(sid=
<database_SID>)))
```

- *<JDBC\_driver>*—The JDBC driver type you want to use to connect to the database. The JDBC driver type can be either `oci8` or `thin`. If you do not specify a character set, Oracle Workflow uses the JDBC OCI8 driver by default.

## Stopping the Java Function Activity Agent

---

Normally, the Java Function Activity Agent runs as a perpetual job. However, you can stop the agent by running a script called `wfjvstop.sql`, located in the `admin/sql` subdirectory on your Oracle Workflow server. This script places a stop message on the 'Outbound' queue. See: `wfjvstop.sql`: page 16 – 11.

**Note:** If you are running more than one Java Function Activity Agent simultaneously, you must run the `wfjvstop.sql` script once for each Java Function Activity Agent.

## Step 14 Setting Up the Business Event System

The Business Event System is an application service delivered with Oracle Workflow that uses Oracle Advanced Queuing (AQ) to communicate business events between systems. You need to perform this step only if you want to use event processing. See: Overview of the Oracle Workflow Business Event System: page 8 – 235 and Managing Business Events: page 13 – 2.

To set up the Business Event System, perform the following steps:

1. If you want to communicate business events between the local system and external systems, create database links to those external systems.
2. If you want to use custom queues for propagating events, set up your queues.
3. Schedule a listener to monitor the standard WF\_ERROR queue to enable error handling for the Business Event System. See: Setting Up Queues: page 2 – 97 and Scheduling Listeners for Local Inbound Agents: page 13 – 56.

You can either create database links and set up queues manually, or use Oracle DBA Studio in the Oracle Enterprise Manager to perform these steps. Oracle DBA Studio allows workflow administrators to quickly and easily create and administer database links, queue tables, queues, and queue propagation without requiring knowledge of the SQL DDL commands. See: DBA Management Pack, *Oracle Enterprise Manager Administrator's Guide*.

**Context:** You need to perform this step only once.

### Creating Database Links

---

To propagate event messages between systems, you must create database links from your local system to the remote systems. You should fully qualify the database link name with the domain name.

You can use the following syntax to create a database link:

```
CREATE DATABASE LINK <database link name> CONNECT TO
    <user> IDENTIFIED BY <password>
    USING '<connect string>';
```

For example:

```
CREATE DATABASE LINK wf817.us.oracle.com CONNECT TO
    wfuser IDENTIFIED BY welcome
    USING 'wf817';
```

If you have multiple installations of Oracle Workflow on both the local database and the remote database, and you want to use the same username and password to access both systems, you can omit the `<user> IDENTIFIED BY <password>` clause. In this case, the database link uses the username and password of the user who is connected to the database.

```
CREATE DATABASE LINK <database link name> CONNECT TO
    USING '<connect string>';
```

If you want to create a public database link available to all users, specify the parameter PUBLIC.

```
CREATE PUBLIC DATABASE LINK <database link name> CONNECT TO
    <user> IDENTIFIED BY <password>
    USING '<connect string>';
```

To verify the names of your database links, use the following syntax:

```
SELECT db_link FROM all_db_links
```

You can also use the Check Setup web page to verify that your database links are set up. See: [Checking the Business Event System Setup: page 13 – 53](#).

## See Also

CREATE DATABASE LINK, *Oracle SQL Reference*

## Setting Up Queues

---

The Business Event System uses Oracle Advanced Queuing (AQ) to communicate event messages between systems. You must associate a queue with each agent you define in the Event Manager.

When you install Oracle Workflow, four standard queues are created automatically for the four standard Workflow agents. These queues all use the standard WF\_EVENT\_T structure as their payload type. See: [Standard Agents: page 13 – 25](#) and [Event Message Structure: page 8 – 242](#).

The following table lists the standard queues.

Queue Table	Queue Name	Description
WF_IN	WF_IN	Default inbound queue
WF_OUT	WF_OUT	Default outbound queue
WF_DEFERRED	WF_DEFERRED	Default queue for deferred subscription processing
WF_ERROR	WF_ERROR	Default queue for error handling

Table 2 – 2 (Page 1 of 1)

The default retention time set for consumed messages on the standard Workflow queues is seven days. You can change the retention time if necessary using the PL/SQL procedure `DBMS_AQADM.Alter_Queue`. You must not change any other part of the setup of these queues.

However, you must schedule listeners for `WF_DEFERRED` and `WF_ERROR` to enable deferred subscription processing and error handling for the Business Event System, respectively. Also, if you want to use `WF_IN` and `WF_OUT` for event message propagation, schedule a listener for `WF_IN` and propagations for `WF_OUT` as well. See: [Scheduling Listeners for Local Inbound Agents: page 13 – 56](#) and [Scheduling Propagations for Local Outbound Agents: page 13 – 56](#).

You can also set up your own queues for event message propagation. To set up a queue, you must create the queue table, create the queue, and start the queue.

- To create a queue table, use the PL/SQL procedure `DBMS_AQADM.Create_Queue_Table`. Use the following syntax:

```
DBMS_AQADM.Create_Queue_Table (
    queue_table => '<queue table name>',
    queue_payload_type => '<queue payload type>',
    sort_list => 'PRIORITY,ENQ_TIME',
    multiple_consumers => TRUE
    compatible => '8.1');
```

For queues that you want use the standard Workflow format, specify the queue payload type as `WF_EVENT_T`. These queues can use the standard queue handler provided with Oracle Workflow, `WF_EVENT_QH`. If you define a queue with a different payload type, you must create a queue handler to translate between the standard Workflow format and the format required by the queue. See: [Standard APIs for a Queue Handler: page 7 – 23](#).



You can also use the `storage_clause` parameter to specify the tablespace where you want to create the queue table. You may want to specify a tablespace if you expect a queue to be very large.

- To create a queue, use the PL/SQL procedure `DBMS_AQADM.Create_Queue`. Use the following syntax:

```
DBMS_AQADM.Create_Queue (  
    queue_name => '<queue name>',  
    queue_table => '<queue table name>');
```

**Note:** If you want other database users to enqueue messages onto or dequeue messages from your queue, you must grant those users the appropriate privileges using the PL/SQL procedure `DBMS_AQADM.Grant_Queue_Privilege`.

- To start a queue, use the PL/SQL procedure `DBMS_AQADM.Start_Queue`. Use the following syntax:

```
DBMS_AQADM.Start_Queue (  
    queue_name => '<queue name>');
```

Oracle Workflow provides a sample script called `wfevquec.sql` which you can modify to set up your queues, as well as a sample script called `wfevqued.sql` which you can modify to drop queues. These scripts are located on your server in the Oracle Workflow `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.

You can use the Check Setup web page to verify that your queues are set up properly. See: [Checking the Business Event System Setup](#): page 13 – 53.

**Note:** SQL\*Plus version 8.1.6 does not allow you to select the `USER_DATA` column from queue tables. You must have SQL\*Plus version 8.1.7 or higher, which allows you to select `USER_DATA`, if you want to be able to select the event message payload from your Workflow queues.

## See Also

*Administrative Interface, Oracle Application Developer's Guide – Advanced Queuing*

*DBMS\_AQADM, Oracle Supplied PL/SQL Packages Reference*

## Step 15 Setting Up the WF\_EVENT\_OMB\_QH Queue Handler

You need to perform this step only if you are using the Business Event System with Oracle8*i* and you want to use Oracle Message Broker (OMB) to propagate event messages between systems. The WF\_EVENT\_OMB\_QH queue handler translates between the standard Workflow event message structure, WF\_EVENT\_T, and the structure required by OMB queues, OMBAQ\_TEXT\_MSG.

After you set up WF\_EVENT\_OMB\_QH, you can assign this queue handler to Business Event System agents that use propagation protocols you have implemented through OMB. See: Agents: page 13 – 22.

**Note:** You do not need to perform this step if you are using Oracle9*i*. In Oracle9*i*, you can use the Messaging Gateway and Internet access features of Oracle Advanced Queuing to propagate event messages, in place of Oracle Message Broker.

**Context:** You need to perform this step only once.

### ► To Set Up WF\_EVENT\_OMB\_QH

1. Use OMB to create the AQ queues that you want to use for event message propagation. The queues should be single-consumer queues created in your Oracle Workflow schema. You should create at least one inbound and one outbound queue. For example, create queues called WF\_OMB\_IN and WF\_OMB\_OUT.
2. Run the script wfquhndos.pls to create the PL/SQL spec for the WF\_EVENT\_OMB\_QH package. This script is located in the *wf/sql* subdirectory in your Oracle Home.
3. Run the script wfquhndob.pls to create the PL/SQL body for the WF\_EVENT\_OMB\_QH package. This script is located in the *wf/sql* subdirectory in your Oracle Home.

## See Also

Standard APIs for a Queue Handler: page 7 – 23

Event Message Structure: page 8 – 242

Mapping Between WF\_EVENT\_T and OMBAQ\_TEXT\_MSG: page 8 – 257

---

## 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 – 107.

**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.

The following table shows which access levels can access an object under different settings of the 'Preserve Customizations' and 'Lock at this Access Level' options.

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

Table 2 – 3 (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, Windows 2000, or Windows NT PC, Oracle Universal Installer assigns a default access level that is global to the PC and the operating system you are installing on. After installing Oracle Workflow Builder, you can have individual users on the PC change their access level to a new setting which overrides the default access level set for the PC. If a user does not define an access level, 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 require the access level for your users or seed data consumers to be 101 or higher.

You can set a user's access level in Oracle Workflow Builder by having 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.

You can also set the access level directly in the Microsoft Windows registry by using a registry editor such as regedit to edit the decimal value under  
HKEY\_LOCAL\_MACHINE\SOFTWARE\ORACLE\Workflow\Level.

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 – 21.



**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.

**Note:** You can use the Workflow Definitions Loader Release 2.6.1 to upload and download process definitions from all versions of the Oracle Workflow Server embedded in Oracle Applications Release 11*i*, as well as Release 2.6.1, base Release 2.6, and Release 2.5 of the standalone version of the Oracle Workflow Server. However, when you use the Workflow Definitions Loader to upload process definitions to an earlier Oracle Workflow Server version, those processes cannot include any new features introduced in later releases. For more details about which features you must not use with earlier versions, see: Using Oracle Workflow Builder with Different Server Versions: page 3 – 21.

► **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 Home directory structure.
2. Run the program from your operating system prompt as follows (replacing `<username/password@database>` with the username, password and Oracle Net connect string or alias to your database):

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

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

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

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

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

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

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

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

`<input_file>` represents the name and full path of the input file you want to upload from. When using the force option, you should be certain that the process definition in the file is correct as it overwrites the entire process stored in the database. The force option is useful for fixing data integrity problems in a database with a known, reliable file backup. The force option is

also useful for loading .wft files from Oracle Workflow Release 1.0 or 1.0.1, which reflect an older data model.

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

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

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

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

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

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

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

3. In the Parameters window, enter values for the following parameters:

<b>Mode</b>	<p>Specify "Download" to download a process definition from the database to a flat file.</p> <p>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.</p> <p>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.</p> <p>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.</p>
<b>File</b>	<p>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.</p>
<b>Item Type</b>	<p>If you set Mode to "Download", use the List button to choose the item type for the process definition you want to download.</p>

**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>
```

---

## Using the Workflow XML Loader

The Workflow XML Loader lets you upload and download XML definitions for Business Event System objects between a database and a flat file. When you download Business Event System object definitions from a database, Oracle Workflow saves the definitions as an XML file. When you upload object definitions to a database, Oracle Workflow loads the definitions from the source XML file into the Business Event System tables in the database, creating new definitions or updating existing definitions as necessary.

The XML definitions for Business Event System objects are structured according to the following document type definitions (DTDs):

- Events—WF\_EVENTS DTD: page 8 – 302
- Event group members—WF\_EVENT\_GROUPS DTD: page 8 – 305
- Systems—WF\_SYSTEMS DTD: page 8 – 308
- Agents—WF\_AGENTS DTD: page 8 – 311
- Event subscriptions—WF\_EVENT\_SUBSCRIPTIONS DTD: page 8 – 314

You can download Business Event System object definitions in either normal download mode or exact download mode.

- Normal download mode lets you save a generic copy of object definitions from one system that you can use to create similar definitions in other systems. In this mode, the Workflow XML Loader replaces certain system-specific data within the object definitions with tokens. Choose normal download mode, for example, when you want to save Business Event System object definitions from a development system as seed data that can be uploaded to a production system.
- Exact download mode lets you save object definitions exactly as they are specified in the database. In this mode, the Workflow XML Loader does not convert any data to tokens; instead, all values, including system-specific values, are copied to the XML file. Choose exact download mode, for example, when you want to save Business Event System object definitions from one production system so that you can replicate them to another production system that communicates with the first.

In normal download mode, the Workflow XML Loader uses the following tokens to replace system-specific data within Business Event System object definitions. The tokens are prefixed by #.

- **#NEW**—Replaces the global unique identifier for an agent within an agent definition, or for an event subscription within a subscription definition.
- **#LOCAL**—Replaces the global unique identifier for the local system wherever it appears within an agent or subscription definition.
- **#OWNER**—Replaces the name of the schema that owns a queue when the schema appears as part of the queue name and agent address within an agent definition.
- **#SID**—Replaces the database system identifier (SID) when it appears as part of the agent address within an agent definition.
- **#WF\_IN**—Replaces the global unique identifier for the WF\_IN agent on the local system when it appears as the Source Agent, Out Agent, or To Agent within an event subscription definition.
- **#WF\_OUT**—Replaces the global unique identifier for the WF\_OUT agent on the local system when it appears as the Source Agent, Out Agent, or To Agent within an event subscription definition.
- **#WF\_ERROR**—Replaces the global unique identifier for the WF\_ERROR agent on the local system when it appears as the Source Agent, Out Agent, or To Agent within an event subscription definition.

By converting these system-specific values to tokens, the loader produces template definitions that you can use to create similar objects in other systems. When you upload object definitions that contain tokens to a database, Oracle Workflow replaces the tokens with the appropriate values for that system.

## See Also

Managing Business Events: page 13 – 2

### ► **To Download Business Event System XML Definitions from a Database**

To download Business Event System object definitions from a database to a flat XML file, you can either run the Workflow XML Loader manually, or, if you are using the standalone version of Oracle Workflow, you can use a script to run the loader.

To run the Workflow XML Loader manually, run JRE against `oracle.apps.fnd.wf.WFXLoad`. You must specify your CLASSPATH pointing to the Java Runtime Environment, the directory containing the Workflow JAR files, the Oracle JDBC implementation, and the following Workflow JAR files:

- `wfjava.jar`—Workflow Java utilities
- `wfapi.jar`—Workflow Java APIs

**Note:** If you are using the standalone version of Oracle Workflow with Oracle9i, the Workflow JAR files are located in the `<ORACLE_HOME>/jlib` directory. If you are using the version of Oracle Workflow embedded in Oracle Applications, the Workflow JAR files are located in the `<ORACLE_HOME>/wf/java/oracle/apps/fnd/wf/jar/` directory.

For example, on UNIX, use the following command to run the Workflow XML Loader:

```
jre -classpath
"$<JREPATH>/rt.jar:<Workflow_JAR_file_directory>:
<Workflow_JAR_file_directory>/wfjava.jar:
<Workflow_JAR_file_directory>/wfapi.jar:
<ORACLE_HOME>/jdbc/lib/classes111.zip:"
oracle.apps.fnd.wf.WFXLoad -d[e] <user> <password>
<connect_string> <protocol> <lang> <output_file> <object>
<key>
```

On Windows NT, use the following command:

```
jre -classpath
";<JREPATH>\rt.jar;<Workflow_JAR_file_directory>;
<Workflow_JAR_file_directory>\wfjava.jar;
<Workflow_JAR_file_directory>\wfapi.jar;
<ORACLE_HOME>\jdbc\lib\classes111.zip;"
oracle.apps.fnd.wf.WFXLoad -d[e] <user> <password>
<connect_string> <protocol> <lang> <output_file> <object>
<key>
```

If you are using the standalone version of Oracle Workflow, you can use sample scripts called `wfxload` for UNIX or `wfxload.bat` for Windows NT to run the Workflow XML Loader. These scripts are located on your server in the Oracle Workflow *admin* subdirectory. For example, on UNIX, use the following command:

```
wfxload -d[e] <user> <password> <connect_string> <protocol>
<lang> <output_file> <object> <key>
```



On Windows NT, use the following command:

```
wfxload.bat -d[e] <user> <password>  
<connect_string> <protocol> <lang> <output_file> <object>  
<key>
```

When running the Workflow XML Loader, use either the `-d` option or the `-de` option to specify the download mode that you want.

- `-d` —Normal download mode. The loader converts system-specific data within the object definitions to tokens prefixed with #, where appropriate.
- `-de` —Exact download mode. The loader copies the object definitions exactly and does not convert any data to tokens.

Additionally, replace the variables in the download command with your parameters as follows:

- `<user>`—The user name of your database account.
- `<password>`—The password for your database account.
- `<connect_string>`—The connect string for the database. The format of the connect string depends on the JDBC driver type.
  - For a JDBC OCI8 driver, the connect string should be the database name as specified in its TNSNAMES entry, in the following format:  
`<database_name>`
  - For a JDBC THIN driver, you can use two different types of connect string. For the first type, the connect string should include the host name, port number, and database system identifier (SID) in the following format:  
`<host_name>:<port_number>:<database_SID>`  
  
For the second type, the connect string should include an Oracle Net name-value pair with the host name, protocol, port number, and SID in the following format:  
`(description=(address=(host=<host_name>)(protocol=  
<protocol>)(port=<port_number>))(connect_data=(sid=  
<database_SID>)))`
- `<protocol>`—The JDBC driver type you want to use to connect to the database. The JDBC driver type can be either `oci8` or `thin`.
- `<lang>`—The abbreviation for the language of the XML file. This parameter is case insensitive. Use the standard language abbreviations for the Oracle database server, such as `US` for American or `JA` for Japanese. For a list of the standard language

abbreviations, see: Locale Data, *Oracle National Language Support Guide*.

- `<output_file>`—The name and full path of the output file to which you want to save the definitions.
- `<object>`—The type of object definitions you want to download.
  - EVENT—Event, event group member, and event subscription definitions
  - SYSTEMS—System definitions
  - AGENTS—Agent definitions
  - ALL—All Business Event System object definitions, including events, event group members, systems, agents, and event subscriptions

**Note:** The Workflow XML Loader only downloads system, agent, and event subscription definitions that belong to the local system.

- `<key>`—An optional key to restrict the definitions that are downloaded. If you specify a key, the loader retrieves definitions only for those objects whose internal names include that key. The key value is case sensitive and cannot contain any spaces. To retrieve all object definitions of the specified type, you can omit this parameter.

**Note:** If you specify ALL for the object type, the Workflow XML Loader ignores the key and downloads all Business Event System object definitions from the system.



**Attention:** To use the Workflow XML Loader in download mode, you must have a version 8.1.7 or higher database. The download utility is not supported for earlier versions of Oracle8i. To replicate Business Event System objects from one system to another for earlier database versions, you should follow the steps to synchronize systems using predefined subscriptions provided with the Business Event System. See: *Synchronizing Systems*: page 13 – 70.

You can, however, use the Workflow XML Loader in upload mode with versions of Oracle8i earlier than 8.1.7.

### ► To Upload Business Event System XML Definitions to a Database

To upload Business Event System object definitions from an XML file to a database, you can either run the Workflow XML Loader manually, or,

if you are using the standalone version of Oracle Workflow, you can use a script to run the loader.

To run the Workflow XML Loader manually, run JRE against `oracle.apps.fnd.wf.WFXLoad`. You must specify your CLASSPATH pointing to the Java Runtime Environment, the directory containing the Workflow JAR files, the Oracle JDBC implementation, and the following Workflow JAR files:

- `wfjava.jar`—Workflow Java utilities
- `wfapi.jar`—Workflow Java APIs

**Note:** If you are using the standalone version of Oracle Workflow with Oracle9i, the Workflow JAR files are located in the `<ORACLE_HOME>/jlib` directory. If you are using the version of Oracle Workflow embedded in Oracle Applications, the Workflow JAR files are located in the `<ORACLE_HOME>/wf/java/oracle/apps/fnd/wf/jar/` directory.

For example, on UNIX, use the following command to run the Workflow XML Loader:

```
jre -classpath
"$<JREPATH>/rt.jar:$<Workflow_JAR_file_directory>:
<Workflow_JAR_file_directory>/wfjava.jar:
<Workflow_JAR_file_directory>/wfapi.jar:
<ORACLE_HOME>/jdbc/lib/classes111.zip:"
oracle.apps.fnd.wf.WFXLoad -u <user> <password>
<connect_string> <protocol> <lang> <source_file>
```

On Windows NT, use the following command:

```
jre -classpath
";<JREPATH>\rt.jar;<Workflow_JAR_file_directory>;
<Workflow_JAR_file_directory>\wfjava.jar;
<Workflow_JAR_file_directory>\wfapi.jar;
<ORACLE_HOME>\jdbc\lib\classes111.zip;"
oracle.apps.fnd.wf.WFXLoad -u <user> <password>
<connect_string> <protocol> <lang> <source_file>
```

If you are using the standalone version of Oracle Workflow, you can use sample scripts called `wfxload` for UNIX or `wfxload.bat` for Windows NT to run the Workflow XML Loader. These scripts are located on your server in the Oracle Workflow *admin* subdirectory. For example, on UNIX, use the following command:

```
wfxload -u <user> <password> <connect_string> <protocol>
<lang> <source_file>
```

On Windows NT, use the following command:

```
wfxload.bat -u <user> <password> <connect_string> <protocol>  
<lang> <source_file>
```

When running the Workflow XML Loader, use the `-u` option to specify that you want to run the loader in upload mode. Additionally, replace the variables with your parameters as follows:

- `<user>`—The user name of your database account.
- `<password>`—The password for your database account.
- `<connect_string>`—The connect string for the database. The format of the connect string depends on the JDBC driver type.
  - For a JDBC OCI8 driver, the connect string should be the database name as specified in its TNSNAMES entry, in the following format:  
`<database_name>`
  - For a JDBC THIN driver, the connect string should include the host name, port number, and database system identifier (SID) in the following format:  
`<host_name>:<port_number>:<database_SID>`
- `<protocol>`—The JDBC driver type you want to use to connect to the database. The JDBC driver type can be either `oci8` or `thin`.
- `<lang>`—The abbreviation for the language of the XML file. This parameter is case insensitive. Use the standard language abbreviations for the Oracle database server, such as `US` for American or `JA` for Japanese. For a list of the standard language abbreviations, see: *Locale Data, Oracle National Language Support Guide*.
- `<source_file>`—The name and full path of the source file from which you want to upload definitions.

CHAPTER

# 3

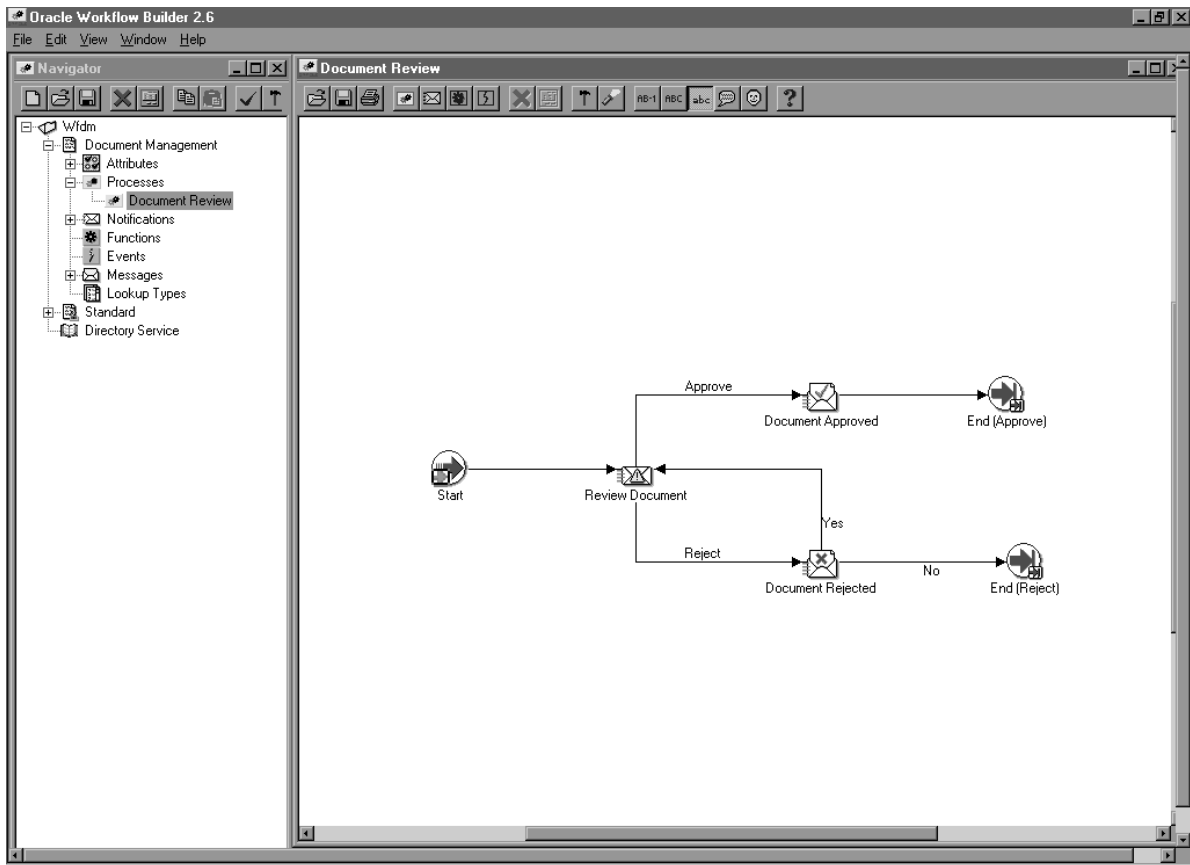
## Defining a Workflow Process

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



**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 – 42.
- **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 – 42.
- **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 – 42.
- **Events**—lists the event activities associated with the current item type. An event activity represents a business event that the process receives, raises, or sends. See: *Activities*: page 4 – 42.
- **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 – 23.
- **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 – 19.

**Note:** Each data store also contains a Directory Service branch. The Directory Service branch lists all the directory service roles that you load from your Oracle Workflow database. See: Roles: page 5 – 24.

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.

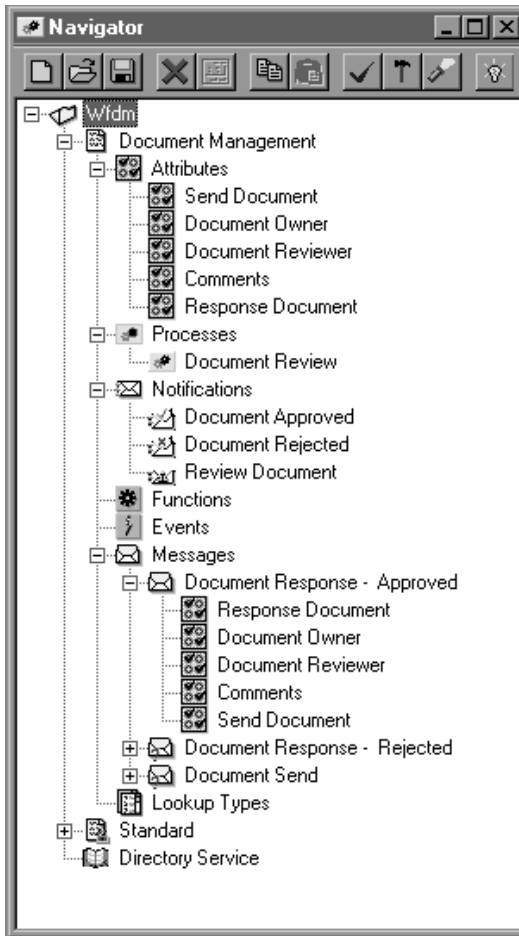
---

## 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.





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

The image shows a 'Search' dialog box with a title bar. It contains a 'Search Text' input field. Below it are two checkboxes: 'Display Name' and 'Internal Name'. A section titled 'Object Type' contains several checkboxes: 'Item Type', 'Function', 'Notification', 'Process', 'Role', 'Message', 'Lookup Type', 'Lookup Code', and 'Attribute'. At the bottom of this section is an 'All Objects' checkbox. At the very bottom of the dialog are two buttons: 'Search' and 'Close'.

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 15 – 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.

## See Also

Modifying Objects in Oracle Workflow Builder: page 4 – 69

### **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 located in the Application Development folder within the Oracle – <SID NAME> program group. If you are

using Windows 95 or NT 4.0 or higher, you can also select the Oracle Workflow Builder icon from the appropriate program folder of the Start menu.

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



**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 – 7.
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 – 20.

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

6. Create new messages. See: To Create a Message: page 4 – 29.

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 – 34.

7. Create a new process activity, notification activity, function activity, or event activity. You may also use predefined standard activities associated with the Standard item type. See: Activities: page 4 – 42 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 – 5.

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 – 3.

## See Also

To Modify a Process Definition: page 3 – 11

Deleting Objects in Oracle Workflow Builder: page 4 – 68

Modifying Objects in Oracle Workflow Builder: page 4 – 69

Item Type Definition Web Page: page 3 – 24

### ► To Create a Process Definition from Top Down

1. To start Oracle Workflow Builder, double-click on the Oracle Workflow Builder icon located in the Application Development folder within the Oracle – <SID NAME> program group. If you are using Windows 95 or NT 4.0 or higher, you can also select the Oracle Workflow Builder icon from the appropriate program folder of the Start menu.
2. Use the Quick Start Wizard to create the framework for your new process definition. Specify the requested information for the new item type and new process activity. See: To Use the Quick Start Wizard: page 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 – 8.  
  
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 – 5.
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 – 68

Modifying Objects in Oracle Workflow Builder: page 4 – 69

Item Type Definition Web Page: page 3 – 24

### ► To Modify a Process Definition

1. To start Oracle Workflow Builder, double-click on the Oracle Workflow Builder icon located in the Application Development folder within the Oracle – <SID NAME> program group. If you are using Windows 95 or NT 4.0 or higher, you can also select the Oracle Workflow Builder icon from the appropriate program folder of the Start menu.
2. Choose Open from the File menu to open a connection to the database or file that contains the process definition you want to

modify. See: To Access Process Definitions in an Existing Data Store: page 3 – 13.

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 – 7, To Define an Item Type or Activity Attribute: page 4 – 8, To Create Lookup Types: page 4 – 20, To Create a Message: page 4 – 29, To Define a Message Attribute: page 4 – 34, or Activities: page 4 – 42.
5. You can also modify the process diagram by displaying the Process window for your process activity. See: Diagramming a Process: page 5 – 5.
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 – 68

Modifying Objects in Oracle Workflow Builder: page 4 – 69

Item Type Definition Web Page: page 3 – 24

---

## 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

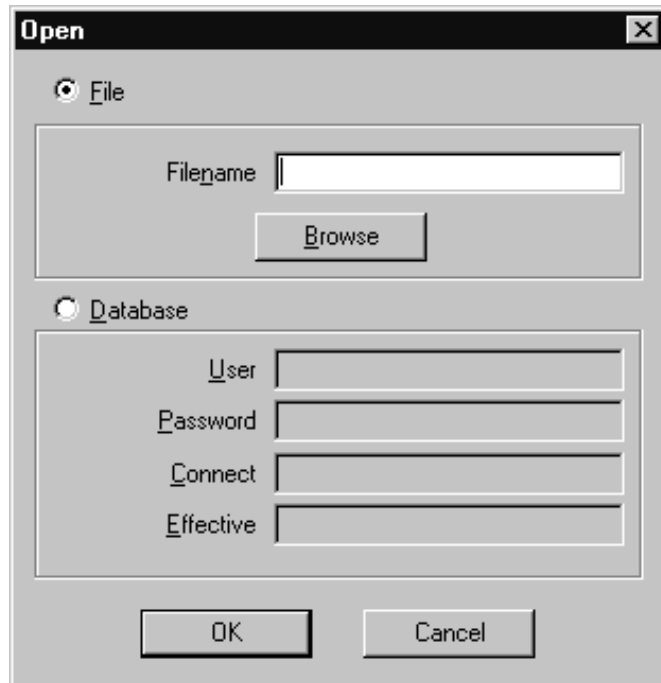


database as your source controlled version, as others with access to the database can update the definition.

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

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

1. To start Oracle Workflow Builder, double-click on the Oracle Workflow Builder icon located in the Application Development folder within the Oracle – <SID NAME> program group. If you are using Windows 95 or NT 4.0 or higher, you can also select the Oracle Workflow Builder icon from the appropriate program folder from the Start menu. In Oracle Workflow Builder, select Open... from the File menu.



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).

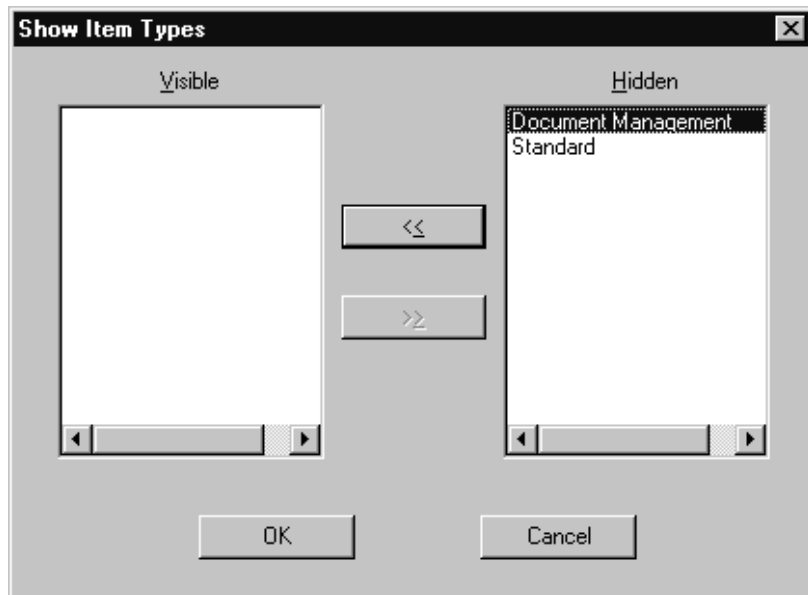
**Note:** You can also drag and drop a .wft file from the Microsoft Windows 95/98/2000/NT 4.0 Explorer or Microsoft Windows NT File Manager into the navigator tree to open that file in Oracle Workflow Builder.

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

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

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

5. If you wish to retrieve a process definition that was effective at a particular point in time, you can specify a date and time in the Effective field and have Oracle Workflow Builder retrieve that data from the database. The format that you should use to specify the date and time depends on the date and time preferences defined in the Regional Settings of your Windows Control Panel.



6. If multiple item types exist in the data store, the Show Item Types window appears. Select from the Hidden list, the item type(s) you want to view, and choose << to move it into the Visible list. Choose OK to load these item types into the navigator tree.
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

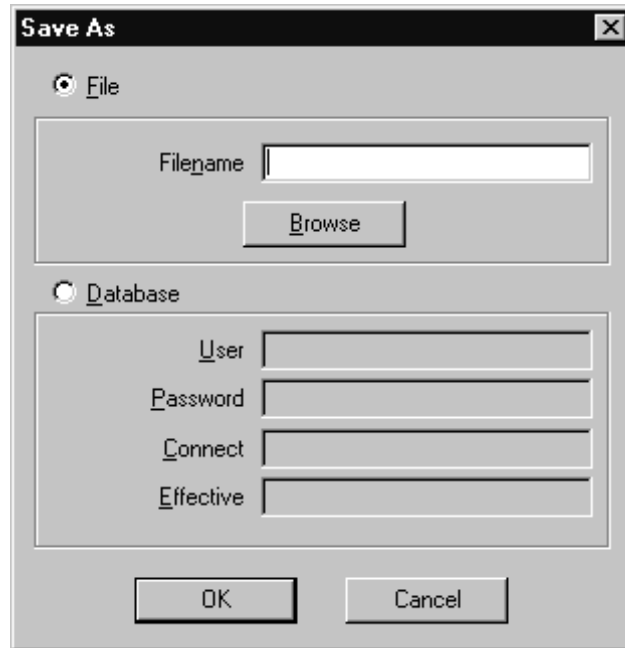
1. Choose Save from the File menu to save your work and make the changes immediately effective.

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 – 18 and Using the Workflow Definitions Loader: page 2 – 107.



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 – 11.

**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:

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

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

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

**Note:** If you run Oracle Workflow Builder in Microsoft Windows 95 or Windows NT 4.0 or higher, you can also double-click on a workflow data file (.wft) from the Windows Explorer to automatically open that file and start Oracle Workflow Builder.

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

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

```
-E <item_type>
```

For example:

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

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

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

For example:

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

## See Also

Using the Workflow Definitions Loader: page 2 – 107

Creating a Shortcut to a Workflow Process: page 5 – 22

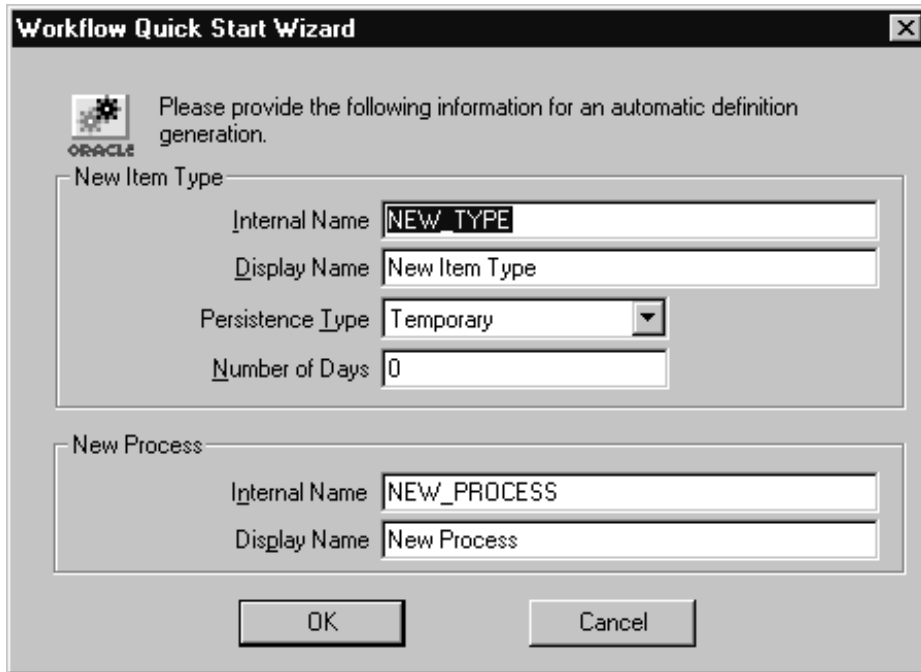
---


## 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 called wfchitt.sql. See: Wfchitt.sql: page 16 – 8.
- 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 called wfchact.sql. See: Wfchact.sql: page 16 – 7.

**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.



---

## Using Oracle Workflow Builder with Different Server Versions

The Oracle Workflow Builder Release 2.6.2 is compatible with all versions of the Oracle Workflow Server embedded in Oracle Applications Release 11*i*, as well as with Release 2.6.2, Release 2.6.1, base Release 2.6, and Release 2.5 of the standalone version of the Oracle Workflow Server.

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



**Attention:** The Oracle Workflow Builder is the only Release 2.6.2 client component that is compatible with earlier versions of the Oracle Workflow Server. The Oracle Workflow Mailer Release 2.6.2 is not compatible with any earlier versions of the server; it is only certified with the Oracle Workflow Server Release 2.6.2.

---

### Using the Release 2.6.2 Oracle Workflow Builder with a Release 2.6 Embedded Server or a Release 2.6.2 or Release 2.6.1 Standalone Server

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

You can also open existing process definitions that were created with the Release 2.5 Oracle Workflow Builder or with the base Release 2.6 Oracle Workflow Builder and save these process definitions to a database with Oracle Workflow Server Release 2.6 embedded in Oracle Applications, or with standalone Oracle Workflow Server Release 2.6.2 or Release 2.6.1.

## Using the Release 2.6.2 Oracle Workflow Builder with a Release 2.6 Standalone Server

---

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

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

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

You can also create new workflow processes using only base Release 2.6 components. However, the version of the Standard item type used by the Oracle Workflow Builder Release 2.6.2 contains some later Release 2.6 components. If you want to save a new process definition to a database with base standalone Oracle Workflow Server Release 2.6, perform the following steps:

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

## Using the Release 2.6.2 Oracle Workflow Builder with a Release 2.5 Standalone or Embedded Server

---

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

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

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

You can also create new workflow processes using only Release 2.5 components. However, the version of the Standard item type used by the Oracle Workflow Builder Release 2.6.2 contains some Release 2.6 components. If you want to save a new process definition to a database with Oracle Workflow Server Release 2.5, perform the following steps:

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

---

## Item Type Definition Web Page

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

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

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

### ► To Query an Item Type

1. Enter the following URL in your web browser:

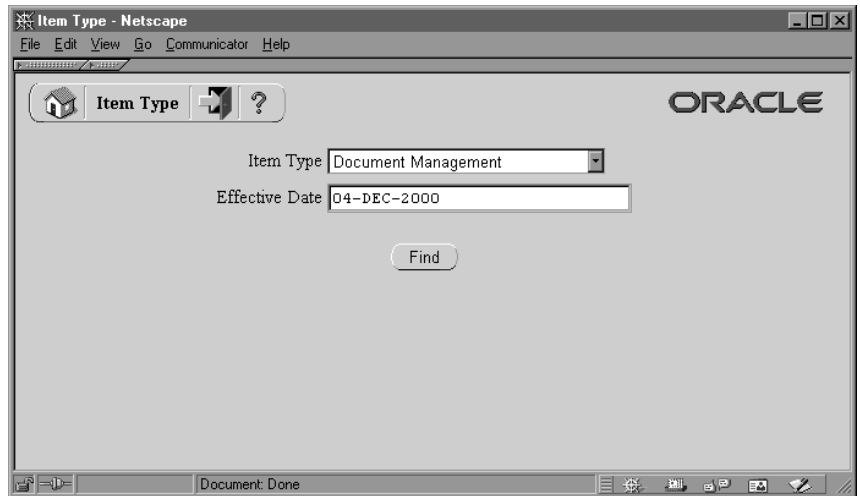
```
<webagent>/wf_item_definition.find_item_type
```

Replace *<webagent>* with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.

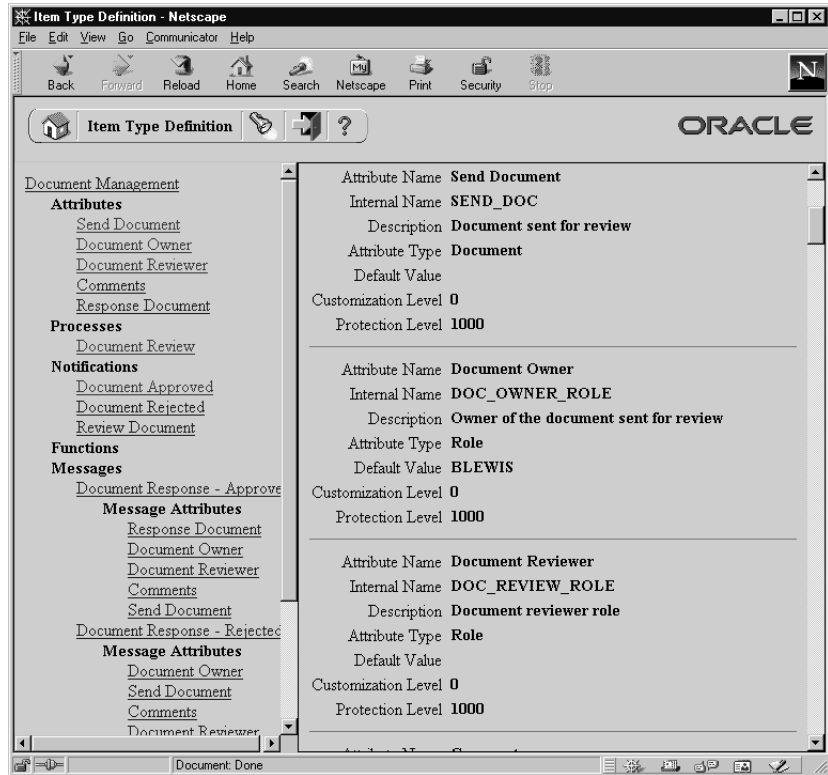


**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears.

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



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 – 6.
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

**T**his 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 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 – 7](#).

---

#### 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 4 – 34.

## See Also

Item Attributes: page C – 3

### Attribute Types

---

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

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

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

- URL—The attribute value is a Universal Resource Locator (URL) to a network location. If you reference a URL attribute in a notification message as a message attribute, the notification, when viewed from the Notification Details web page or as an HTML-formatted e-mail, displays 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 as a character string, generated from a PL/SQL procedure.
- PL/SQL CLOB document—a document representing data from the database as a character large object (CLOB), generated from a PL/SQL procedure.

See: To Define a Document Attribute: page 4 – 14.

- Role—The attribute value is the internal name of a role. If a message attribute of type role is included in a notification message, the attribute automatically resolves to the role’s display name, eliminating the need for you to maintain separate attributes for the role’s internal and display names. Also when you view the notification from a web browser, the role display name is a hypertext link to the e-mail address for that role. To set a default value for the attribute, you must initially load roles from the database. See: Roles: page 5 – 24.
- Attribute—The attribute value is the internal name of another existing item type attribute that you want to maintain references to in a process.
- Event—The attribute value is a Business Event System event message in the standard WF\_EVENT\_T structure. See: Event Message Structure: page 8 – 242.

**Note:** If you store an event message in an item attribute of type event, you can access the event data within that event message by creating an item attribute of type URL and setting the value of the URL attribute to reference the event data. See: SetItemAttribute: page 8 – 48.

## Persistence Type

---

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

If you set an item type’s Persistence to Temporary, you must also specify the number of days of persistence (*n*). The status audit trail for each instance of a Temporary item type is maintained for at least *n* days of persistence after its completion date. After the *n* days of persistence, you can then use any of the WF\_PURGE APIs to purge the item type’s runtime status information. See: WF\_PURGE: page 8 – 111.

If you set an item type's Persistence to Synchronous, Oracle Workflow expects instances of that item type to be run as forced synchronous processes with an item key of #SYNCH. Forced synchronous processes complete in a single SQL session from start to finish and never insert into or update any database tables. Since no runtime status information is maintained, you do not normally need to perform any purging for a process with the Synchronous persistence type. However, if you run the process with a unique item key in asynchronous mode for testing or debugging purposes, Oracle Workflow does maintain runtime status information for that process instance. You can purge this information by changing the item type's Persistence to Temporary and running any of the WF\_PURGE APIs. Then change the item type's Persistence back to Synchronous. See: Synchronous, Asynchronous, and Forced Synchronous Processes: page 8 – 14, WF\_PURGE: page 8 – 111, and Purging for Performance: page C – 8.

**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 – 119.

### **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 – 13.

## **External Document Integration**

---

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

In a workflow process, you can attach documents generated by a PL/SQL procedure, which we call PL/SQL or PL/SQL CLOB 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 – 34.

For PL/SQL documents and PL/SQL CLOB 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 APIs for "PL/SQL" and "PL/SQL CLOB" Documents: page 7 – 17.

► To Create an Item Type

The screenshot shows a dialog box titled "Navigator Control Properties" with a close button (X) in the top right corner. It has three tabs: "Item Type", "Roles", and "Access". The "Item Type" tab is selected and contains the following fields:

- Internal Name**: A text input field.
- Display Name**: A text input field.
- Description**: A text area with vertical scroll bars.
- Persistence**: A dropdown menu currently set to "Temporary".
- Number of Days**: A text input field containing the value "0".
- Selector**: A text input field.

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

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 called wfchitt.sql. You should only use this script to correct errors in an item type's internal name during design time. Do not use this script to rename item types that are involved in running instances of processes. See: Wfchitt.sql: page 16 – 8.

**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 – 13.
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 – 17.
9. Choose Apply to save your changes, OK to save your changes and close the property page or Cancel to cancel your changes and close the property page.
10. A secondary branch appears in the navigator tree that represents the item type you just created. You can review or edit the properties of this item type at any time by double-clicking on the item type in the navigator tree or by selecting the item type and choosing Properties from the Edit menu.
11. Define as many item type attributes as necessary to use as global variables in your process. You use these item type attributes to pass values to and from your function, notification, and event activities. See: To Define an Item Type or Activity Attribute: page 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.

**Navigator Control Properties**

Attribute Access

Item Type: Document Management

Internal Name:

Display Name:

Description:

Type: Text

Length:

Default

Type: Constant

Value:

OK Cancel Apply Help

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

**Navigator Control Properties**

Attribute

Function: Once Only

Internal Name:

Display Name:

Description:

Type: Text

Length:

Default

Type: Constant

Value:

OK Cancel Apply Help

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:** To update the internal name for an attribute once it is defined, you must use special SQL scripts called wfchita.sql and wfchacta.sql. You should only use these scripts to correct errors in an attribute's internal name during design time. Do not use these scripts to rename attributes that are involved in running instances of processes. See: Wfchita.sql: page 16 – 8 and Wfchacta.sql: page 16 – 7.

**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 – 12.

**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 – 13.



- **Document**—Enter an optional string that identifies the document in the default value field. See: To Define a Document Attribute: page 4 – 14.

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

- **Role**—Specify a role name. See: Roles: page 5 – 24.
- **Attribute**—Specify the name of an item type attribute that you want to maintain references to in a process by choosing from the list of existing item type attributes.
- **Event**—If you are defining an item type attribute, you cannot specify any default value for an event attribute. If you are defining an activity attribute, you can only specify an event item type attribute as the default value.

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

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

**Note:** An activity attribute type of 'Text' is compatible with any item attribute type, 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 – 17.
10. Choose Apply to save your changes.

11. Any item type attribute you create appears beneath the Attributes branch in the navigator tree. Any function activity attribute you define appears beneath the activity you defined it for in the navigator tree. You can review or edit the properties of an attribute at any time by double-clicking on the attribute in the navigator tree or by selecting the attribute and choosing Properties from the Edit menu.



**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. Additionally, if you are defining a message attribute of type URL, you can include argument strings that are token substituted with other message attributes. The message attributes used for token substitution can have constant values or can reference the values returned from item type attributes. See: To Define a Message Attribute: page 4 – 34 and To Token Substitute an Attribute: page 4 – 41.

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

```
-&message_attr-
```

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

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

**Note:** If you are defining a message attribute of type URL, you can also include a special token in your argument string called

- &#NID– which Oracle Workflow substitutes with the notification ID of the runtime notification.
3. If your URL attribute contains an argument string, you must adhere to the following restrictions:
    - You cannot token substitute that argument string with another item attribute of type Document.
    - You can token substitute that argument string with another Form attribute or URL attribute. However, the argument string for the other attribute is not further token substituted.
  4. If you need to pass a date and time as an argument to a URL, you should use TO\_CHAR to format the string as YYYY/MM/DD+HH24:MI:SS. Similarly, you need to do the correlating format translation in the function that the URL calls, using TO\_DATE. This formatting is required because in multibyte databases, the month portion of the DD–MON–YYYY format could potentially translate to a value that is not acceptable across a URL.
  5. 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 – 41.

**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.

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

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

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

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

**Note:** The PL/SQL procedure must follow a standard API format. See: Standard APIs for "PL/SQL" and "PL/SQL CLOB" Documents: page 7 – 17.

For example, the following string represents the PL/SQL document, *po\_req:2034*, generated by the procedure *po\_wf.show\_req*.

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

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

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

```
plsqcllob:<procedure>/<document_identifier>.
```

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

**Note:** The PL/SQL procedure must follow a standard API format. See: Standard APIs for "PL/SQL" and "PL/SQL CLOB" Documents: page 7 – 17.

For example, the following string represents the PL/SQL CLOB document, po\_req:2036, generated by the procedure po\_wf.show\_req\_clob.

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

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

- Do not use tokens within the CLOB.
  - Ensure that the PL/SQL procedure performs any formatting you require.
4. If you wish to generate the document identifier for a PL/SQL or PL/SQL CLOB document dynamically, you can token substitute the document identifier with other item type attributes. The item attribute names must be in uppercase and must be separated by a colon. See: To Token Substitute an Attribute: page 4 – 41.

For example:

```
plsqcllob: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. 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.

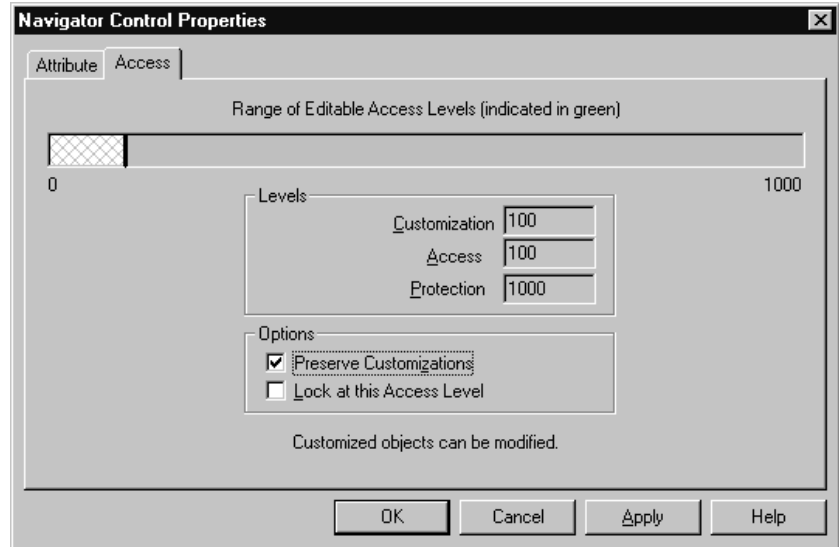
When you are done, choose OK.

**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 – 101.

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 – 18 and Using the Workflow Definitions Loader: page 2 – 107.

► **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
<b>NONE</b> —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).
<b>Preserve Customizations</b> —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
<b>Lock at this Access Level</b> —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.
<b>BOTH</b> —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.

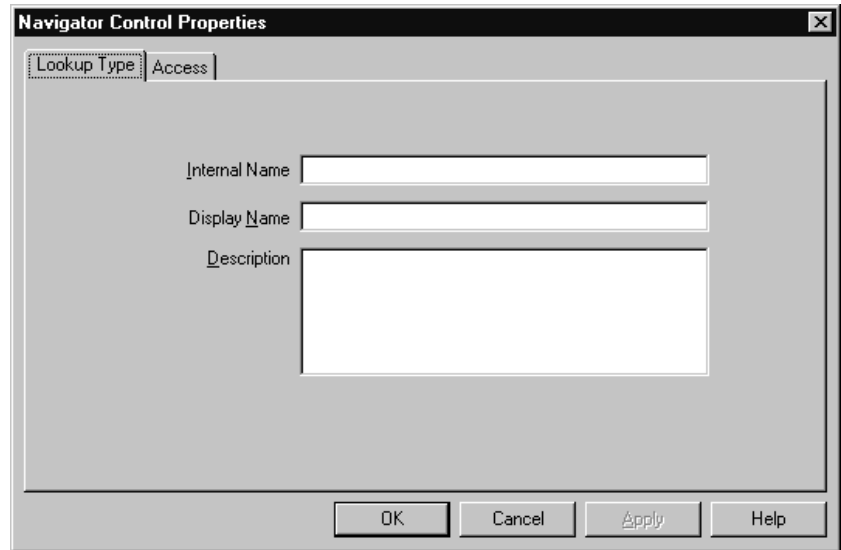
---

## 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, lookup types are not item type specific; when you create an activity or an attribute, you can reference any lookup type in your current data store, regardless of the item type that the lookup type is associated with. See: To Create Lookup Types: page 4 – 20.

► 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:** To update the internal name for a lookup type once it is defined, you must use a special SQL script called wfchlut.sql. You should only use this script to correct errors in a lookup type's internal name during design time. Do not use this script to rename lookup types that are involved in running instances of processes. See: Wfchlut.sql: page 16 – 9.

**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 – 17.
4. Choose Apply to save your changes, OK to save your changes and close the property page or Cancel to cancel your changes and close the property page.

5. The lookup type you just defined now appears beneath the Lookup Types branch in the navigator tree. You can review or edit the properties of this lookup type at any time by double-clicking on the lookup type in the navigator tree or by selecting the lookup type and choosing Properties from the Edit menu.
6. Now define the lookup codes for your lookup type. See: To Create Lookup Codes for a Lookup Type: page 4 – 21.

► **To Create Lookup Codes for a Lookup Type**

1. Select a lookup type from the navigator tree and choose New Lookup Code from the Edit menu. A Lookup Code property page appears.
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:** To update the internal name for a lookup code once it is defined, you must use a special SQL script called wfchluc.sql. You should only use this script to correct errors in a lookup code's internal name during design time. Do not use this script to rename lookup codes that are involved in running instances of processes. See: Wfchluc.sql: page 16 – 8.

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

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

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

#### ► To Copy a Lookup Code

1. Select the lookup code to copy in the navigator tree.

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

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

---

## Messages

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

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

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

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

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 – 24.

## Send and Respond Message Attributes

---

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

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

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

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

You can, however, embed Document message attributes within the body of a message for token substitution.

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

Message Attributes: page C – 5

### **#HIDE\_REASSIGN Attribute**

---

You can use a special message attribute or notification attribute with the internal name #HIDE\_REASSIGN to hide the Reassign button in the Notification Detail web page. When users view a notification from their Worklist web page, the response frame in the Notification Detail page includes the Reassign button by default. If you want to prevent users from reassigning a notification, you can add the #HIDE\_REASSIGN attribute to control whether the Reassign button is displayed or hidden.

The #HIDE\_REASSIGN attribute must be of type text. To hide the Reassign button, set the value of this attribute to Y. To display the Reassign button, set the value to N.

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

## **See Also**

To Reassign a Notification to Another User: page 10 – 22

### **#FROM\_ROLE Attribute**

---

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

reassigned, Oracle Workflow automatically sets the From Role to the original recipient when sending the notification to the new recipient.

The From Role for each notification is displayed in the Worklist web page to give users additional information for reviewing and responding to the notifications. Additionally, the Find Notifications page lets you search for notifications based on the From Role.

To specify the From Role for a message, create a message attribute with the following properties:

- Internal Name—#FROM\_ROLE
- Display Name—From Role
- Description—From Role
- Type—Role
- Source—Send
- Default Type—Item Attribute
- Default Value—The item attribute that holds the value you want to set as the From Role for the message

## See Also

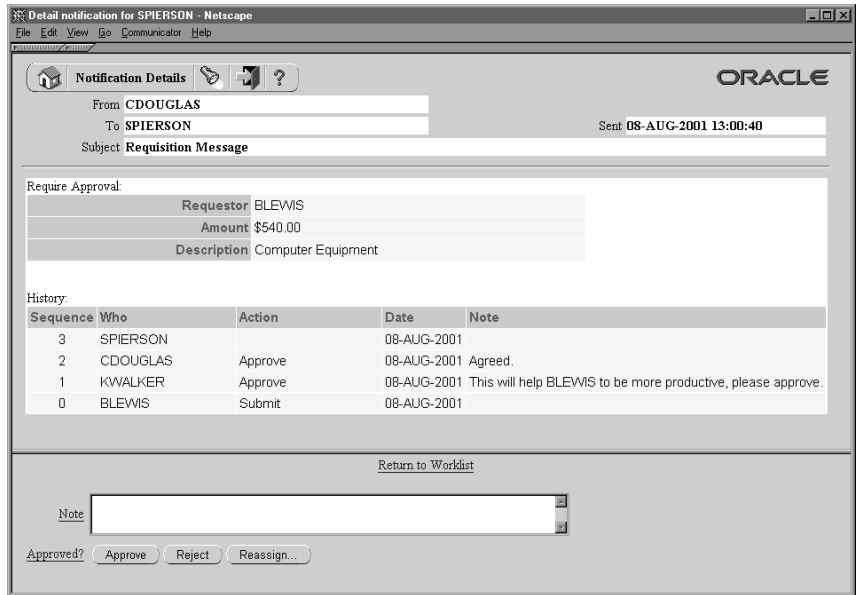
To View Notifications from the Worklist: page 10 – 17

To Find Notifications: page 10 – 15

### **WF\_NOTIFICATION() Message Function**

In addition to message attribute tokens, you can also use a special message function called WF\_NOTIFICATION() to add context-sensitive content to a message body. Depending on the parameters you provide, the WF\_NOTIFICATION() function can produce either a table of message attributes or a notification history table. The tables are created in a standard Oracle Workflow format.





### Message Attribute Table

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

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

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

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

### Notification History Table

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

```
WF_NOTIFICATION(HISTORY)
```

The notification history table contains a row for each previous execution of the notification activity in the process, as well as a row for the initial submission of the process. For example, for a requisition approval notification that is sent to several approvers in turn, the

notification history table would contain a row for each approver to whom the notification was sent, as well as a row for the process owner.

The notification history table includes the following columns:

- **Sequence**—The order in which the executions of the notification activity took place, beginning at zero (0) for the initial submission of the process by the process owner.
- **Who**—The notification recipient or process owner.
- **Action**—The action with which the recipient responded to the notification.
- **Date**—The notification date.
- **Note**—An additional note from the recipient. To allow a recipient to add a note for the notification history table, you must create a special 'Respond' message attribute with the internal name WF\_NOTE.

Define the message attribute with the following properties:

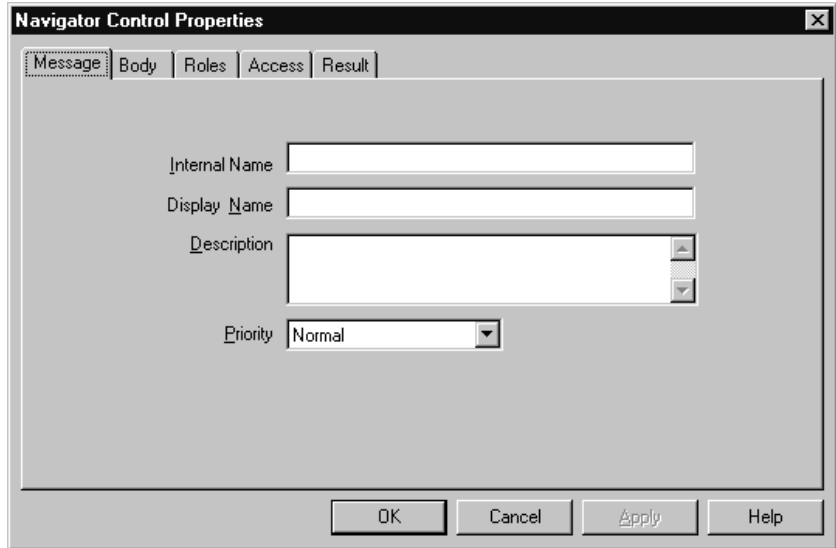
- **Internal Name**—WF\_NOTE
- **Display Name**—Note
- **Description**—Note
- **Type**—Text
- **Source**—Respond

When the WF\_NOTE attribute is defined with a source of 'Respond', it appears as part of the notification response section, and the recipient can enter a note there when responding to the notification. The WF\_NOTIFICATION() function retrieves the note text stored in the WF\_NOTE attribute and displays it in the notification history table.

If the recipient did not enter a note, or if the WF\_NOTE message attribute was not defined for the notification, then the Note column in the notification history table is left blank.

**Note:** The process owner cannot add a note for the notification history table when submitting the process. Only a notification recipient can add a note when responding to the notification.

► 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 called wfchmsg.sql. You should only use this script to correct errors in a message's internal name during design time. Do not use this script to rename messages that are involved in running instances of processes. See: Wfchmsg.sql: page 16 – 9.

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

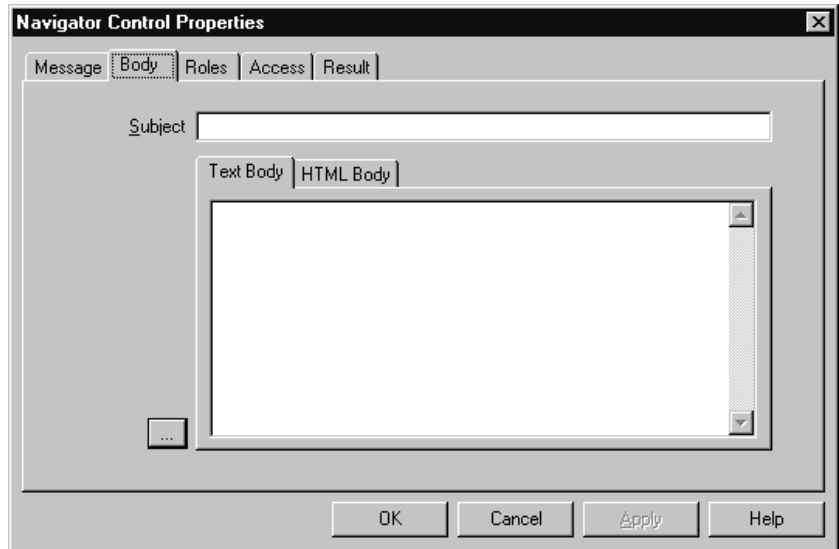
3. Choose High, Normal, or Low for the default priority of the message. The priority level simply informs the recipient of the urgency of the message. It does not affect the processing or delivery of the message.

**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 – 6.

**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 ampersand (&) followed by the message attribute's internal name. See: *Send and Respond Message Attributes*: page 4 – 24 and *To Define a Message Attribute*: page 4 – 34.



**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 an HTML-formatted notification message from the Notification Details web page, or from an e-mail reader that displays HTML-formatted messages or HTML-formatted message attachments.

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



**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 attributes with runtime values when it delivers the notification. To embed a message attribute, enter an "&" followed by the message attribute's internal name.

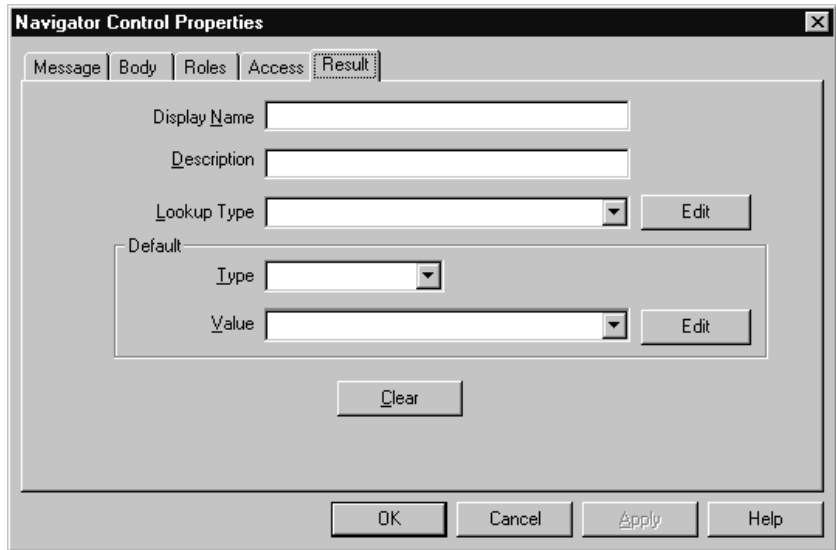


**Attention:** The text in a message body must be less than 4000 bytes. If you include message attributes in the text for token substitution, then the final message body can increase up to 32000 bytes.

**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.

Additionally, you can use the message function WF\_NOTIFICATION() to include a formatted table of message attributes or a notification history table in the text or HTML message body.

10. Choose Apply to save your changes.
11. Select the Roles tab page to specify the roles that have access to this message. (This functionality will be supported in a future release.)
12. Select the Access tab page to set the access levels allowed to modify this message. See: Allowing Access to an Object: page 4 – 17.
13. If you want the notification message to prompt the performer for a response value and you want Oracle Workflow to interpret that response value as the result of the notification activity, select the Result tab page and complete the information requested. Oracle Workflow uses the information you specify in the Result tab page to create a special 'Respond' message attribute called RESULT. See: Message Result: page 4 – 24



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

**Note:** To create any other type of message attribute, see: To Define a Message Attribute: page 4 – 34.

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 – 34.

### ► To Define a Message Attribute

The screenshot shows the 'Navigator Control Properties' dialog box with the 'Attribute' tab selected. The 'Message' field is set to 'Document Response - Approved'. Other fields include 'Internal Name', 'Display Name', 'Description', 'Type' (set to 'Text'), 'Source' (set to 'Send'), and 'Length'. A 'Default' section contains 'Type' (set to 'Constant') and 'Value'.

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 called wfchmsga.sql. You should only use this script to correct errors in a message attribute's internal name during design time. Do not use this script to rename message attributes that



are involved in running instances of processes. See: Wfchmsga.sql: page 16 – 9.

**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 – 12.



**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. Your 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 – 13.





**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 – 19.

- **Document**—Enter a string that identifies the document in the Default Value field. See: To Define a Document Attribute: page 4 – 14.
- **Role**—Specify a role name. If a message attribute of type role is included in a notification message, the attribute automatically resolves to the role's display name, eliminating the need for you to maintain separate attributes for the role's internal and display names. Also when you view the notification from a web browser, the role display name is a hypertext link to the e-mail address for that role. To set a default value for the attribute, you must initially load roles from the database. See: Roles: page 5 – 24.
- **Event**—Specify an event item type attribute as the default value.

 **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, specify a Frame Target. When you reference this message attribute in a message, the URL opens according to what you specify as the frame target. The frame target can be:
  - New Window—the URL loads in a new, unnamed browser window.
  - Same Frame—the URL loads in the same frame as the element that references the URL attribute.
  - Parent Frameset—the URL loads into the immediate FRAMESET parent of the current frame. This value is equivalent to Same Frame if the current frame has no parent.
  - Full Window—the URL loads into the full, original window, thus cancelling all other frames. This value is equivalent to Same Frame if the current frame has no parent.
9. If your message attribute is a Send attribute and is of type 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) Document attributes in the notification message and are not limited to one or the other.

10. If your message attribute is a Send attribute and is of type URL, you can check Attach Content to append an attachment called Notification References to the notification message. This attachment includes a link to each URL attribute for the message

that has Attach Content checked. You can navigate to a URL by choosing its link.

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

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

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

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

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



**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 – 39

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

### Example 'Respond' Message Attributes

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

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

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: " "

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:

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

### ► 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 – 6.

### ► To Copy a Message

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

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

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

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

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

---

## Activities

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

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

Oracle Workflow provides an item type called Standard that includes generic activities you can use in any process you define. For example, some of the activities perform standard functions such as comparing two values. See: Standard Activities: page 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 – 26.

## **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 – 48.

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 – 8 and Roles: page 5 – 24.

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. When the Notification System completes execution of the

post-notification function in RESPOND mode, the Workflow Engine then runs the post-notification function again in RUN mode.

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 – 13 and Notification Model: page 8 – 192.

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 – 3.

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 – 61.

## **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 – 17.

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

As an external Java program, a function activity is able to enqueue payload information into an Oracle Advanced Queuing outbound queue for the Java Function Activity Agent to dequeue and consume. The results of the Java program are enqueued into an inbound queue that the Workflow Engine consumes and processes. This functionality is

currently only available for the standalone version of Oracle Workflow. See: To Create a Function Activity: page 4 – 50.

## **Event Activity**

---

An event activity represents a business event from the Business Event System within a workflow process. Include event activities in workflow processes to model complex processing or routing logic for business events beyond the standard event subscription options of running a function or sending the event to a predefined agent. See: Managing Business Events: page 13 – 2.

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

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

When the activity receives an event, the Workflow Engine stores the event name, event key, and event message in item type attributes, as specified in the node's event details; sets any parameters in the event message parameter list as item type attributes for the process, creating new item type attributes if a corresponding attribute does not already exist for any parameter; and then continues the thread of execution from the event activity. If that activity has already received an event, then the On Revisit flag for the activity determines whether the Workflow Engine reexecutes the activity. See: To Define Optional Activity Details: page 4 – 59.

If the event was originally raised by a Raise event activity in another workflow process, the item type and item key for that process are included in the parameter list within the event message. In this case, the Workflow Engine automatically sets the specified process as the parent for the process that receives the event, overriding any existing parent setting. See: SetItemParent: page 8 – 79.

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

Additionally, the activity retrieves the names and values of any activity attributes defined for it and sets these attributes as parameters in the parameter list for the event message. If the event message is later received by another process, the Workflow Engine sets the event parameters as item type attributes for that process. See: Event Message Structure: page 8 – 242.

The activity also automatically sets the item type and item key for the current workflow process in the parameter list for the event message. If the event message is later received by another process, the Workflow Engine uses that item type and item key to automatically set the process that raised the event as the parent for the process that receives the event. See: SetItemParent: page 8 – 79.

- A Send event activity retrieves the event name, event key, event message, outbound agent, and inbound agent, as specified in the node's event details. If no correlation ID is initially specified in the event message, the correlation ID is automatically set to the item key of the process. Then the Send event activity sends the event directly from the outbound agent to the inbound agent. The event details can be dynamically determined at runtime using item type attributes. You can also specify the event name, outbound agent, and inbound agent as predefined constants for the event activity node. See: To Create an Event Activity: page 4 – 54 and To Define Event Details for an Event Node: page 5 – 12.

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

## Process Activity

---

A process activity represents a collection of activities in a specific relationship. When a process activity is contained in another process it is called a subprocess. In other words, activities in a process can also be processes themselves. There is no restriction on the depth of this hierarchy. See: To Create a Process Activity: page 4 – 57.

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

## See Also

Subprocesses: page C – 5

## Activity Cost

---

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

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



**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 – 43, *To Set Engine Thresholds*: page 2 – 47, and *Deferring Activities*: page C – 7.

► To Create a Notification Activity

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

- Internal Name: Text input field.
- Display Name: Text input field.
- Description: Text input field.
- Icon: A dropdown menu showing 'NOTIFY.ICO' and a 'Browse' button with an envelope icon.
- Function Name: Text input field.
- Function Type: A dropdown menu showing 'PL/SQL'.
- Result Type: A dropdown menu showing '<None>' and an 'Edit' button.
- Message: A dropdown menu and an 'Edit' button.
- Expand Roles: A checkbox.

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

1. Select the item type that you want to create a 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 called wfchact.sql. You should only use this script to correct errors in an activity's internal name during design time. Do not use this script to rename activities that are involved in running instances of processes. See: Wfchact.sql: page 16 – 7.

**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 – 20.

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

4. Select the name of the message you want this notification to send. See: To Create a Message: page 4 – 29.
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 – 43.
6. You can optionally specify a PL/SQL stored procedure in the Function field. The procedure is known as a post-notification function and lets you couple processing logic to the notification activity. 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. When the Notification System completes execution of the post-notification function in RESPOND mode, the Workflow Engine then runs the post-notification function again in RUN mode. See: Standard API for PL/SQL Procedures Called by Function Activities: page 7 – 3 and Post-Notification Functions: page 8 – 13.

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 – 61.

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 – 85.

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 – 59.
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 – 17.
12. Choose OK to save your changes and close the property pages.
13. The notification activity now appears beneath Notifications in the navigator tree. You can review or edit the properties of this activity at any time by double-clicking on the activity in the navigator tree or by selecting the activity and choosing Properties from the Edit menu or by pressing Enter on your keyboard.

## See Also

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

### ► To Create a Function Activity

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

- Internal Name:
- Display Name:
- Description:
- Icon:  [Gear icon] [Browse]
- Function Name:
- Function Type:
- Result Type:  [Edit]
- Cost:

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

1. Select the item type that you want to create a function for in the navigator tree, then choose New Function from the Edit menu.



Define your function activity in the Activity property page that appears.

2. A function activity must have an Internal Name (all uppercase and no 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 called `wfchact.sql`. You should only use this script to correct errors in an activity's internal name during design time. Do not use this script to rename activities that are involved in running instances of processes. See: `Wfchact.sql`: page 16 – 7.

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

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

For a PL/SQL function, set the function type to PL/SQL and specify the function as `<package_name>.<procedure_name>`. The function must be defined according to a standard API. See: Standard API for PL/SQL Procedures Called by Function Activities: page 7 – 3.

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

```
<schema_name><item_type>
```

See: Workflow Queue APIs: page 8 – 162.

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 send 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 – 9.

For an external Java function activity, set the function type to External Java and enter the class name of your custom Java class as the function name. This functionality is currently only available for the standalone version of Oracle Workflow. If the custom class is

within a package, prefix the class name with the package name in the following format:

```
<customPackage>.<customClass>
```

The Java class must be defined according to a standard API. See: Standard API for Java Procedures Called by Function Activities: page 7 – 8.

The Workflow Engine enqueues an entry in the 'Outbound' queue. The Java Function Activity Agent dequeues messages of this type, executes your Java program, and enqueues the results onto the 'Inbound' queue. The background engine then takes care of messages on the inbound queue and restarts your original workflow process. See: Setting Up the Java Function Activity Agent: page 2 – 86 and Deferred Processing: page 8 – 9.

**Note:** These 'Outbound' and 'Inbound' queues are separate from the queues used for the Business Event System. See: Workflow Queue APIs: page 8 – 162.

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

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

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

5. Specify the cost of this function activity. See: Activity Cost: page 4 – 47.
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 – 85.

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 – 59](#).
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 – 17](#).
11. The function activity now appears beneath Functions in the navigator tree. You can review or edit the properties of this activity at any time by double-clicking on the activity in the navigator tree or by selecting the activity and choosing Properties from the Edit menu or by pressing Enter on your keyboard.
12. If your function requires input arguments, you can expose those arguments in Oracle Workflow Builder as attributes of the function activity. Function activity attributes behave as parameters whose values you can modify for each usage of the activity in a process. Function activity attributes are specific to a function activity and are not global to a process. See: [To Define an Item Type or Activity Attribute: page 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 – 17](#).

## See Also

[Using the Edit Button in a Property Page: page 3 – 8](#)

► To Create an Event Activity

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

- Internal Name:** A text input field.
- Display Name:** A text input field.
- Description:** A text input field.
- Icon:** A dropdown menu showing 'EVENT.ICO', a lightning bolt icon, and a 'Browse' button.
- Event Action:** A dropdown menu showing 'Receive'.
- Event Filter:** A text input field.
- Cost:** A text input field showing '0.00'.

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

1. Select the item type that you want to create an event for in the navigator tree, then choose New Event from the Edit menu. Define your event activity in the Activity property page that appears.
2. An event activity must have an Internal Name (all uppercase and no leading/trailing spaces) and a Display Name, which is the translatable name that appears in your process diagram. Use the description to provide an explanation about this activity.



**Attention:** To update the internal name for an activity once it is defined, you must use a special SQL script called wfchact.sql. You should only use this script to correct errors in an activity's internal name during design time. Do not use this script to rename activities that are involved in running instances of processes. See: Wfchact.sql: page 16 – 7.

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

3. Choose an icon that identifies your activity. You can use any icon, as long as the icon is stored in a .ico file, to symbolize the action of an activity. See: Adding Custom Icons to Oracle Workflow: page 2 – 85.

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

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

4. Select the Event Action for the activity.
  - Receive—Receive an event from the Business Event System.
  - Raise—Raise an event to the Business Event System.
  - Send—Send an event directly from one Event agent to another agent without re-raising the event to the Business Event System.

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

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

When you include the event activity as a node in a process, you can use the item type attributes to specify where to store or retrieve the required event detail information for that node.

The item type attributes that you use for event details must be associated with the same item type that the event activity itself is associated with. See: *To Define an Item Type or Activity Attribute*: page 4 – 8 and *To Define Event Details for an Event Node*: page 5 – 12.

5. If you are defining a Receive event activity, you can optionally enter an Event Filter to specify the event that the activity can receive.

- To allow only a specified event for the activity, enter the full internal event name.

**Note:** You can only specify an individual event as the event filter. The event filter cannot be an event group.

- To allow any event for the activity, leave the Event Filter field blank.

See: *To Define an Event*: page 13 – 5.

6. Enter an optional cost for the activity. For event activities with the event actions Raise or Send, you can use the cost to defer long

running activities to a background engine. See: Activity Cost: page 4 – 47.

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 – 59.
9. Select the Roles tab page to specify the roles that have access to this function activity. (This functionality will be supported in a future release.)
10. Select the Access tab page to set the access levels allowed to modify this event. See: Allowing Access to an Object: page 4 – 17.
11. The event activity now appears beneath Events in the navigator tree. You can review or edit the properties of this activity at any time by double-clicking on the activity in the navigator tree or by selecting the activity and choosing Properties from the Edit menu or by pressing Enter on your keyboard.
12. For a Raise event activity, if the event raised by the activity requires additional parameters to be included in the event message, you can define those parameters as attributes of the Raise event activity. When the event is raised, the activity attributes are set as parameters in the parameter list for the event message. If the event message is later received by another process, the Workflow Engine sets the event parameters as item type attributes for that process. You can modify the values of the attributes for each usage of the Raise event activity in a process. Event activity attributes are specific to an event activity and are not global to a process. See: To Define an Item Type or Activity Attribute: page 4 – 8.

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

When you include an event activity as a node in a process, you can assign a value to the event activity attribute that is specific to that node. See: To Define Activity Attribute Values: page 5 – 17.

**Note:** A Raise event activity also automatically sets the item type and item key for the current workflow process in the parameter list for the event message. If the event message is later received by another process, the Workflow Engine uses that item type and item key to automatically set the process

that raised the event as the parent for the process that receives the event. See: SetItemParent: page 8 – 79.

### ► 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.

The screenshot shows the 'Navigator Control Properties' dialog box. It has four tabs: 'Activity', 'Details', 'Roles', and 'Access'. The 'Activity' tab is active. The dialog contains the following fields and controls:

- Internal Name: Text input field.
- Display Name: Text input field.
- Description: Text input field.
- Icon: A dropdown menu showing 'PROCESS.ICO', a gear icon, and a 'Browse' button.
- Result Type: A dropdown menu showing '<None>' and an 'Edit' button.
- Runnable: A checked checkbox.

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

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

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 called wfchact.sql. You should only use this script to correct errors in an activity's internal name during design time. Do not use this script to

rename activities that are involved in running instances of processes. See: Wfchact.sql: page 16 – 7.

**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 – 20.

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

4. Choose an icon that identifies your activity. You can use any icon, as long as the icon is stored in a .ico file, to symbolize the action of an activity. See: Adding Custom Icons to Oracle Workflow: page 2 – 85.

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

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

5. Check Runnable so that the process that this activity represents can be initiated as a top-level process and run independently. If your process activity represents a subprocess that should only be executed if it is called from a higher level process, then uncheck Runnable. See: CreateProcess: page 8 – 21.

**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 – 59.
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 – 17.
9. Choose Apply to save your changes, OK to save your changes and close the property page or Cancel to cancel your changes and close the property page.

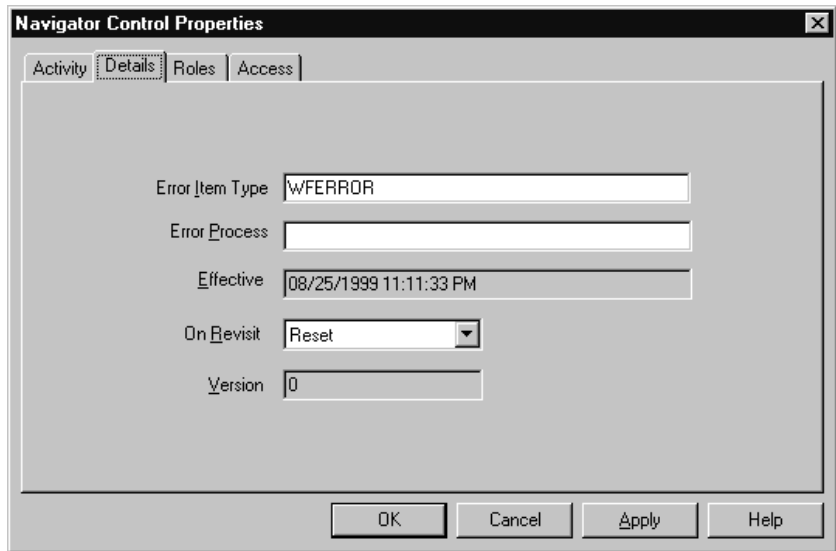


10. The process activity now appears beneath Processes in the navigator tree. You can review or edit the properties of this activity at any time by selecting the activity and choosing Properties from the Edit menu or by pressing Enter on your keyboard.

## See Also

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

### ► To Define Optional Activity Details



The screenshot shows a dialog box titled "Navigator Control Properties" with a close button (X) in the top right corner. The dialog has four tabs: "Activity", "Details", "Roles", and "Access". The "Details" tab is selected. The dialog contains the following fields and controls:

- Error Item Type:** A text input field containing the value "WFERROR".
- Error Process:** An empty text input field.
- Effective:** A text input field containing the date and time "08/25/1999 11:11:33 PM".
- On Revisit:** A dropdown menu with "Reset" selected.
- Version:** A text input field containing the value "0".

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

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 – 26.
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 – 10.

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

#### ► To Copy an Activity

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

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

4. When you are done, choose OK.

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

---

## 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 – 20.
2. Create a voting message that prompts a recipient to respond with one of the values in the voting lookup type. Complete the Result tab for the message. Set the lookup type in the Result tab to the voting lookup type defined in Step 1. See: To Create a Message: page 4 – 29
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 called `wfchact.sql`. You should only use this script to correct errors in an activity's internal name during design time. Do not use this script to rename activities that are involved in running instances of processes. See: `Wfchact.sql`: page 16 – 7.

**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 – 43.
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 – 10.  
  
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 – 3.
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 – 59.

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 – 17.
14. If you use the `WF_STANDARD.VOTEFORRESULTTYPE` tallying function, create a custom activity attribute of type Number for each possible voting response. Remember that each possible voting response is a lookup code associated with the voting activity's result type. Hence, when you define your custom activity attribute, the internal name of the activity attribute must match the internal name of the lookup code, that is, the response value.

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

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

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

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

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

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

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

## See Also

Vote Yes/No Activity: page 6 – 10

### Example Voting Methods

---

#### 1. Simple Majority

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

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

Table 4 – 3 (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

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

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

Table 4 – 4 (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

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

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

Table 4 – 5 (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

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

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

Table 4 – 6 (Page 1 of 1)

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

#### 5. Black Ball

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

Response	Custom Response Activity Attribute Value
YES	100
NO	0

Table 4 – 7 (Page 1 of 1)

Any vote for response NO makes NO the result.

#### 6. Jury

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

Response	Custom Response Activity Attribute Value
GUILTY	100
NOT_GUILTY	100

Table 4 – 8 (Page 1 of 1)

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



## See Also

Vote Yes/No Activity: page 6 – 10

---

## 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.

---

## Modifying Objects in Oracle Workflow Builder

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

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

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

Table 4 – 9 (Page 1 of 1)

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

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

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

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

If the modified object is no longer compatible with the rest of the workflow definition used by the work item, errors may arise. To avoid such errors, you must take all references to the object into consideration

when planning your changes to ensure that the changes do not cause any incompatibility.

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

---

## Workflow Objects That Support Versioning

The following Workflow objects support versioning:

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

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

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

For example:

- If you update a notification activity to reference a new message, the notification will be versioned.
- If you update a function activity to reference a new lookup type, the function will be versioned.
- If you update a function activity to reference a new API, the function will be versioned.

- If you remove a process activity, or node, from a process diagram, the owning process will be versioned, as well as all the process activities that exist within the process.
- If you add an activity attribute to an activity, the owning activity will be versioned.

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

---

## Workflow Objects That Do Not Support Versioning

The following Workflow objects do not support versioning:

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

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

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

**Note:** The Workflow Builder does not support the editing of PL/SQL code. PL/SQL code is listed as a Workflow object here solely for the purpose of explaining the consequences of changing the code referenced by a Workflow function activity.

### **Item Attributes**

---

When a work item is initiated, Oracle Workflow creates a runtime copy of each item attribute that is defined for that item type. The Workflow Engine refers to these runtime copies whenever an item attribute is referenced in the PL/SQL code for a function activity in the workflow process.

Adding a new item attribute after work items have been initiated will not affect the active work items. However, these work items will not

include the new item attribute unless you specifically create the attribute for them by calling the *AddItemAttr()* or *AddItemAttributeArray* APIs. If you also add references to the new item attribute in the existing PL/SQL code within the workflow process, those references may cause errors in active work items that do not include the attribute.

For example, if you change the PL/SQL API for a function activity by calling a Workflow Engine API to communicate with a new item attribute, the function activity will fail for active work items that do not have the new item attribute defined.

You should plan carefully when making any modifications to the existing PL/SQL code within a workflow process to ensure that you do not add references to a new item attribute without also creating the item attribute for active work items that require it. See: PL/SQL Code: page 4 – 74.

**Note:** You can, however, add references to new item attributes in the API that starts a workflow process, without making special provisions for active work items. For example, you can call the *SetItemAttribute* or *SetItemAttributeArray* APIs to populate the new item attributes at the start of the process. Active work items will not be affected by such changes, since these work items have already passed through this code.

## Messages

---

When the Workflow Engine requests the Notification System to send a message, the Notification System creates a notification attribute in the notification tables for every message attribute. The notification attribute rows contain the variable data that will be token-replaced into the message body, including the subject line and body text, at runtime.

The message body, however, is not copied into the notification tables. Instead, the message body is referenced by the various Notification System APIs at runtime, when the notification is displayed to the user. As a result, any modifications to a message body will affect notifications in active work items that were sent before the change, as well as notifications that are sent after the change.

You can make certain types of modifications to a message body without risking incompatibility errors. These modifications include:

- Adding static text
- Editing static text
- Removing static text
- Removing message attribute tokens

For example, if you add a static sentence such as "Please approve within five days" to a message body, all notifications in active work items will include the additional sentence when you access the notifications. The Notification System can display the modified message body without any errors because no further information is required to resolve the additional sentence.

However, inappropriate modifications, such as adding tokens for newly created message attributes, may cause notifications in active work items to be displayed incorrectly. You should plan your changes carefully to avoid errors.

If you need to add tokens for new message attributes to a message body, you should implement the change by creating a new message rather than by modifying the existing message. You can attach the new message to your existing notification activity without affecting active work items, since notification activities support versioning.

For example, if you create a new message attribute such as *Approver Name* and you add a token for that attribute in the message body, all notifications in active work items will include the new token when you access the notifications. However, notifications that were sent before the change will not include the new message attribute *Approver Name* as a notification attribute. The Notification System will not be able to resolve the reference to the new message attribute and will display the token "&APPROVER\_NAME" in the notifications instead.

In this example, instead of modifying the original message body, you should create a new message that includes the new message attribute, add the token for the new attribute to the body of the new message, and attach the new message to your notification activity. When you save your changes, the notification activity will be versioned. Then active work items will continue to reference the old version of the notification activity, and the incompatible token will not appear in those notifications.

## Lookup Types and Codes

---

Lookup types have the following important uses in Oracle Workflow:

- Determining the possible completion statuses (lookup codes) for workflow activities
- Determining the possible completion statuses (lookup codes) for 'Respond' message attributes.

Inappropriate modifications to lookup types may cause active work items that reference those lookup types to fail.

To avoid errors caused by backward incompatibility, follow these guidelines for lookup types that are referenced by active work items:

- Do not delete lookup types.
- Do not delete lookup codes from existing lookup types.
- Do not add lookup codes to existing lookup types.

If you need to make any of the above changes, you should implement the change by creating a new lookup type rather than by modifying the existing lookup type.

You can attach new lookup types to existing activities without affecting active work items, since activities support versioning. However, you should not attach new lookup types to existing message attributes, since Workflow messages do not support versioning.

The following examples show some errors that can be caused by inappropriate lookup type modifications:

- If you add a lookup code to a lookup type that is referenced by a 'Respond' message attribute, the new lookup code will be available for users to select as a response to a notification. However, previous versions of the notification activity will not have branching logic modeled for the new lookup code. If a user selects the new code, the process will enter an 'ERROR' state.
- If you delete a lookup code from a lookup type that is referenced by a 'Respond' message attribute, users will no longer be able to choose that result code to respond to a notification.

## PL/SQL Code

---

Although function activities support versioning, the underlying PL/SQL code does not support versioning, unless you implement versioning for your code yourself. Modifying PL/SQL code without versioning changes the business flow for active work items that reference that code. Inappropriate modifications may cause these work items to fail.

To prevent changes in the PL/SQL API for a function activity from affecting active work items, you should implement the changes by creating a new API rather than by modifying the existing API. You can attach the new API to your existing function activity without affecting active work items, since function activities support versioning.

If you need to modify an existing API and you cannot create a new API instead, you should plan your changes carefully to ensure that the changes do not cause any backward incompatibility.



For example, if you plan to add a lookup code to the group of values that an API can return, you should first ensure that the function activity node has an outgoing transition, such as 'Default,' that the Workflow Engine can follow when the API returns that value. Otherwise, the process will enter an 'ERROR' state when that value is returned. If there is no appropriate outgoing transition, you must implement the change in a new API and update the process to model branching logic for the additional return value.

Also, if you change the existing PL/SQL API for a function activity by calling a Workflow Engine API to communicate with a new item attribute, you should ensure that you also create the new item attribute for active work items. Otherwise, the function activity will fail for active work items which do not have the new item attribute defined.

Calls to any of the following APIs with newly created item attributes as parameters may cause the function activity to fail if active work items do not have the item attributes defined:

- wf\_engine.SetItemAttrText
- wf\_engine.SetItemAttrNumber
- wf\_engine.SetItemAttrDate
- wf\_engine.SetItemAttrEvent
- wf\_engine.SetItemAttrTextArray
- wf\_engine.SetItemAttrNumberArray
- wf\_engine.SetItemAttrDateArray
- wf\_engine.GetItemAttrText
- wf\_engine.GetItemAttrNumber
- wf\_engine.GetItemAttrDate
- wf\_engine.GetItemAttrEvent

To create copies of a new item attribute for active work items, call *AddItemAttr()* or one of the *AddItemAttributeArray* APIs. You can place this call either in a custom upgrade script or in an exception handler.

- **Upgrade script** – Before you modify your API, write and execute a custom upgrade script that creates and populates the new item attribute for any active work items that reference that API. The following example shows one way to structure an upgrade script.

```

for <each active work item>
begin
    wf_engine.AddItemAttr(itemtype,
                          itemkey,
                          '<new_attribute_name>');
    wf_engine.SetItemAttrText(itemtype,
                              itemkey,
                              '<new_attribute_name>',
                              '<New attribute value>');

end;
end loop;

```

**Note:** Active work items are identified as those items for which WF\_ITEMS.END\_DATE is null.

- **Exception handler** – After the reference to the new item attribute in your modified API, add an exception handler to create and populate the attribute when the attribute does not already exist. The following example shows one way to structure such an exception handler.

```

procedure <procedure_name>(
    itemtype in varchar2,
    itemkey in varchar2,
    actid in number,
    funcmode in varchar2,
    result in out varchar2)
is
begin
    --
    -- RUN mode - normal process execution
    --
    if (funcmode = 'RUN') then
        -- your run code goes here
        null;
        wf_engine.SetItemAttrText(itemtype,
                                  itemkey,
                                  '<existing_attribute_name>',
                                  '<Existing attribute value>');

        begin
            wf_engine.SetItemAttrText(itemtype,
                                      itemkey,
                                      '<new_attribute_name>',
                                      '<New attribute value>');

```

```

exception
when others then
    if (wf_core.error_name = 'WFENG_ITEM_ATTR') then
        wf_engine.AddItemAttr(itemtype,
                               itemkey,
                               '<new_attribute_name>');
        wf_engine.setitemattrtext(itemtype,
                                   itemkey,
                                   '<new_attribute_name>',
                                   '<New attribute value>');
    else
        raise;
    end if;
end;
-- example completion
result := 'COMPLETE: ';
return;
end if;

--
-- CANCEL mode - activity 'compensation'
--
-- This is in the event that the activity must be undone,
-- for example when a process is reset to an earlier point
-- due to a loop back.
--
if (funcmode = 'CANCEL') then
    -- your cancel code goes here
    null;
    -- no result needed
    result := 'COMPLETE';
    return;
end if;

--
-- Other execution modes may be created in the future. Your
-- activity will indicate that it does not implement a mode
-- by returning null
--
result := '';
return;

exception
when others then

```

```
-- The line below records this function call in the error
-- system in the case of an exception.
wf_core.context('<package_name>',
                '<procedure_name>',
                itemtype,
                itemkey,
                to_char(actid),
                funcmode);

    raise;
end <procedure_name>;
```

## See Also

Item Attributes: page 4 – 71

CHAPTER

# 5

## Defining a Workflow Process Diagram

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

---

### Transitions

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

For an activity that has a defined result type, you must associate the transition arrow that you create with one of the activity's possible results. The result that the activity returns when it completes then determines what the next eligible activity is, as defined by the results-based transitions that originate from the completed activity. For example, "Notify Approver" with a result of 'REJECTED' transitions to "Reject Requisition." See: Requisition Process Activities: page 15 – 15.

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 – 43.

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 – 8.

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 – 59.

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

### **Creating Multiple Transitions to a Single Activity**

---

You can create multiple transitions to a single activity in a process diagram. Sometimes these multiple transitions indicate that there are multiple ways that the process can transition to this one node 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 – 10 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 or when a Business Event System subscription sends an event to launch the process. A subprocess is started when the Workflow Engine transitions to a process activity that represents the subprocess.

To launch a workflow process using the Business Event System, follow these steps:

1. Define a business event.



2. Define a subscription to this business event. In the subscription properties, specify the workflow item type and process that you want to launch.

By default, Oracle Workflow uses the event key as the item key for the workflow process that is launched. If you want to generate the item key based on a custom rule, create a function that populates the correlation ID in the event message with the item key you want, and assign that function as the subscription's rule function.

3. Add the Raise() API to your custom application code at the point where you want to launch the workflow process.

## See Also

Workflow Engine APIs: page 8 – 19

Raise: page 8 – 261

Managing Business Events: page 13 – 2

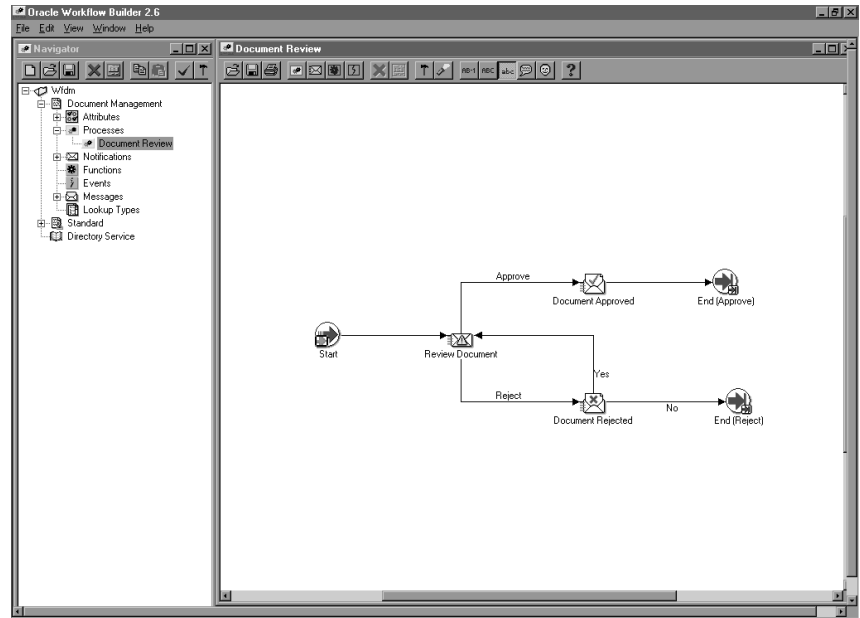
### **Diagramming a Process**

---

This section discusses how to draw and define a workflow process in the Process window:

- To Add Nodes to a Workflow Process: page 5 – 6
- To Define Nodes: page 5 – 8
- To Define Event Details for an Event Node: page 5 – 12
- To Define Activity Attribute Values: page 5 – 17
- To Create and Edit a Transition: page 5 – 18
- To Display a Process Overview: page 5 – 19
- To Print a Process: page 5 – 20
- To Copy a Process Diagram to the Clipboard: page 5 – 20
- To Validate a Process Definition: page 5 – 21

## ► 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, event, or process activity from the navigator tree into the Process window. The activity you drag must belong to the same data store as the process you are dragging it to.

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

- Choose the New Function, New Process, New Event, or New Notification toolbar button to create a new activity.
  - 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, event, 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 – 8.
  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 – 18.

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

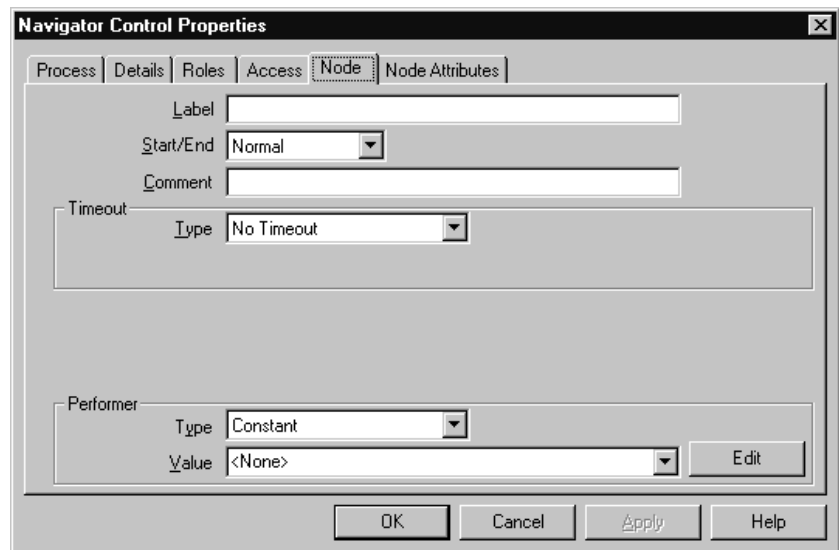
1. Open the Process window for your process activity.
2. To create a new function, notification, event, or process node, first select the New Function, New Notification, New Event, or New Process icon from the Process window toolbar. Next, click on the position within the Process window where you want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

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

3. In the Item Type field, select the item type that you want this activity node to be associated with.
4. Choose one of the following methods to define the remaining information for the node.
  - Select either the internal name or display name of a predefined activity. Oracle Workflow Builder then populates all the fields with predefined information from the master activity as shown in the Navigator window.

- Alternatively, choose the New button to define a new activity. To complete the following tabs of the property page, refer to the sections listed:
  - Process—To Create a Process Activity: page 4 – 57
  - Function—To Create a Function Activity: page 4 – 50
  - Notification—To Create a Notification Activity: page 4 – 48
  - Event—To Create an Event Activity: page 4 – 54
  - Details—To Define Optional Activity Details: page 4 – 59
  - Roles—The information in this tab is currently not supported.
  - Access—To Set the Access Level for an Object: page 4 – 18

**Caution:** Any changes that you make to the any of the above tabs automatically propagate to the master activity and affect all other instances of that activity. Only changes that you make to the Node and Node Attributes tabs, and to the Event Details tab for an event activity, are local and specific to the current node activity.



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

activity is used more than once in the process, *-N* is appended to the activity name, where *N* represents the 'Nth' 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 – 19.

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 that requires a response, a process activity that is a subprocess within another process, an event activity with an event action of Receive, or a blocking function activity, specify whether the activity must be completed by some specified time. If the activity is not completed by a given time, you can redirect the parent process to transition to a different activity. See: Timeout Transitions: page 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, or for an event activity node with an event action of Send, you can override the priority assigned to the activity's message. Choose 'Default' to keep the default priority of the message.

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

Choose 'Item Attribute' to override the default priority with a new priority level that is dynamically determined at runtime. Note that you must first create an item attribute of type number to store the computed priority value and reference that predefined item attribute here. See: Item Type Attributes: page 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 – 24.

**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 – 211.

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

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

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

13. If the node is an event activity, you can specify additional required event information by choosing the Event Details tab. See: To Define Event Details for an Event Node: page 5 – 12.
14. If the node is a function, notification, or event activity and the activity has activity attributes, you can assign values to those activity attributes by choosing the Node Attributes tab. See: To Define Activity Attribute Values: page 5 – 17.
15. If the node is a process activity, then a small subprocess overlay icon appears over the upper right corner of process activity icon. The subprocess overlay icon identifies the node as a subprocess within the process diagram.

## See Also

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

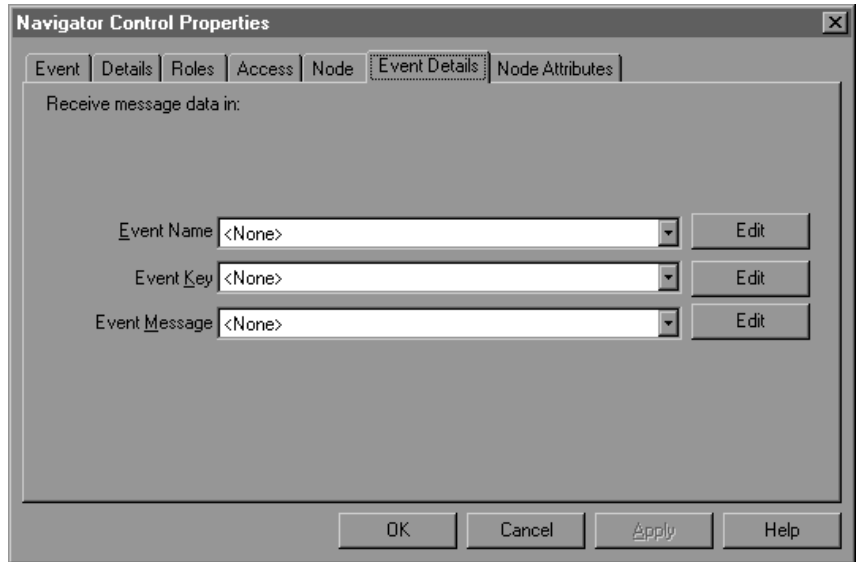
Using the Edit Button in a Property Page: page 3 – 8

### ► To Define Event Details for an Event Node

The event action defined for the event activity determines which event details you must define for an event node. For each event detail, it is either required or optional to use an item type attribute to store or retrieve the detail information. Note that you must first create item type attributes of the appropriate types before you can reference those predefined item attributes here. The item type attributes you use for event details must be associated with the same item type that the event activity itself is associated with. See: Item Type Attributes: page 4 – 2 and To Define an Item Type or Activity Attribute: page 4 – 8.



1. Display the property pages of an event activity node. Select the Event Details tab.



2. For an activity with the event action Receive, enter the following event details:

- Event Name—Optionally select an item type attribute of type text where you want to store the event name that the node receives.

**Note:** The event activity can only receive events that match the event name specified as the event filter. See: *To Create an Event Activity*: page 4 – 54.

- Event Key—Optionally select an item type attribute of type text where you want to store the event key that the node receives.
- Event Message—Optionally select an item type attribute of type event where you want to store the event message that the node receives.

**Note:** When the activity receives an event, the Workflow Engine stores the event name, event key, and event message in the item type attributes you specify, and also sets any parameters in the event message parameter list as item type attributes for the process, creating new item type attributes if a corresponding attribute does not already exist for any parameter. See: *To Define Optional Activity Details*: page 4 – 59.

Additionally, if the event was originally raised by a Raise event activity in another workflow process, the item type and item key for that process are included in the parameter list within the event message. In this case, the Workflow Engine automatically sets the specified process as the parent for the process that receives the event, overriding any existing parent setting. See: SetItemParent: page 8 – 79.

The screenshot shows the 'Navigator Control Properties' dialog box with the 'Event Details' tab selected. The 'Raise event as:' section contains the following fields:

- Event Name:** A group box containing a 'Type' dropdown menu set to 'Constant' and a 'Value' text input field.
- Event Key:** A dropdown menu set to '<None>' with an 'Edit' button to its right.
- Event Data:** A dropdown menu set to '<None>' with an 'Edit' button to its right.

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

3. For an activity with the event action Raise, enter the following event details:
  - **Event Name**—Enter the name of the event that the node raises. You can either specify a constant event name or select an item type attribute of type text that dynamically determines the event name at runtime.

**Note:** You can only raise an individual event. You cannot raise event groups.
  - **Event Key**—Select the item type attribute of type text that contains the event key for the event that the node raises.
  - **Event Data**—Optionally select an item type attribute of type text that contains the event data for the event that the node raises.

The maximum length of the data you can enter in a text attribute is 4000 bytes. If the event data exceeds 4000 bytes, you should assign a Generate function in the event definition to generate the

event data, rather than providing the event data through a text attribute. See: To Define an Event: page 13 – 5.

**Note:** The Event Name and Event Key are required for a Raise event activity.

**Note:** In addition to these event details, you can use the activity attributes for a Raise event activity node to specify parameters that you want to include in the parameter list for the event message. If the event message is later received by another process, the Workflow Engine sets the event parameters as item type attributes for that process. See: To Define Activity Attribute Values: page 5 – 17.

Also, a Raise event activity automatically sets the item type and item key for the current workflow process in the parameter list for the event message. If the event message is later received by another process, the Workflow Engine uses that item type and item key to automatically set the process that raised the event as the parent for the process that receives the event. See: SetItemParent: page 8 – 79.

The screenshot shows the 'Navigator Control Properties' dialog box with the 'Event Details' tab selected. The dialog has several sections for configuring event details:

- Send message as:** A dropdown menu for 'Event Message' set to '<None>' with an 'Edit' button.
- Event Name:** A section with a 'Type' dropdown set to 'Constant' and a 'Value' text field.
- Event Key:** A dropdown menu set to '<None>' with an 'Edit' button.
- Out Agent:** A section with a 'Type' dropdown set to 'Constant' and a 'Value' text field.
- To Agent:** A section with a 'Type' dropdown set to 'Constant' and a 'Value' text field.

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

4. For an activity with the event action Send, enter the following event details:
  - Event Message—Select the item type attribute of type event that contains the event message that the node sends.

- **Event Name**—Optionally enter the name of the event that the node sends. You can either specify a constant event name or select an item type attribute of type text that dynamically determines the event name at runtime. The event name that you enter here overrides the previous event name value in the event message.
- **Event Key**—Optionally select an item type attribute of type text that contains the event key of the event that the node sends. The event key that you enter here overrides the previous event key value in the event message.
- **Out Agent**—Optionally enter the outbound agent from which the node sends the event. Specify both the agent name and the system name for the agent using the following format:

`<agent_name>@<system_name>`

You can either specify a constant Out Agent name or select an item type attribute of type text that dynamically determines the Out Agent name at runtime. The Out Agent that you enter here overrides the previous outbound agent value in the event message.

- **To Agent**—Optionally enter the inbound agent to which the node sends the event. Specify both the agent name and the system name for the agent using the following format:

`<agent_name>@<system_name>`

You can either specify a constant To Agent name or select an item type attribute of type text that dynamically determines the To Agent name at runtime. The To Agent that you enter here overrides the previous inbound agent value in the event message.

**Note:** The Event Message is required for a Send event activity. Additionally, you must either include a To Agent or a From Agent within the event message, or specify a To Agent or an Out Agent in the event details for this node. If you neither specify an inbound agent nor an outbound agent, the event cannot be sent.

**Note:** If no correlation ID is initially specified in the event message, Oracle Workflow automatically sets the correlation ID to the item key of the process.

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

## See Also

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

Event Activity: page 4 – 45

To Create an Event Activity: page 4 – 54

### ► To Define Activity Attribute Values

Activity attribute values for a function or notification activity are used by the PL/SQL stored procedure that the activity calls. Activity attribute values for a Raise event activity are set as parameters in the parameter list for the event message. See: To Define an Item Type or Activity Attribute: page 4 – 8.

The screenshot shows the 'Navigator Control Properties' dialog box with the 'Node Attributes' tab selected. The dialog has a title bar with a close button. Below the title bar are tabs for 'Process', 'Details', 'Roles', 'Access', 'Node', and 'Node Attributes'. The 'Node Attributes' tab is active and contains a section labeled 'Attribute' with a 'Name' dropdown menu. Below this is a table with columns for 'Name', 'Value Type', 'Value', 'Type', and 'Description'. The table is currently empty. At the bottom of the dialog are buttons for 'OK', 'Cancel', 'Apply', and 'Help'.

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 of 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**

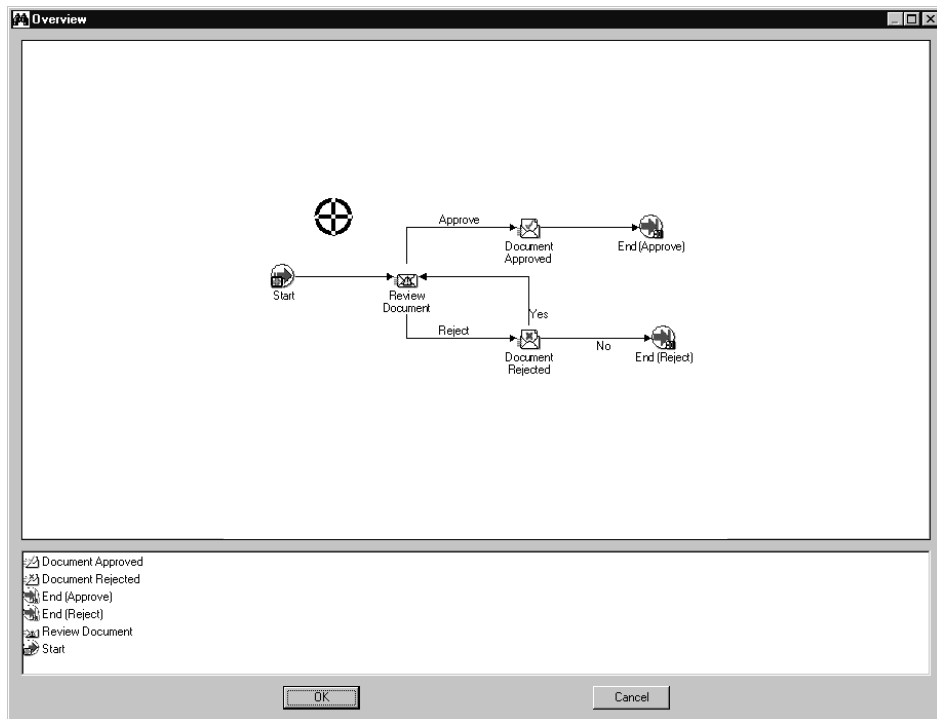
1. To create a transition between two activities, hold down your right mouse button and drag your mouse from a source activity to a destination activity.

**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 selected as the active window, choose Print Diagram from the File menu or from the right mouse button menu.

The Print Diagram option captures your process diagram as a picture (metafile), enlarges it to the correct size to print and sends it to a printer. If your diagram is large, it may span more than one page when printed. However, depending on the printer driver you use, you may get a Print dialog box that lets you scale your image down to one page for printing.

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

#### ► **To Copy a Process Diagram to the Clipboard**

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

This copies the process to the clipboard in the form of a metafile and a bitmap diagram.

3. To paste the metafile-version or bitmap-version of the process diagram into another application window, you should consult the other application's documentation on how to paste metafiles or bitmaps.

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



## ► To Validate a Process Definition

1. Choose Verify from the File menu to validate all process definitions for the currently selected data store.
2. The following list is an example of some of the validation that the Verify command performs:
  - Checks that a process has at least one Start and one End activity.
  - Verifies that a process does not contain itself as a process activity.
  - 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.

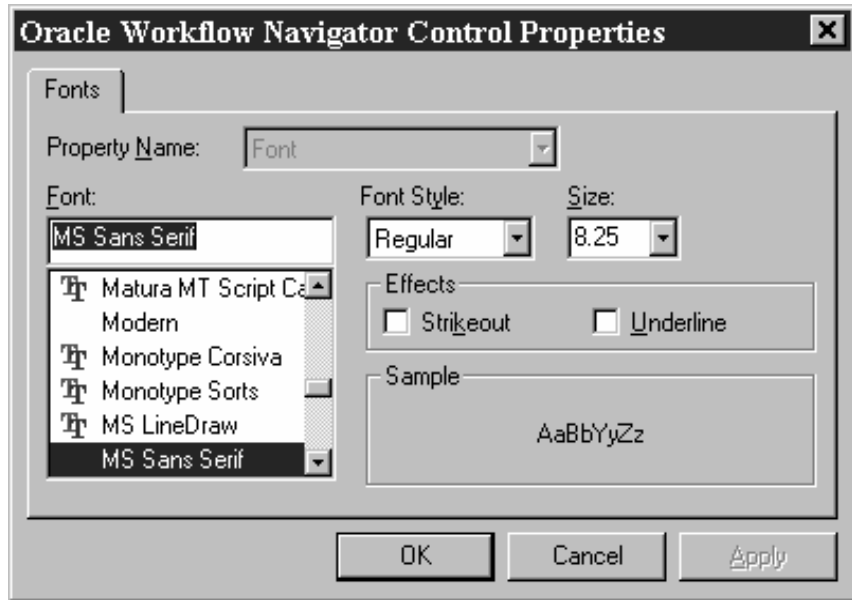


**Attention:** You should always validate any new process definition you create as it helps you to identify any potential problems with the definition that might prevent it from executing successfully.

---

## 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 – 21 and *To Define Nodes in a Process*: page 5 – 8.

**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 – 3 and *Workflow Directory Service APIs*: page 8 – 121.

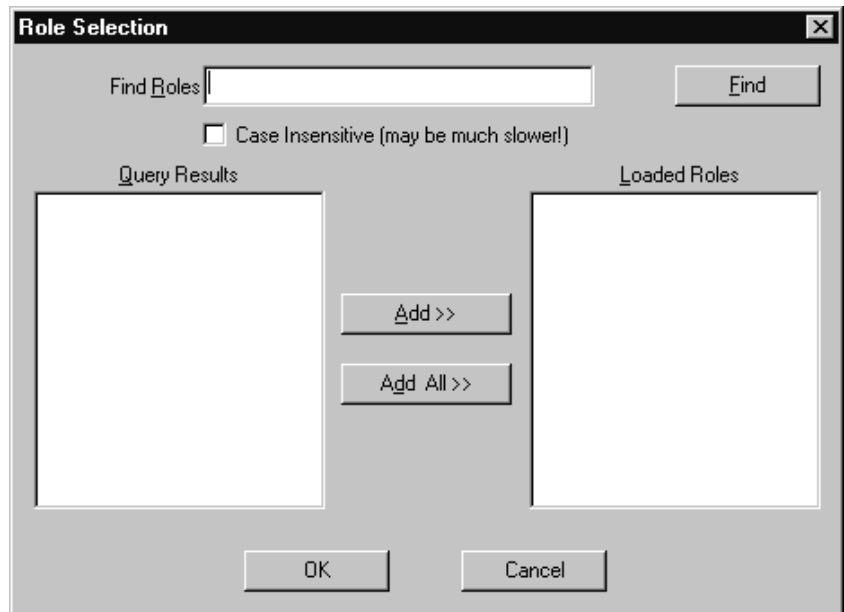
## See Also

To Load Roles: page 5 – 25

To Display the Directory Service in Oracle Workflow Builder: page 5 – 26

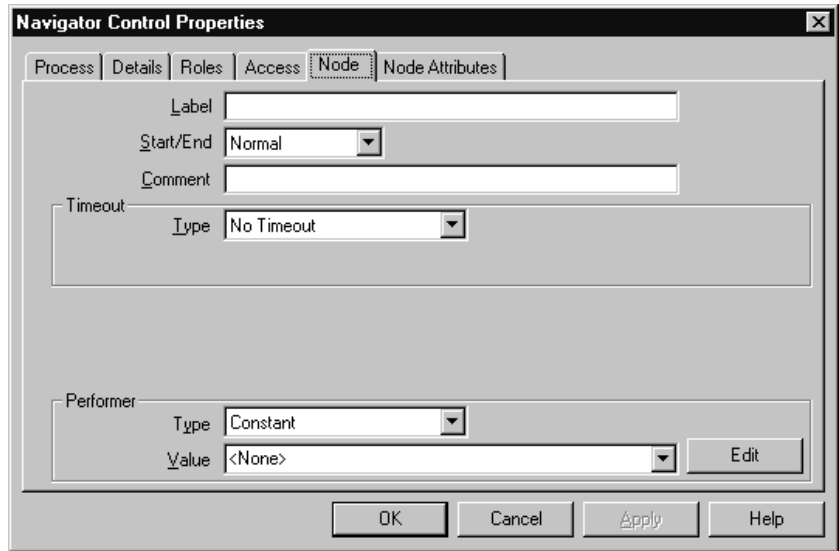
► **To Load Roles**

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 – 25.
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** [X]

Role

Internal Name

Display Name

Description

Language  Territory

Email Address  Fax

Notification Preference

Status  Expiration

OK Cancel Apply Help





CHAPTER

# 6

## Predefined Workflow Activities

**T**his 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 – 17.

---

## 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 – 17.

---

## **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 Time, enter a relative time expressed in *<days>.<fraction of days>*. For example, enter 0.5 for a wait time of half a day (12 hours).
- Day of Month—If Wait Mode is set to Day of Month, choose a day of the month from the list. If the day you choose has already passed in the current month, then the activity waits until that day in the following month.
- Day of Week—If Wait Mode is set to Day of Week, choose a day of the week from the list. If the day you choose has already passed in the current week, then the activity waits until that day in the following week.
- Time of Day—The Wait activity always pauses until midnight of the time specified, unless you use this Time of Day activity attribute to specify a time other than midnight that the Wait activity should pause until.

See: To Define Activity Attribute Values: page 5 – 17.

---

## 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 – 69

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 – 17.

---

## 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*.

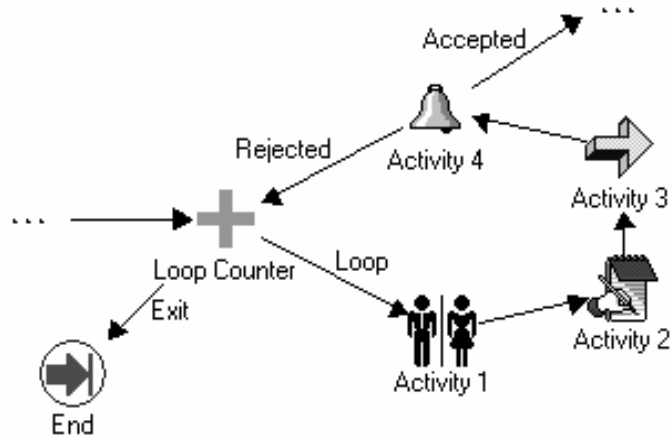
---

### 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, 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 – 17.

In this example, the engine moves from the Loop Counter activity through Activities 1, 2, 3, and 4 in the loop. If the result of Activity 4 is 'Accepted,' the process moves along the 'Accepted' transition. Otherwise, if the result is 'Rejected,' the process moves along the 'Rejected' transition to return to the Loop Counter activity. If the item is rejected multiple times, once the number of visits to the Loop Counter activity exceeds the Loop Limit value, the process moves along the 'Exit' transition and ends.




---

## 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 – 17.

---

## 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 want to define an activity that always sends the same message, you should define a notification activity and not use this Notify function activity.

The Notify activity calls a PL/SQL procedure named `WF_STANDARD.NOTIFY`.

### Activity Attributes

---

The Notify activity has 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 – 17.

---

## Vote Yes/No Activity

The Vote Yes/No activity lets you send a notification to a group of users in a role and tally the Yes/No responses from those users. The results of the tally determine the activity that the process transitions to next.

The Vote Yes/No activity, classified as a notification activity, first sends a notification message to a group of users and then performs a PL/SQL post-notification function to tally the users’ responses (votes).

## Activity Attributes

---

The Vote Yes/No activity has three activity attributes:

- Percent Yes—The percentage of Yes votes cast in order for the activity to complete with a result of Yes.
- Percent No—The percentage of No votes cast in order for the activity to complete with a result of No
  - Note:** The values for the Percent Yes and Percent No attributes are both defined as null in order to use a Popularity voting method, in which the result is the response with the highest number of votes. See: Example Voting Methods: page 4 – 64
- 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 – 17.

### See Also

Voting Activity: page 4 – 61

---

## Master/Detail Coordination Activities

The Master/Detail coordination activities let you coordinate the flow of master and detail processes. For example, a master process may spawn

detail processes that need to be coordinated such that the master process continues only when every detail process has reached a certain point in its flow or vice versa.

When you spawn a detail process from a master process in Oracle Workflow, you are in effect creating a separate process with its own unique item type and item key. You define the master/detail relationship between the two processes by making a call to the Workflow Engine *SetItemParent* API after you call the *CreateProcess* API and before you call the *StartProcess* API when you create the detail process. See: *SetItemParent*: page 8 – 79.

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 – 17.

## 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:

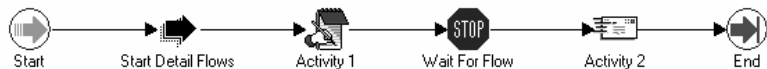
- **Waiting Flow**—specify whether the halted process that is waiting for this activity to complete is a “Master” or “Detail” flow.
- **Waiting Activity**—specify the label of the activity node in the halted process that is waiting for this activity to complete. See: *To Define Activity Attribute Values*: page 5 – 17.

## Example

---

The following figures show an example of how these coordination activities can be used. In the master process example, after the process begins with the Start activity, the Start Detail Flows activity initiates several detail processes. The master process then completes Activity 1 before it pauses at the Wait For Flow activity. Wait For Flow is defined to wait for all its detail processes to complete a Continue Flow activity before allowing the master process to transition to Activity 2 and finally end. An example of one of the detail processes below shows that after the detail process begins with the Start activity, it completes Activity A. When it reaches the Continue Flow activity, it signals to the Workflow Engine that the master process can now continue from the Wait For Flow activity. The detail process itself then transitions to Activity B and finally ends.

### 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 – 17.

---

## Get Event Property Activity

The Get Event Property activity lets you retrieve a property of an event message from the Business Event System and store the property value in an item attribute. This activity calls the PL/SQL procedure named `WF_STANDARD.GETEVENTPROPERTY`.

### Activity Attributes

---

The Get Event Property activity has four activity attributes:

- **Event**—choose the item attribute of type event that contains the event message from which you want to retrieve a property.
- **Property**—the event property whose value you want to retrieve. This attribute takes as its value a lookup code from the Event Property lookup type. Possible values are "Priority," "Send Date," "Receive Date," "Correlation ID," "Event Parameter," "Event Name," "Event Key," "From Agent," "From Agent Name," "From Agent System," "To Agent," "To Agent Name," and "To Agent System." See: Event Message Structure: page 8 – 242.
- **Event Parameter**—if you choose the Event Parameter property in the Property attribute, enter the name of the parameter whose value you want to retrieve. Oracle Workflow uses this name to identify the parameter within the event message's parameter list. If you choose any property other than Event Parameter, leave this attribute blank.
- **Item Attribute**—the item attribute where you want to store the event property value. See: To Define Activity Attribute Values: page 5 – 17.

---

## Set Event Property Activity

The Set Event Property activity lets you set the value of a property in an event message from the Business Event System. This activity calls

the PL/SQL procedure named *WF\_STANDARD.SETEVENTPROPERTY*.

### **Activity Attributes**

---

The Set Event Property activity has six activity attributes:

- **Event**—choose the item attribute of type event that contains the event message whose property you want to set.
- **Property**—the event property whose value you want to set. This attribute takes as its value a lookup code from the Event Property lookup type. Possible values are "Priority," "Send Date," "Receive Date," "Correlation ID," "Event Parameter," "Event Name," "Event Key," "From Agent," "From Agent Name," "From Agent System," "To Agent," "To Agent Name," and "To Agent System." See: Event Message Structure: page 8 – 242.
- **Event Parameter**—if you choose the Event Parameter property in the Property attribute, enter the name of the parameter whose value you want to set. Oracle Workflow uses this name to identify the parameter within the event message's parameter list. If you choose any property other than Event Parameter, leave this attribute blank.
- **Date Value**—the value of type date that you want to set for the event property, if you choose the Send Date or Receive Date property.
- **Numeric Value**—the value of type number that you want to set for the event property, if you choose the Priority property.
- **Text Value**—the value of type text that you want to set for the event property, if you choose the Correlation ID, Event Parameter, Event Name, Event Key, From Agent Name, From Agent System, To Agent Name, or To Agent System property. See: To Define Activity Attribute Values: page 5 – 17.

**Note:** You must enter the value to set in the activity attribute that matches the data type of the event property you choose.

---

## **Compare Event Property Activity**

The Compare Event Property activity lets you compare a property of an event message from the Business Event System with a test value that



you specify. This activity calls the PL/SQL procedure named `WF_STANDARD.COMPAREEVENTPROPERTY`.

### **Activity Attributes**

---

The Compare Event Property activity has six activity attributes:

- **Event**—choose the item attribute of type event that contains the event message whose property you want to compare to a test value.
- **Property**—the event property whose value you want to compare to a test value. This attribute takes as its value a lookup code from the Event Property lookup type. Possible values are "Priority," "Send Date," "Receive Date," "Correlation ID," "Event Parameter," "Event Name," "Event Key," "From Agent," "From Agent Name," "From Agent System," "To Agent," "To Agent Name," and "To Agent System." See: Event Message Structure: page 8 – 242.
- **Event Parameter**—if you choose the Event Parameter property in the Property attribute, enter the name of the parameter whose value you want to compare to a test value. Oracle Workflow uses this name to identify the parameter within the event message's parameter list. If you choose any property other than Event Parameter, leave this attribute blank.
- **Date Value**—the test value of type date with which to compare the event property value, if you choose the Send Date or Receive Date property.
- **Numeric Value**—the test value of type number with which to compare the event property value, if you choose the Priority property.
- **Text Value**—the test value of type text with which to compare the event property value, if you choose the Correlation ID, Event Parameter, Event Name, Event Key, From Agent Name, From Agent System, To Agent Name, or To Agent System property.

The Compare Event Property activity uses the Comparison lookup type for a result code. Possible values are "Greater Than," "Less Than," "Equal," or "Null," if the test activity attribute value is null. You can guide your workflow process based on how the event property value compares to the test value. See: To Define Activity Attribute Values: page 5 – 17.

**Note:** You must enter the test value in the activity attribute that matches the data type of the event property you choose. If

you enter the test value in an inappropriate activity attribute, the Compare Event Property activity returns the "Null" result code.

---

## XML Get Tag Value Activity

Use the XML Get Tag Value activity to retrieve data from the contents of an event message from the Business Event System. This functionality is currently only available for the standalone version of Oracle Workflow. This activity retrieves the data contained within a particular XML tag set in the event message and stores the data in an item attribute that you specify. The XML Get Tag Value activity calls the external Java function named *oracle.apps.fnd.wf.XMLGetTagValue*.

**Note:** When the Workflow Engine encounters an external Java function activity, it places an entry on the 'Outbound' queue. To continue executing the activity, you must run the Java Function Activity Agent, which calls the appropriate Java function and places the result on the 'Inbound' queue. You must then run a background engine to process the 'Inbound' queue and complete the function activity. See: *Setting Up the Java Function Activity Agent: page 2 – 86* and *Setting Up Background Engines: page 2 – 43*.

---

### Activity Attributes

The XML Get Tag Value activity has three activity attributes:

- **Event**—choose the item attribute of type event that contains the event message from which you want to retrieve data.
- **Tag**—the tag set within the event message from which you want to retrieve data. Specify the tag set in XPath notation. For example, for an XML document containing a purchase order, the XML path for the order number tag could be specified in the following format:

```
/order/header/ordernumber
```

The following example path locates the ITEMNO node on the third line of the purchase order document:

```
/order/orderlines/line[3]/itemno
```

The following example path locates the COST node on the second line of the purchase order document whose currency attribute is set to "AUD." The notation // indicates that the

specified node is located among the descendants of the root node.

```
//line[2]/cost[@currency="AUD"]
```

For more information, see the W3C Recommendation *XML Path Language (XPath)*.

- **Item Attribute**—choose the item attribute of type date, number, or text where you want to store the data. The type of the item attribute must match the type of the data that you want to retrieve.

See: To Define Activity Attribute Values: page 5 – 17.

---

## XML Compare Tag Value Activities

Use the XML Compare Tag Value activities to compare data from an event message received through the Business Event System with a test value. This functionality is currently only available for the standalone version of Oracle Workflow. These activities compare the data contained within a particular XML tag set in the event message with the test value that you specify.

**XML Compare Tag Value (Date)** Use this activity to compare date values.

**XML Compare Tag Value (Number)** Use this activity to compare number values.

**XML Compare Tag Value (Text)** Use this activity to compare text values.

All the XML Compare Tag Value activities call the external Java function named *oracle.apps.fnd.wf.XMLCompareTag*.

**Note:** When the Workflow Engine encounters an external Java function activity, it places an entry on the 'Outbound' queue. To continue executing the activity, you must run the Java Function Activity Agent, which calls the appropriate Java function and places the result on the 'Inbound' queue. You must then run a background engine to process the 'Inbound' queue and complete the function activity. See: Setting Up the Java Function Activity Agent: page 2 – 86 and Setting Up Background Engines: page 2 – 43.

## Activity Attributes

---

Each XML Compare Tag Value activity has three activity attributes:

- **Event**—choose the item attribute of type event that contains the event message whose data you want to compare.
- **Tag**—the tag set within the event message that contains the data you want to compare to a test value. Specify the tag set in XPath notation. For example, for an XML document containing an order, the XML path for the order number tag could be specified in the following format:

```
/order/header/ordernumber
```

The following example path locates the ITEMNO node on the third line of the purchase order document:

```
/order/orderlines/line[3]/itemno
```

The following example path locates the COST node on the second line of the purchase order document whose currency attribute is set to "AUD." The notation // indicates that the specified node is located among the descendants of the root node.

```
//line[2]/cost[@currency="AUD"]
```

For more information, see the W3C Recommendation *XML Path Language (XPath)*.

- **Value**—the test value of type date, number, or text with which to compare the event message data.

The XML Compare Tag Value activities use the Comparison lookup type for a result code. Possible values are "Greater Than," "Less Than," "Equal," or "Null," if the test activity attribute value is null. You can guide your workflow process based on how the event message data compares to the test value. See: To Define Activity Attribute Values: page 5 – 17.

---

## XML Transform Activity

The XML Transform activity lets you apply an XML style sheet to the payload of an event message from the Business Event System. This functionality is currently only available for the standalone version of Oracle Workflow. The resulting document is stored in an item attribute of type event. This activity calls the external Java function named *oracle.apps.fnd.wf.XSLTTransform*.

**Note:** When the Workflow Engine encounters an external Java function activity, it places an entry on the 'Outbound' queue. To continue executing the activity, you must run the Java Function Activity Agent, which calls the appropriate Java function and places the result on the 'Inbound' queue. You must then run a background engine to process the 'Inbound' queue and complete the function activity. See: [Setting Up the Java Function Activity Agent: page 2 – 86](#) and [Setting Up Background Engines: page 2 – 43](#).

## **Activity Attributes**

---

The XML Transform activity has three activity attributes:

- **Event**—choose the item attribute of type event that contains the event message you want to transform.
- **Stylesheet**—a reference to the location of the style sheet that you want to apply. Specify this reference as a URL.
- **New Document**—choose the item attribute of type event where you want to store the new document produced by applying the style sheet. See: [To Define Activity Attribute Values: page 5 – 17](#).

---

## 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 – 13 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 – 17.

---

## **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_app1_id)` in SET\_CTX mode in your Selector/Callback function. See: Standard API for an Item Type Selector or Callback Function: page 7 – 13.

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 – 17.

---

## 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*.

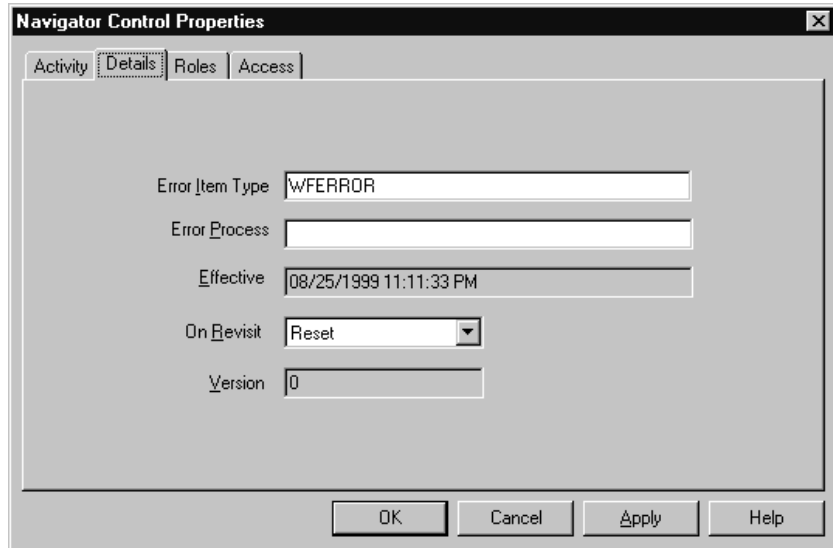
### **Activity Attributes**

---

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 – 17.

## 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, function, or event 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.



The screenshot shows the 'Navigator Control Properties' dialog box with the 'Details' tab selected. The dialog contains the following fields and controls:

- Error Item Type:** A text field containing the value 'WFERROR'.
- Error Process:** An empty text field.
- Effective:** A text field containing the date and time '08/25/1999 11:11:33 PM'.
- On Revisit:** A dropdown menu with 'Reset' selected.
- Version:** A text field containing the value '0'.

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

Oracle Workflow provides a special item type called System: Error, which contains three 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 contains the following item attributes:

- Error Activity ID
- Error Activity Label
- Error Assigned User
- Error Item Type
- Error Item Key
- Error User Key
- Error Message
- Error Name
- Error Notification ID
- Error Result Code
- Error Stack
- Error Monitor URL
- Timeout Value
- Event Name
- Event Details
- Event Message
- Event Key
- Event Data URL
- Event Subscription
- Error Type

These item attributes are referenced by the function, notification, and event activities that make up the error processes called Default Error Process, Retry-only, and Default Event Error Process.



**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's item type using Oracle Workflow Builder.

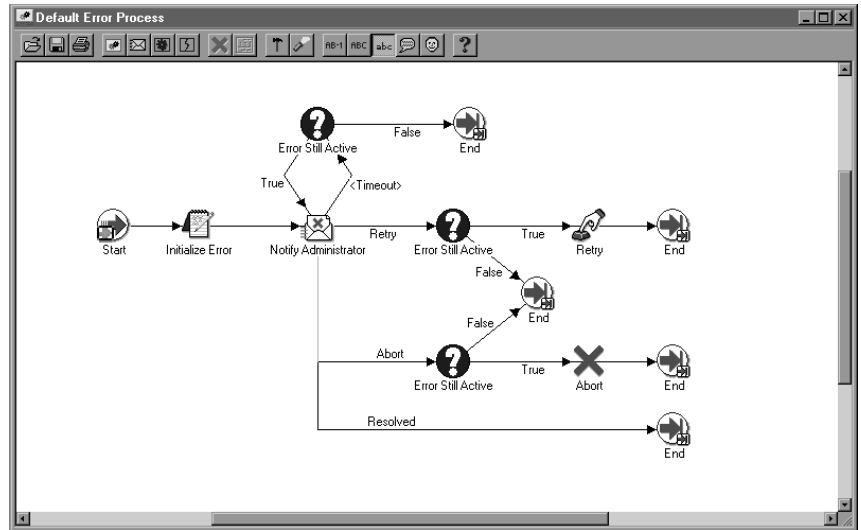
---

## 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 can 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's item type, the Initialize Error activity lets you specify who you want a notification to be sent to in the case of an error in your specific process without modifying the error process.

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 are as follows:

**Subject:** Error in Workflow &ERROR\_ITEM\_TYPE/&ERROR\_ITEM\_KEY  
&ERROR\_MESSAGE

**Body:** An Error occurred in the following Workflow.

```
Item Type = &ERROR_ITEM_TYPE  
Item Key = &ERROR_ITEM_KEY  
User Key =&ERROR_USER_KEY
```

```
Error Name = &ERROR_NAME  
Error Message = &ERROR_MESSAGE  
Error Stack = &ERROR_STACK
```

```
Activity Id = &ERROR_ACTIVITY_ID  
Activity Label = &ERROR_ACTIVITY_LABEL  
Result Code = &ERROR_RESULT_CODE  
Notification Id = &ERROR_NOTIFICATION_ID  
Assigned User = &ERROR_ASSIGNED_USER
```

```
&MONITOR
```

The Notify Administrator notification activity has a dynamic timeout value assigned to it. It checks the item type of the errored process for

an item type attribute whose internal name is `ERROR_TIMEOUT`. `ERROR_TIMEOUT` must be an attribute of type `NUMBER`. The Workflow Engine interprets the value of this attribute as a relative offset from the begin date of the activity, in the unit of `MINUTES` to determine the timeout value of Notify Administrator. If `ERROR_TIMEOUT` contains a null value, a value of zero, or is not defined at all, then Notify Administrator has no timeout.

---

### **'Error Still Active' Function Activity**

---

The Workflow Engine initiates the Error Still Active function activity if the Notify Administrator activity times out or returns `Abort` or `Retry` as a result.

The Error Still Active activity calls a PL/SQL procedure called `WF_STANDARD.CHECKERRORACTIVE`. The purpose of the Error Still Active activity is to determine whether the errored process is still in error before continuing with the error handling. If it is, Error Still Active returns `TRUE` and the Workflow Engine takes the appropriate transition to either send another notification or abort or retry the errored process. If the errored process is no longer in error, this activity returns `False` and the error handling process ends, as modelled in the process diagram.

---

### **'Retry' Function Activity**

---

The Retry function activity executes the PL/SQL procedure `WF_STANDARD.RESETERROR` to clear the activity that was in error and run it again. This procedure calls the `WF_ENGINE.HandleError` API to rerun the activity.

---

### **'Abort' Function Activity**

---

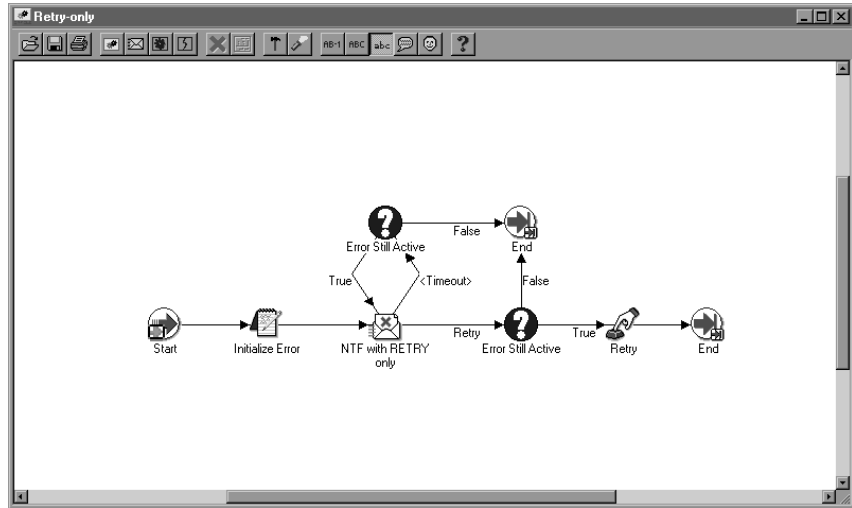
The Abort function activity executes the PL/SQL procedure `WF_STANDARD.ABORTPROCESS`, which in turn calls the `WF_ENGINE.AbortProcess` API to abort the process that encountered the error.

## **See Also**

Workflow Core APIs: page 8 – 101

## 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 are as follows:

**Subject:** Error in Workflow &ERROR\_ITEM\_TYPE/&ERROR\_ITEM\_KEY  
&ERROR\_MESSAGE

**Body:** An Error occurred in the following Workflow.

Item Type = &ERROR\_ITEM\_TYPE  
Item Key = &ERROR\_ITEM\_KEY  
User Key = &ERROR\_USER\_KEY

Error Name = &ERROR\_NAME  
Error Message = &ERROR\_MESSAGE  
Error Stack = &ERROR\_STACK

Activity Id = &ERROR\_ACTIVITY\_ID  
Activity Label = &ERROR\_ACTIVITY\_LABEL  
Result Code = &ERROR\_RESULT\_CODE  
Notification Id = &ERROR\_NOTIFICATION\_ID  
Assigned User = &ERROR\_ASSIGNED\_USER

&MONITOR

The NTF with RETRY Only notification activity has a dynamic timeout value assigned to it. It checks the item type of the process in error for an item attribute that has an internal name called ERROR\_TIMEOUT. ERROR\_TIMEOUT must be an attribute of type NUMBER. The

Workflow Engine interprets the timeout value of this attribute as a relative offset from the begin date of the activity, in the unit of MINUTES. If ERROR\_TIMEOUT contains a null value, a value of zero, or is not defined at all, then NTF with RETRY Only has no timeout.

---

### **'Error Still Active' Function Activity**

---

The Workflow Engine initiates the Error Still Active function activity if the NTF with RETRY Only activity times out.

The Error Still Active activity calls a PL/SQL procedure called *WF\_STANDARD.CHECKERRORACTIVE*. The purpose of the Error Still Active activity is to determine whether the errored process is still in error before continuing with the error handling. If it is, Error Still Active returns TRUE and the Workflow Engine transitions back to the NTF with RETRY Only notification activity to send another notification to the administrator. If the errored process is no longer in error, this activity returns False and the error handling process ends, as modelled in the process diagram.

---

### **'Retry' Function Activity**

---

The Retry function activity executes the PL/SQL procedure *WF\_STANDARD.RESETERROR* to clear the activity that was in error and run it again. This procedure calls the *WF\_ENGINE.HandleError* API to rerun the activity.

## **See Also**

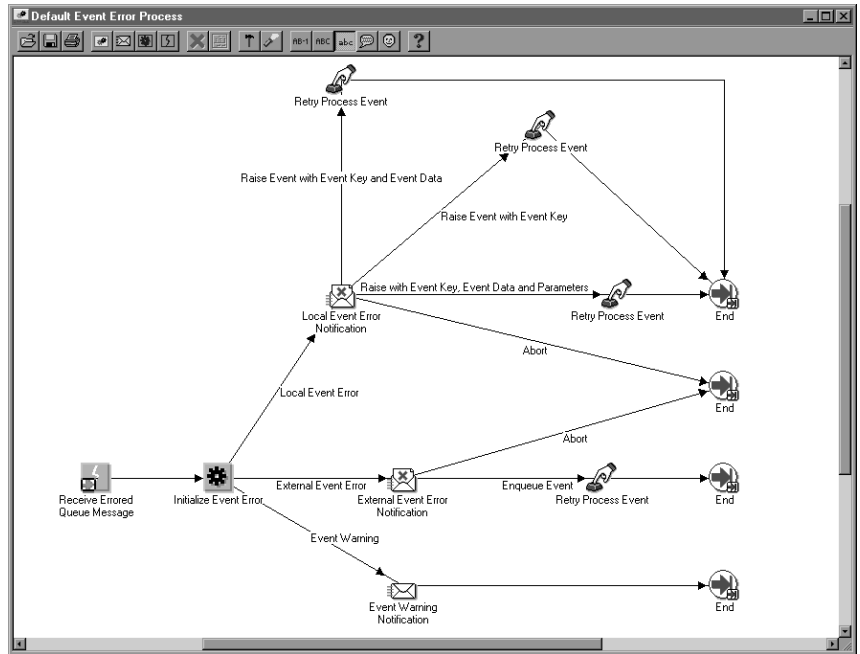
Workflow Core APIs: page 8 – 101

---

## **Default Event Error Process**

DEFAULT\_EVENT\_ERROR is the internal name of the Default Event Error Process for the Business Event System. The purpose of this error handling process is to:

- send an administrator a notification when an error or warning condition occurs during event subscription processing
- provide information to the administrator about the error
- allow the administrator to abort or retry the event subscription processing



### **'Receive Errored Queue Message' Event Activity**

The Receive Errored Queue Message activity receives an event message that has encountered an error or warning condition during subscription processing. For instance, if a subscription rule function returns an ERROR or WARNING status code, or if an unexpected event is received, the Event Manager executes default subscriptions that send the event message to the Default Event Error process. See: *Predefined Workflow Events: page 14 – 2*.

The Receive Errored Queue Message activity stores the event name, event key, and the complete event message in item type attributes.

### **'Initialize Event Error' Function Activity**

The Initialize Error activity calls a PL/SQL procedure named *WF\_STANDARD.INITIALIZEEVENTERROR*. This procedure determines the error type.

- Event Warning—A warning condition occurred, but subscription processing continued. For this error type, the Workflow Engine sends the Event Warning Notification.
- External Event Error—An error occurred to halt subscription processing for an event received from an external source. For this

error type, the Workflow Engine sends the External Event Error Notification.

- Local Event Error—An error occurred to halt subscription processing for an event raised on the local system. For this error type, the Workflow Engine sends the Local Event Error Notification.

### **'Event Warning Notification' Activity**

The Workflow Engine initiates the Event Warning Notification activity when the errored event has an error type of Event Warning. The activity sends the Default Event Warning message to the system administrator to indicate that a warning condition has occurred during subscription processing. This message is an FYI message and does not require a response.

The subject and body of the Default Event Warning message are as follows:

**Subject:** Event WARNING : &EVENT\_NAME / &EVENT\_KEY

**Body:** A Warning occurred in the following Event  
Subscription: &EVENT\_SUBSCRIPTION

Event Error Name: &ERROR\_NAME

Event Error Message: &ERROR\_MESSAGE

Event Error Stack: &ERROR\_STACK

Event Data: &EVENT\_DATA\_URL

Other Event Details: &EVENT\_DETAILS

### **'External Event Error Notification' Activity**

The Workflow Engine initiates the External Event Error Notification activity when the errored event has an error type of External Event Error. The activity sends the Default External Event Error message to the system administrator. This message indicates that an error has occurred during subscription processing for an event received from an external source, and that a response is needed. The response options and their resulting actions are:

- Abort—aborts subscription processing and ends the Default Event Error process. For example, if the event data contained in an event message is corrupted, the system administrator can abort subscription processing on that event message.

- **Enqueue Event**—executes the Retry Process Event activity to enqueue the event message back onto the queue where it was originally received, and ends the Default Event Error process. The event message is enqueued with a priority of -1 so that it will be the first message to be dequeued the next time the listener runs.

The system administrator can attempt to correct the error before re-enqueuing the event. For example, the system administrator can create a subscription to handle an unexpected event and then re-enqueue the event message to trigger the new subscription.

The subject and body of the Default External Event Error message are as follows:

**Subject:** External Event &ERROR\_TYPE : &EVENT\_NAME / &EVENT\_KEY

**Body:** An Error occurred in the following Event Subscription: &EVENT\_SUBSCRIPTION

Event Error Name: &ERROR\_NAME

Event Error Message: &ERROR\_MESSAGE

Event Error Stack: &ERROR\_STACK

Event Data: &EVENT\_DATA\_URL

Other Event Details: &EVENT\_DETAILS

### **'Local Event Error Notification' Activity**

---

The Workflow Engine initiates the Local Event Error Notification activity when the errored event has an error type of Local Event Error. The activity sends the Default Local Event Error message to the system administrator. This message indicates that an error has occurred during subscription processing for an event raised on the local system, and that a response is needed. The response options and their resulting actions are:

- **Abort**—aborts subscription processing and ends the Default Event Error process.
- **Raise Event with Event Key**—executes the Retry Process Event activity to reraise the event with only the event name and event key, and ends the Default Event Error process.
- **Raise Event with Event Key and Event Data**—executes the Retry Process Event activity to reraise the event with only the event

name, event key, and event data, and ends the Default Event Error process.

- **Raise Event with Event Key, Event Data and Parameters**—executes the Retry Process Event activity to reraise the event with the event name, event key, event data, and parameters, and ends the Default Event Error process.

The system administrator can choose the level of information to provide to the Event Manager when reraising the event. For example, if an error exists in the event data that was originally provided, the event can be reraised with only the event name and the event key, forcing the Event Manager to regenerate the event data using the event's Generate function.

The system administrator can also attempt to correct the error before reraising the event.

The subject and body of the Default Local Event Error message are as follows:

**Subject:** Local Event &ERROR\_TYPE : &EVENT\_NAME / &EVENT\_KEY

**Body:** An Error occurred in the following Event

Subscription: &EVENT\_SUBSCRIPTION

Event Error Name: &ERROR\_NAME

Event Error Message: &ERROR\_MESSAGE

Event Error Stack: &ERROR\_STACK

Event Data: &EVENT\_DATA\_URL

Other Event Details: &EVENT\_DETAILS

### **'Retry Process Event' Function Activity**

---

The Retry Process Event activity executes the PL/SQL procedure WF\_STANDARD.RETRYRAISE. Depending on the notification response selected by the system administrator, this procedure either re-enqueues or reraises an errored event. The responses that can initiate the Retry Process Event activity and their resulting actions are:

- **Enqueue Event**—enqueues an errored external event message back onto the queue where it was originally received. The event message is enqueued with a priority of -1 so that it will be the first message to be dequeued the next time the listener runs.
- **Raise Event with Event Key**—reraises a local errored event with only the event name and event key.

- Raise Event with Event Key and Event Data—reraises a local errored event with only the event name, event key, and event data.
- Raise Event with Event Key, Event Data and Parameters—reraises a local errored event with the event name, event key, event data, and parameters.

## See Also

Workflow Core APIs: page 8 – 101

Managing Business Events: page 13 – 2





CHAPTER

# 7

## Defining Procedures and Functions for Oracle Workflow

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

---

## Defining Procedures and Functions for Oracle Workflow

Oracle Workflow lets you integrate your own custom PL/SQL and Java procedures and functions at certain points in your workflow processes and in the Business Event System. To ensure that Oracle Workflow can properly execute your custom code, follow these standard APIs when developing your procedures and functions.

- Standard API for PL/SQL Procedures Called by Function Activities: page 7 – 3
- Standard API for Java Procedures Called by Function Activities: page 7 – 8
- Standard API for an Item Type Selector or Callback Function: page 7 – 13
- Standard APIs for "PL/SQL" and "PL/SQL CLOB" Document: page 7 – 17
- Standard API for an Event Data Generate Function: page 7 – 21
- Standard APIs for a Queue Handler: page 7 – 23
- Standard API for an Event Subscription Rule Function: page 7 – 25

---

## 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>
    if (<condition_ok_to_proceed>) then
        resultout := 'COMPLETE';
    else
        resultout := wf_engine.eng_timedout;
    end if;
    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:

<b>itemtype</b>	The internal name for the item type. Item types are defined in the Oracle Workflow Builder.
<b>itemkey</b>	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.

<b>actid</b>	The ID number of the activity from which this procedure is called.
<b>funcmode</b>	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', 'TIMEOUT', or 'RUN'. Other execution modes may be added in the future.
<b>resultout</b>	<p>If a result type is specified in the Activities properties page for the activity in the Oracle Workflow Builder, this parameter represents the expected result that is returned when the procedure completes. The possible results are:</p> <p><b>COMPLETE:</b><i>&lt;result_code&gt;</i>—activity completes with the indicated result code. The result code must match one of the result codes specified in the result type of the function activity.</p> <p><b>WAITING</b>—activity is pending, waiting on another activity to complete before it completes. An example is the Standard 'AND' activity.</p> <p><b>DEFERRED:</b><i>&lt;date&gt;</i>—activity is deferred to a background engine for execution until a given date. <i>&lt;date&gt;</i> must be of the format:</p> <pre>to_char(&lt;date_string&gt;, wf_engine.date_format)</pre> <p><b>NOTIFIED:</b><i>&lt;notification_id&gt;</i>:<i>&lt;assigned_user&gt;</i>—an external entity is notified that an action must be performed. A notification ID and an assigned user can optionally be returned with this result. Note that the external entity must call <i>CompleteActivity()</i> to inform the Workflow Engine when the action completes.</p> <p><b>ERROR:</b><i>&lt;error_code&gt;</i>—activity encounters an error and returns the indicated error code.</p>

2⇒ This section declares any local arguments that are used within the procedure.

3⇒ The procedure body begins in this section with an IF statement. This section contains one or more executable statements that run if the value of funcmode is 'RUN'. One of the executable statements can

return a result for the procedure. For example, a result can be 'COMPLETE:APPROVED'.

**Note:** The Workflow Engine automatically runs a post-notification function in RUN mode after the Notification System completes execution of the post-notification function in RESPOND mode. The RUN mode executable statements can perform processing such as vote tallying and determine what result to return for the notification activity.

4⇒ This section clears the activity and can contain executable statements that run if the value of `funcmode` is 'CANCEL'. Often, this section contains no executable statements to simply return a null value, but this section also provides you with the chance to 'undo' something if necessary. An activity can have a `funcmode` of 'CANCEL' in the special case where the activity is part of a loop that is being revisited.

The first activity in a loop must always have the Loop Reset 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 – 10 and Loop Counter Activity: page 6 – 7.

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. After the Notification System completes execution of the post-notification function in RESPOND mode, the Workflow Engine then runs the post-notification function again in RUN mode. See: Post-notification functions: page 8 – 13.

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 – 13 and Notification Model: 8 – 192

8⇒ This section is needed only for post-notification functions. Use this section to include execution statements that run if a notification activity times out. You can include logic to test whether the workflow can proceed normally, and if so, to complete the activity so that the workflow can continue to the next activity. For example, if a Voting activity times out before all recipients respond, you can include logic that determines how to interpret the responses based on the current response pool and completes the activity with the appropriate result.

You should also include logic to return a result of `wf_engine.eng_timeout` if the workflow cannot proceed normally. Model any subsequent behavior in your process diagram using a <Timeout> transition to another activity. The Workflow Engine will follow the <Timeout> transition when the result `wf_engine.eng_timeout` is returned.

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 – 108.

---

## Standard API for Java Procedures Called by Function Activities

You can create custom Java classes to be called by external Java function activities in an Oracle Workflow process. This functionality is currently only available for the standalone version of Oracle Workflow. Java procedures that are called by function activities are implemented as classes that extend the `WFFunctionAPI` class. The custom Java classes should follow a standard API format so that they can be properly executed by the Oracle Workflow Java Function Activity Agent.



**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, just as with PL/SQL procedures, you should never commit within the Java procedure of a function activity. The Workflow Engine never issues a commit as it is the responsibility of the calling application to commit.

Many Workflow Engine and Notification APIs have corresponding Java methods that your Java program can call to communicate with Oracle Workflow. The `WFFunctionAPI` and `WFAttribute` classes also contain methods that your Java program can call to access item type and activity attributes. See: Oracle Workflow Java Interface: page 8 – 5, Workflow Function APIs: page 8 – 82, and Workflow Attribute APIs: page 8 – 90.

To invoke a custom Java class from within a workflow process, create an external Java function activity that calls the class. See: To Create a Function Activity: page 4 – 50.

Java function activities are implemented as external procedures. When the Workflow Engine reaches an external Java function activity, the Workflow Engine places a message on the Workflow 'Outbound' queue. The Java Function Activity Agent monitors this queue and calls the class indicated in the Function Name property for the function activity. When the Java procedure is complete, the Java Function Activity Agent enqueues the results onto the 'Inbound' queue. See: Setting Up the Java Function Activity Agent: page 2 – 86.

**Note:** These 'Outbound' and 'Inbound' queues are separate from the queues used for the Business Event System. In a future release, this function processing will be implemented within the Business Event System using a specialized queue handler to handle dequeue and enqueue operations. See: Workflow Queue APIs: page 8 – 162.



After a Java procedure completes, you must run a background engine to process the 'Inbound' queue and complete the function activity. Otherwise, the activity will remain in the DEFERRED status. See: [Setting Up Background Engines: page 2 – 43](#).

You must include the JAR files containing your custom classes in your CLASSPATH to make the classes accessible to the Java Function Activity Agent. The custom class files should reside on the same platform where the Java Function Activity Agent is run. The Java Function Activity Agent does not need to reside on the same tier as the database, however.

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

```
1⇒ package oracle.apps.fnd.wf;
2⇒ import java.io.*;
   import java.sql.*;
   import java.math.BigDecimal;
   import oracle.sql.*;
   import oracle.jdbc.driver.*;

   import oracle.apps.fnd.common.*;
   import oracle.apps.fnd.wf.engine.*;
   import oracle.apps.fnd.wf.*;

3⇒ public class className extends WFFunctionAPI {
4⇒     public boolean execute(WFContext pWCtx){
5⇒         ErrorStack es = pWCtx.getWFErrorStack();
           try
           {
6⇒             WFAttribute lAAttr = new WFAttribute();
             WFAttribute lIAttr = new WFAttribute();

7⇒             loadActivityAttributes(pWCtx, itemType, itemKey, actID);
             loadItemAttributes(pWCtx);

8⇒             lAAttr = getActivityAttr("AATTR");
             lIAttr = getItemAttr("IATTR");

9⇒             <your executable statements>
```

```

10⇒         lIAttr.value((Object) "NEWVALUE");
            setItemAttrValue(pWCtx, lIAttr);

11⇒     }

            catch (Exception e)
            {
                es.addMessage("WF", "WF_FN_ERROR");
                es.addToken("MODULE", this.getClass().getName());
                es.addToken("ITEMTYPE", itemType);
                es.addToken("ITEMKEY", itemKey);
                es.addToken("ACTID", actID.toString());
                es.addToken("FUNCMODE", funcMode);
                es.addToken("ERRMESSAGE", e.getMessage());
                return false;
            }
12⇒     return true;
        }
    }

```

1⇒ By default, Java classes supplied by Oracle Workflow will be in the `oracle.apps.fnd.wf` package. This section is optional.

2⇒ For correct operation, you must include the listed packages.

3⇒ The custom Java class must extend the `WFFunctionAPI` class. This class provides class variables and methods that are essential to the operation of your function activity.

The parameters that are normally supplied to a PL/SQL function activity are available to the custom class as class variables. They are initialized prior to the call of the `boolean execute()` method. The `resultOut` and the `errorStack` are then passed back to the Oracle Workflow Engine.

The status of the completed activity will be set to `COMPLETE` unless a value is present in the `errorStack` variable. If there is a value in this variable, then the activity status will be set to `ERROR`. The contents of the `errorStack` variable can be set by using the `ErrorStack` class within the `WFContext` class. Refer also to sections 5 and 11 of this API for catching exceptions.

The predefined class variables include:

<b>itemType</b>	The internal name for the item type. Item types are defined in the Oracle Workflow Builder.
-----------------	---

<b>itemKey</b>	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.
<b>ActID</b>	The ID number of the activity from which this procedure is called.
<b>funcMode</b>	The execution mode of the activity. Currently the only supported mode for external Java function activities is the 'RUN' mode.
<b>resultOut</b>	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.

**Note:** Unlike the resultout for a PL/SQL procedure called by a function activity, the resultOut for a Java procedure does not include a status code. In the Java API, only the result type value is required. The status of the activity will be set automatically by the Workflow Engine depending on whether there is a value in the errorStack variable.

4⇒ The custom Java class must implement the `boolean execute()` method. This will be the main entry point for your Java class. On successful completion, this method should return `true`.

5⇒ It is important to catch exceptions with your custom Java class and pass them back to the engine via the `ErrorStack` class. Refer also to section 11 of this API for catching exceptions.

6⇒ To access item and activity attributes, a `WFAttribute` class is provided.

7⇒ The values of the item attributes are not automatically available to the Java class. They are loaded on demand. The values can be loaded explicitly with the `void loadItemAttributes(WFContext)` or the `void loadActivityAttributes(WFContext)` methods. The values are also loaded implicitly when you call the `WFAttribute getItemAttr(String)` or `WFAttribute getActivityAttr(String)` methods. This section is optional.

8⇒ The actual values of the item and activity attributes are accessed via the `WFAttribute getItemAttr(String)` and `WFAttribute getActivityAttr(String)` methods. If you have not explicitly loaded the values of the attributes, they will be automatically loaded at this point.

9⇒ This section contains your own executable statements. Usually, you add these executable statements after retrieving the required item and activity attribute details (section 8) and before setting item attribute values (section 10).

10⇒ Setting the value of an item attribute with the `void setItemAttrValue(WFContext, WFAttribute)` method writes the value of your local `WFAttribute` to the database. You need to set the values of the `WFAttribute` class with the `WFAttribute.value(Object)` method.

11⇒ It is important to catch exceptions within your custom Java class and pass them back to the engine via the `ErrorStack` class.

An unsuccessful execution of the external Java function activity will return `false`.

Note that any message in the `WFContext.wErrorStack` class variable will be passed back to the Workflow Engine and will cause the activity to be assigned a completion status of `ERROR`.

12⇒ A successfully executed external Java function activity will return `true`.

---

## Standard API for an Item Type Selector or Callback Function

For any given item type, you can define a single function that operates as both a selector and a callback function. A selector function is a PL/SQL procedure that automatically identifies the specific process definition to execute when a workflow is initiated for a particular item type but no process name is provided. Oracle Workflow also supports using a callback function to reset or test item type context information. You can define one PL/SQL procedure that includes both selector and callback functionality by following a standard API.

Oracle Workflow can call the selector/callback function with the following commands:

- **RUN**—to select the appropriate process to start when either of the following two conditions occur:
  - A process is not explicitly passed to `WF_ENGINE.CreateProcess`.
  - A process is implicitly started by `WF_ENGINE.CompleteActivity` with no prior call to `WF_ENGINE.CreateProcess`.
- **SET\_CTX**—to establish any context information for an item type and item key combination that a function activity in the item type needs in order to execute. The Workflow Engine calls the selector/callback function with this command each time it encounters a new item type and item key combination, to ensure that the correct context information is always set.
- **TEST\_CTX**—to determine if the current item type context information is correct before executing a function. For example, the selector/callback function in `TEST_CTX` mode lets you check if a form can be launched with the current context information just before the Notification Details web page launches a reference form. If the context is incorrect, the form cannot be launched and a message is displayed to that effect. See: [To View the Details of a Notification: page 10 – 19](#).

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:

<b>itemtype</b>	The internal name for the item type. Item types are defined in the Oracle Workflow Builder.
<b>itemkey</b>	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.
<b>actid</b>	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.

<b>command</b>	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.
<b>resultout</b>	<p>A result may be returned depending on the command that is used to call the selector/callback function.</p> <p>If the function is called with 'RUN', the name of the process to run must be returned through the 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.</p>

2⇒ This section declares any local arguments that are used within the procedure.

3⇒ The procedure body begins in this section with an IF statement. This section contains one or more executable statements that make up your selector function. It executes if the value of `command` is 'RUN'. One of the executable statements should return a result for the procedure that reflects the process to run. For example, a result can be 'REQUISITION\_APPROVAL', which is the name of a process activity.

4⇒ This section contains one or more executable statements that set item type context information if the value of `command` is 'SET\_CTX'. The Workflow Engine calls the selector/callback function with this command each time it encounters a new item type and item key combination, before executing any function activities for that combination. This command is useful when you need to set item type context information in a database session before the activities in that session can execute as intended. For example, you might need to set up the responsibility and organization context for function activities that are sensitive to multi-organization data.

5⇒ This section contains one or more executable statements that validate item type context information if the value of `command` is 'TEST\_CTX'. The Workflow Engine calls the selector/callback function with this command to validate that the current database session context is acceptable before the Workflow Engine executes an activity. For example, this callback functionality executes 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 – 108.



---

## Standard APIs for "PL/SQL" and "PL/SQL CLOB" Documents

You can integrate a document into a workflow process by defining an attribute of type document for an item type, message, or activity. Oracle Workflow supports document types called "PL/SQL" documents and "PL/SQL CLOB" documents. A PL/SQL document represents data as a character string, while a PL/SQL CLOB document represents data as a character large object (CLOB).

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 procedures that generate PL/SQL and PL/SQL CLOB documents must follow standard API formats.

---

### "PL/SQL" Documents

The PL/SQL procedure that generates a PL/SQL document must have the following standard API:

```
procedure <procedure name> (document_id in varchar2,  
                             display_type in varchar2,  
                             document in out varchar2,  
                             document_type in out varchar2)
```

The arguments for the procedure are as follows:

**document\_id** A string that uniquely identifies a document. This is the same string as the value that you specify in the default value field of the Attribute property page for a "PL/SQL" document (plsql:<procedure>/<document\_identifier>). <procedure> should be replaced with the PL/SQL package and procedure name in the form of package.procedure. The phrase <document\_identifier> should be replaced with the PL/SQL argument string that you want to pass directly to the procedure. The argument string should identify the document. For example: plsql:po\_wf.show\_req/2034. If you wish to generate the PL/SQL argument string value dynamically, create another item attribute, and

reference that item attribute as `"&ITEM_ATTRIBUTE"` in place of the PL/SQL argument string. Then before any activity that references this other item attribute gets executed, call the `WF_ENGINE.SetItemAttribute` API to dynamically set the PL/SQL argument string value. For example:

```
plssql:po_wf.show_req/&POREQ_NUMBER.
```

**display\_type**

One of three values that represents the content type used for the notification presentation, also referred to as the requested type:

**text/plain**—the document is embedded inside a plain text representation of the notification as viewed from an e-mail message. The entire e-mail message must be less than or equal to 32K, so depending on how large your e-mail template is, some of the plain text document that the procedure generates may get truncated. See: *Modifying Your Message Templates: page 2 – 69.*

**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.

## See Also

To Define a Document Attribute: page 4 – 14

---

## "PL/SQL CLOB" Documents

The PL/SQL procedure that generates a PL/SQL CLOB document must have the following standard API:

```
procedure <procedure name> (document_id in varchar2,  
                             display_type in varchar2,  
                             document in out clob,  
                             document_type in out varchar2)
```

The arguments for the procedure are as follows:

**document\_id** A string that uniquely identifies a document. This is the same string as the value that you specify in the default value field of the Attribute property page for a "PL/SQL CLOB" document (plsqclob:<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:  
plsqclob:po\_wf.show\_req\_clob/2036. If you wish to generate the PL/SQL argument string value dynamically, create another item attribute, and reference that item attribute as "&ITEM\_ATTRIBUTE" in place of the PL/SQL argument string. Then before any activity that references this other item attribute gets executed, call the WF\_ENGINE.SetItemAttribute API to dynamically set the PL/SQL argument string value. For example:  
plsqclob:po\_wf.show\_req\_clob/&POREQ\_NUMBER.

<b>display_type</b>	<p>One of three values that represents the content type used for the notification presentation, also referred to as the requested type:</p> <p><b>text/plain</b>—the document is embedded inside a plain text representation of the notification.</p> <p><b>text/html</b>—the document is embedded inside an HTML representation of the notification as viewed from the Notification Web page. The procedure must generate a HTML representation of the document, but should not include top level HTML tags like &lt;HTML&gt; or &lt;BODY&gt; 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.</p> <p>' '</p>
<b>document</b>	The outbound text buffer where the document text is returned.
<b>document_type</b>	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.

**Note:** You can call `WF_NOTIFICATION.WriteToClob()` to help build the CLOB by appending a string of character data to it. See: `WriteToClob`: page 8 – 234.

## See Also

To Define a Document Attribute: page 4 – 14

---

## Standard API for an Event Data Generate Function

When you define an event in the Business Event System, you can assign the event a Generate function that can produce the complete event data from the event name, event key, and an optional parameter list. The event data gives additional details to describe what occurred and can be structured as an XML document. You should specify a Generate function if the application that raises the event will not produce the event data itself.

When an event is raised locally, the Event Manager checks each subscription before executing it to determine whether the subscription requires the event data. If the event data is required but is not already provided, the Event Manager calls the Generate function for the event to produce the event data. The Generate function returns the event data in character large object (CLOB) format.

**Note:** If the event data is required but no Generate function is defined for the event, Oracle Workflow creates a default set of event data using the event name and event key.

**Note:** If the Generate function is costly, and you want to return control to the calling application more quickly after raising the event, you can defer all the subscriptions that require the complete event data. Then the Event Manager will not run the Generate function until those subscriptions are executed at a later time. See: *Deferred Subscription Processing*: page 13 – 41.

The PL/SQL function that generates the event data must have the following standard API:

```
function <function_name> (p_event_name in varchar2,  
                          p_event_key in varchar2  
                          p_parameter_list in wf_parameter_list_t  
                          default null)  
return clob;
```

The arguments for the function are as follows:

- |                         |   |
|-------------------------|---|
| <b>p_event_name</b>     | The internal name of the event.   |
| <b>p_event_key</b>      | A string generated when the event occurs within a program or application. The event key uniquely identifies a specific instance of the event. |
| <b>p_parameter_list</b> | An optional list of additional parameter name and value pairs for the event.  |

## See Also

To Define an Event: page 13 – 5

To Define an Event Subscription: page 13 – 45

Parameter List Structure: page 8 – 241

---

## Standard APIs for a Queue Handler

When you define an agent in the Business Event System, you must assign the agent a queue handler. The queue handler is a package that translates between the standard Workflow event message format defined by the `WF_EVENT_T` datatype and the message format required by the queue associated with the agent.

Oracle Workflow provides two standard queue handlers for queues that use the `WF_EVENT_T` format, `WF_EVENT_QH` for normal processing and `WF_ERROR_QH` for error queues. Oracle Workflow also provides a standard queue handler named `WF_EVENT_OMB_QH` which you can set up and use if you implement Oracle Message Broker in Oracle8i to propagate event messages between systems.

Additionally, you can create your own custom queue handlers for queues that use other formats. If you create a custom queue handler, you must provide the following standard APIs in your package:

- `Enqueue()`
- `Dequeue()`

### See Also

Event Message Structure: page 8 – 242

Agents: page 13 – 22

Setting Up the `WF_EVENT_OMB_QH` Queue Handler: page 2 – 100

---

## Enqueue

The `Enqueue` procedure in a queue handler package must enqueue an event message onto a queue associated with an outbound agent. You can optionally specify an override agent where you want to enqueue the event message. Otherwise, the event message is enqueued on the From Agent specified within the message. The `Enqueue` procedure transforms the event message's header information if necessary to enqueue the message in the format required by the queue.

When an event message is being sent, the generic `WF_EVENT.Enqueue` procedure determines which queue handler is associated with the specified outbound agent and calls the `Enqueue` procedure in that queue handler to enqueue the message.

The PL/SQL Enqueue procedure must have the following standard API:

```
procedure enqueue (p_event in WF_EVENT_T,  
                  p_out_agent_override in WF_AGENT_T);
```

The arguments for the procedure are as follows:

<b>p_event</b>	The event message.
<b>p_out_agent_override</b>	The outbound agent on whose queue the event message should be enqueued.

---

## Dequeue

The Dequeue procedure in a queue handler package must dequeue an event message from the queue associated with the specified inbound agent, selecting the message to dequeue by the message priority. The procedure transforms the event message's header information if necessary and returns the event message in the standard WF\_EVENT\_T structure. Additionally, the Dequeue procedure can set the date and time when the message is dequeued into the RECEIVE\_DATE attribute of the event message.

When an event message is being received, the WF\_EVENT.Listen procedure determines which queue handler to use with the specified inbound agent and calls the Dequeue procedure in that queue handler to dequeue the message.

The PL/SQL Dequeue procedure must have the following standard API:

```
procedure dequeue (p_agent_guid in raw,  
                  p_event out WF_EVENT_T);
```

The arguments for the procedure are as follows:

<b>p_agent_guid</b>	The globally unique identifier of the inbound agent from whose queue the event message should be dequeued.
<b>p_event</b>	The event message.



---

## Standard API for an Event Subscription Rule Function

When you define an event subscription, you can choose to run a PL/SQL function called a rule function on the event message. Oracle Workflow provides a standard default rule function to perform basic subscription processing. This function is executed by default if no other rule function is specified for the subscription. The default rule function includes the following actions:

- Sending the event message to a workflow process, if specified in the subscription definition
- Sending the event message to an agent, if specified in the subscription definition

See: `Default_Rule`: page 8 – 281.

Oracle Workflow also provides some standard rule functions that you can use for testing and debugging or other purposes. See: `Event Subscription Rule APIs`: page 8 – 279.

You can extend your subscription processing by creating custom rule functions. Custom rule functions must be defined according to a standard API.

A rule function may read from or write to the event message or perform any other database action. However, you should never commit within a rule function. The Event Manager never issues a commit as it is the responsibility of the calling application to commit. Additionally, the rule function must not change the connection context in any way, including security and NLS settings.

**Note:** If your rule function writes to the event message, any subsequent subscriptions executed on the event will access the changed message.

If the subscription processing that you want to perform for an event includes several successive steps, you may find it advantageous to define multiple subscriptions to the event with simple rule functions that you can reuse, rather than creating complex specialized rule functions that cannot be reused. You can enter phase values for the subscriptions to specify the order in which they should be executed.

By default, the Event Manager uses the event key as the correlation ID for the event message when no other correlation ID is specified. If you want to specify a different correlation ID, you can use `WF_EVENT_FUNCTIONS_PKG.AddCorrelation` to add a correlation ID to the event message, either by calling this function within your custom rule function or by defining another subscription that uses

`WF_EVENT_FUNCTIONS_PKG.AddCorrelation` as its rule function. See: `AddCorrelation`: page 8 – 294.

If you want to send the event message to a workflow process or to an agent after running custom code on the message, you must either include the send processing in your rule function, or define a separate subscription that uses the default rule function to perform the send processing.

- Call `WF_ENGINE.Event()` to send the event message to a workflow process.
- Call `WF_EVENT.Send()` to send the event message to an agent.
- Call `WF_RULE.Default_Rule()` to include the default subscription processing that can send the event message both to a workflow process and to an agent.

**Note:** When you define a subscription in the Event Manager, you can define the workflow item type, workflow process name, out agent, to agent, priority, and parameters for your send processing, as well as defining the rule function. Any rule function can access these send attributes, but if you do not use the default rule function, you must explicitly include the send processing in your custom rule function if you want to send the event from the same subscription.

The standard API for a rule 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⇒ function <function_name> (p_subscription_guid in raw,
                               p_event in out WF_EVENT_T) return varchar2 is
2⇒   <local declarations>
3⇒   begin
         <your executable statements>
4⇒   <optional code for WARNING>
         WF_CORE.CONTEXT('<package name>', '<function name>',
                           p_event.getEventName( ), p_subscription_guid);
         WF_EVENT.setErrorInfo(p_event, 'WARNING');
         return 'WARNING';
5⇒   return 'SUCCESS';
6⇒   exception
         when others then
         WF_CORE.CONTEXT('<package name>', '<function name>',
                           p_event.getEventName( ), p_subscription_guid);
         WF_EVENT.setErrorInfo(p_event, 'ERROR');
         return 'ERROR';
```

7⇒ end;

1⇒ When the Event Manager calls the rule function, it passes two parameters to the function and expects a return code when the function completes. The parameters are defined here:

**p\_subscription\_**     The globally unique identifier for the subscription.  
**guid**

**p\_event**             The event message.

The function must return one of the following status codes:

- **SUCCESS**—The rule function completed successfully.
- **WARNING**—A warning condition occurred. The rule function reports a warning message using the Workflow Core error APIs and sets the warning information into the event message. The Event Manager places a copy of the event message on the `WF_ERROR` queue, but subscription processing continues.
- **ERROR**—An error occurred. The rule function reports an error message using the Workflow Core error APIs and sets the error information into the event message. The Event Manager halts subscription processing for this event, rolls back any subscriptions already executed for the event, and places the event message on the `WF_ERROR` queue.

2⇒ This section declares any local arguments that are used within the function.

3⇒ The procedure body begins in this section with one or more executable statements that make up your rule function.

4⇒ This optional section calls `WF_CORE.CONTEXT()` if a warning condition occurs, so that you can include context information in the error stack to help you locate the source of an error. It also sets the warning information into the event message and returns the status code 'WARNING'. See: `CONTEXT`: page 8 – 108.

5⇒ This section returns the status code 'SUCCESS' when the rule function's normal processing completes successfully.

6⇒ This section calls `WF_CORE.CONTEXT()` if an exception occurs, so that you can include context information in the error stack to help you locate the source of an error. It also sets the error information into the event message and returns the status code 'ERROR'. See: `CONTEXT`: page 8 – 108.

**Note:** If you raise an exception in the rule function, the Event Manager rolls back all subscription processing for the event and raises the error to the calling application. In this case the event message is not placed on the WF\_ERROR queue.

## See Also

To Define an Event Subscription: page 13 – 45

Event Message Structure: page 8 – 242

Workflow Core APIs: page 8 – 101

Event Subscription Rule APIs: page 8 – 279

Event(): page 8 – 75

Send(): page 8 – 265

Default\_Rule(): page 8 – 281

SetErrorInfo(): page 8 – 273

CHAPTER

# 8



## Oracle Workflow APIs

**T**his chapter describes the APIs for Oracle Workflow. The APIs consist of views and PL/SQL and Java functions and procedures that you can use to access the Workflow Engine, the Notification System, the Business Event System, and workflow data.

---

## Oracle Workflow Procedures and Functions

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

- WF\_ENGINE: page 8 – 19
- WFFunctionAPI: page 8 – 82
- WFAttribute: page 8 – 90
- WF\_CORE: page 8 – 101
- WF\_PURGE: page 8 – 111
- WF\_DIRECTORY: page 8 – 121
- WF\_LDAP: page 8 – 144
- WF\_PREF: page 8 – 148
- WF\_MONITOR: page 8 – 149
- Oracle Workflow Views: page 8 – 157
- WF\_QUEUE: page 8 – 162
- FND\_DOCUMENT\_MANAGEMENT: page 8 – 185
- WF\_NOTIFICATIONS: page 8 – 197
- WF\_EVENT: page 8 – 260
- WF\_RULE: page 8 – 279
- WF\_EVENT\_FUNCTIONS\_PKG: page 8 – 290
- WF\_EVENTS\_PKG: page 8 – 300
- WF\_EVENT\_GROUPS\_PKG: page 8 – 300
- WF\_SYSTEMS\_PKG: page 8 – 300
- WF\_AGENTS\_PKG: page 8 – 300
- WF\_EVENT\_SUBSCRIPTIONS\_PKG: page 8 – 300

---

## 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 the Oracle database server, if the Workflow server goes down for any reason, the Oracle database server 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. The API calls that update activity states are:

- CreateProcess: page 8 – 21
- StartProcess: page 8 – 28
- CompleteActivity: page 8 – 69
- CompleteActivityInternalName: page 8 – 72
- AssignActivity: page 8 – 74
- HandleError: page 8 – 77
- SuspendProcess: page 8 – 32
- ResumeProcess: page 8 – 34
- AbortProcess: page 8 – 36

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.

**Note:** The Oracle database server release 8i and higher offers autonomous transactions. By embedding the pragma `AUTONOMOUS_TRANSACTION` in your procedure, you can perform commits and rollbacks independently of the main transaction. Oracle treats this as a separate session; as such, you will not have access to any database changes that were made in the main session but are not yet committed.

Consequently, you are restricted from updating workflow-specific data in an autonomous transaction; for instance, you cannot set item attributes. You cannot access this data because the item itself has not yet been committed, and because you may have lock contentions with the main session.

Oracle Workflow will not support autonomous commits in any procedure it calls directly. If you need to perform commits, then embed your SQL in a subprocedure and declare it as an autonomous block. This subprocedure must be capable of being rerun. Additionally, note that Oracle Workflow handles errors by rolling back the entire procedure and setting its status to ERROR. Database updates performed by autonomous commits



cannot be rolled back, so you will need to write your own compensatory logic for error handling. For more information, see: *Autonomous Transactions, Oracle Database Concepts*.

---

## 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. See: *Workflow Engine APIs: page 8 – 19* and *Notification APIs: page 8 – 197*.

Additionally, Java functions can be incorporated within Workflow processes as external Java function activities. This functionality is currently only available for the standalone version of Oracle Workflow. The custom Java classes for these activities are implemented as classes that extend the WFFunctionAPI class. The custom classes must follow a standard API format so that they can be properly executed by the Oracle Workflow Java Function Activity Agent. See: *Standard API for Java Procedures Called by Function Activities: page 7 – 8* and *Function Activity: page 4 – 44*.

The WFFunctionAPI class and the WFAttribute class also contain methods that can be called to communicate with Oracle Workflow. These classes are defined in the Java package 'oracle.apps.fnd.wf'. See: *Workflow Function APIs: page 8 – 82* and *Workflow Attribute APIs: page 8 – 90*.

Java programs that integrate with Oracle Workflow should include the following import statements to provide access to classes required by Oracle Workflow:

```
import java.io.*;
import java.sql.*;
import java.math.BigDecimal;
```

```

import oracle.sql.*;
import oracle.jdbc.driver.*;

import oracle.apps.fnd.common.*;
import oracle.apps.fnd.wf.engine.*;
import oracle.apps.fnd.wf.*;

```

## Oracle Workflow Context

---

Each Oracle Workflow Java method that accesses the database requires an input of a WFContext object. The WFContext object consists of database connectivity information which you instantiate and resource context information that the WFContext class instantiates. To call one of these Workflow Java APIs in your Java program, you must first instantiate a database variable of class WFDB with your database username, password and alias. You can also optionally supply a JDBC string. Then you must instantiate the WFContext object with the database variable. You can retrieve the system property CHARSET to specify the character set for the database session. The following code excerpt shows an example of how to instantiate these objects.

```

WFDB myDB;
WFContext ctx;

myDB = new WFDB(m_user, m_pwd, m_jdbcStr, m_conStr);
m_charSet = System.getProperty("CHARSET");
if (m_charSet == null) { // cannot be null
    m_charSet = "UTF8";
}

try {
    ctx = new WFContext(myDB, m_charSet);
    // m_charSet is 'UTF8' by default

    if (ctx.getDB().getConnection() == null) {
        // connection failed
        return;
    }

    // We now have a connection to the database.
}

catch (Exception e) {

```

```
// exit Message for this exception
}
```

If you have already established a JDBC connection, you can simply set that connection into the WFContext object, as shown in the following example:

```
WFContext ctx;

m_charSet = System.getProperty("CHARSET");
if (m_charSet == null) { // cannot be null
    m_charSet = "UTF8";
}

ctx = new WFContext(m_charSet);
// m_charSet is 'UTF8' by default

ctx.setJDBCConnection(m_conn);
// m_conn is a pre-established JDBC connection
```

## Sample Java Program

---

Oracle Workflow provides an example Java program that illustrates how to call most of the Workflow Engine and Notification Java APIs. The Java program is named WFTest. It calls the various Java APIs to launch the WFDEMO process, set and get attributes and suspend, resume and abort the process, as well as the APIs to send a notification, set and get notification attributes, and delegate and transfer the notification. Before running the WFTest Java program, make sure you define CLASSPATH and LD\_LIBRARY\_PATH for the Oracle JDBC implementation and a supported version of Oracle. For example, on UNIX, use the following commands:

```
setenv CLASSPATH
<Workflow_JAR_file_directory>/wfapi.jar:${ORACLE_HOME}/jdbc/
lib/classes111.zip
```

```
setenv LD_LIBRARY_PATH ${ORACLE_HOME}/lib:${LD_LIBRARY_PATH}
```

**Note:** If you are using the standalone version of Oracle Workflow with Oracle9i, the Workflow JAR files are located in the <ORACLE\_HOME>/jlib directory. If you are using the version of Oracle Workflow embedded in Oracle Applications, the Workflow JAR files are located in the

<ORACLE\_HOME>/wf/java/oracle/apps/fnd/wf/jar/  
directory.

To initiate the WFTTest program, run Java against `oracle.apps.fnd.wf.WFTTest`. For example, on UNIX, enter the following statement on the command line:

```
$java oracle.apps.fnd.wf.WFTest
```

The source file for this program is also included in your Oracle Workflow installation so that you can view the sample code. The source file is named `WFTTest.java` and is located in the

<ORACLE\_HOME>/wf/java/oracle/apps/fnd/wf/ directory.

---

## Additional Workflow Engine Features

In addition to managing a process, the Workflow Engine also supports the following features:

- Completion Processing: page 8 – 8
- Deferred Processing: page 8 – 9
- Error Processing: page 8 – 10
- Looping: page 8 – 10
- Version/Effective Date: page 8 – 11
- Item Type Attributes: page 8 – 12
- Post-notification functions: page 8 – 13
- Synchronous, Asynchronous, and Forced Synchronous Processes: page 8 – 14
- Business Events: page 8 – 17

### 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.

## **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 Oracle 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's 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.

## **See Also**

Activity Cost: page 4 – 47

Deferring Activities: page C – 7

## 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 processing 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 – 26.

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 – 59.

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.

If you set On Revisit to Reset for the pivot activity of a loop that includes an FYI notification activity, the Workflow Engine cancels the previous notification before reexecuting the loop and sending a new notification to the current performer of the notification activity.

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.

If you set On Revisit to Loop for the pivot activity of a loop that includes an FYI notification activity, previous notifications remain open when the Workflow Engine reexecutes the loop and sends a new notification to the current performer of the notification 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.

When you define item type attributes at runtime, you can add either individual attributes or arrays containing several attributes of the same type, using the appropriate Workflow Engine APIs. Similarly, you can



set the values of existing attributes either individually or in arrays containing several attributes of the same type.

Use the array APIs whenever you need to add or set the values of large numbers of item type attributes at once. These APIs improve performance by using the bulk binding feature in Oracle8i and higher to reduce the number of database operations. See:

AddItemAttributeArray: page 8 – 46 and SetItemAttributeArray: page 8 – 53.

**Note:** These array APIs handle arrays that are composed of multiple item type attributes grouped together by type. Oracle Workflow does not support individual item type attributes that consist of arrays themselves.

### 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 back-end 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 – 3.

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;

or

WF\_ENGINE.context\_text = *responder*, if the post-notification function gets called in RESPOND mode.

**Note:** 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 Notification System completes execution of the post-notification function in RESPOND mode, the Workflow Engine automatically runs the post-notification function again in RUN mode. In this mode, the post-notification function can perform additional processing such as vote tallying.

If a notification activity times out, the Workflow Engine runs the post-notification function for the activity in TIMEOUT mode. For a Voting activity, the TIMEOUT mode logic should identify how to tally the votes received up until the timeout.

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 – 192

### **Synchronous, Asynchronous, and Forced Synchronous Processes**

---

A workflow process can be either synchronous or asynchronous. A synchronous process is a process that can be executed without

interruption from start to finish. The Workflow Engine executes a process synchronously when the process includes activities that can be completed immediately, such as function activities that are not deferred to the background engine. The Workflow Engine does not return control to the calling application that initiated the workflow until it completes the process. With a synchronous process, you can immediately check for process results that were written to item attributes or directly to the database. However, the user must wait for the process to complete.

An asynchronous process is a process that the Workflow Engine cannot complete immediately because it contains activities that interrupt the flow. Examples of activities that force an asynchronous process include deferred activities, notifications with responses, blocking activities, and wait activities. Rather than waiting indefinitely when it encounters one of these activities, the Workflow Engine sets the audit tables appropriately and returns control to the calling application. The workflow process is left in an unfinished state until it is started again. The process can be restarted by the Notification System, such as when a user responds to a notification; by the background engine, such as when a deferred activity is executed; or by the Business Event System, such as when an event message is dequeued from an inbound queue and sent to the workflow process. With an asynchronous process, the user does not have to wait for the process to complete to continue using the application. However, the results of the process are not available until the process is completed at a later time.

In addition to regular synchronous and asynchronous processes, 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. The process results are available immediately upon completion. However, no audit trail is recorded.

There may be cases when your application requires a forced synchronous process to generate a specific result quickly when 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 to 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 a 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 16 – 16.

## See Also

Synchronous, Asynchronous, and Forced Synchronous Workflows:  
page C – 2

### **Business Events**

---

Events from the Business Event System are represented within workflow processes as event activities. An event activity can either raise, send, or receive a business event.

A Raise event activity raises an event to the Event Manager, triggering any subscriptions to that event. The Workflow Engine calls the WF\_EVENT.Raise API to raise the event. See: Raise: page 8 – 261.

A Send event activity sends an event directly to a Business Event System agent without raising the event to the Event Manager. The Workflow Engine calls the WF\_EVENT.Send API to send the event. See: Send: page 8 – 265.

A Receive event activity receives an event from the Event Manager into a workflow process, which then continues the thread of execution from that activity. The Workflow Engine can receive an event into an activity in an existing process instance that is waiting for the event, using the correlation ID in the event message to match the event with the process to which it belongs. The Workflow Engine can also receive an event into a Receive event activity that is marked as a Start activity to launch a new workflow process. The WF\_ENGINE.Event API is used to receive an event into a workflow process. See: Event: page 8 – 75.

**Note:** If the event received by a Receive event activity was originally raised by a Raise event activity in another workflow process, the item type and item key for that process are included in the parameter list within the event message. In this case, the Workflow Engine automatically sets the specified process as the parent for the process that receives the event, overriding any existing parent setting. See: SetItemParent: page 8 – 79.

## See Also

Managing Business Events: page 13 – 2

Event Activities: page 4 – 54

---

## 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 – 21—PL/SQL and Java
- SetItemUserKey: page 8 – 23—PL/SQL
- GetItemUserKey: page 8 – 24—PL/SQL
- GetActivityLabel: page 8 – 25—PL/SQL
- SetItemOwner: page 8 – 26—PL/SQL and Java
- StartProcess: page 8 – 28—PL/SQL and Java
- LaunchProcess: page 8 – 30—PL/SQL and Java
- SuspendProcess: page 8 – 32—PL/SQL and Java
- ResumeProcess: page 8 – 34—PL/SQL and Java
- AbortProcess: page 8 – 36—PL/SQL and Java
- CreateForkProcess: page 8 – 38—PL/SQL
- StartForkProcess: page 8 – 40—PL/SQL
- Background: page 8 – 41—PL/SQL
- AddItemAttribute: page 8 – 43—PL/SQL and Java
- AddItemAttributeArray: page 8 – 46—PL/SQL
- SetItemAttribute: page 8 – 48—PL/SQL and Java
- SetItemAttrDocument: page 8 – 51—PL/SQL and Java
- SetItemAttributeArray: page 8 – 53—PL/SQL
- getItemTypes: page 8 – 56—Java
- GetItemAttribute: page 8 – 57—PL/SQL

- GetItemAttrDocument: page 8 – 59—PL/SQL
- GetItemAttrClob: page 8 – 60—PL/SQL
- getItemAttributes: page 8 – 61—Java
- GetItemAttrInfo: page 8 – 62—PL/SQL
- GetActivityAttrInfo: page 8 – 63—PL/SQL
- GetActivityAttribute: page 8 – 64—PL/SQL
- GetActivityAttrClob: page 8 – 66—PL/SQL
- BeginActivity: page 8 – 67—PL/SQL
- CompleteActivity: page 8 – 69—PL/SQL
- CompleteActivityInternalName: page 8 – 72—PL/SQL
- AssignActivity: page 8 – 74—PL/SQL
- Event: page 8 – 75—PL/SQL
- HandleError: page 8 – 77—PL/SQL
- SetItemParent: page 8 – 79—PL/SQL
- ItemStatus: page 8 – 80—PL/SQL and Java
- getProcessStatus: page 8 – 81—Java

## See Also

Standard API for PL/SQL Procedures Called by a Function Activities:  
page 7 – 3



---

## CreateProcess

**PL/SQL Syntax**    `procedure CreateProcess`

```
(itemtype in varchar2,  
itemkey in varchar2,  
process in varchar2 default '',  
user_key in varchar2 default null,  
owner_role in varchar2 default null);
```

**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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>itemtype</b>	A valid item type. Item types are defined in the Workflow Builder.
<b>itemkey</b>	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 `#SYNCH` as the `itemkey` to create a forced synchronous process. See: Synchronous, Asynchronous, and Forced Synchronous Processes: page 8 – 14.

<b>process</b>	An optional argument that allows the selection of a particular process for that item. Provide the process internal name. If <code>process</code> 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.

**user\_key** A user-friendly key to assign to the item identified by the specified item type and item key. This argument is optional.

**owner\_role** A valid role to set as the owner of the item. This argument is optional.

**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.

**Example** The following code excerpt shows an example of how to call *createProcess()* in a Java program. The example code is from the *WFTTest.java* program.

```
// create an item
if (WFEEngineAPI.createProcess(ctx, iType, iKey, pr))
    System.out.println("Created Item");
else
{
    System.out.println("createProcess failed");
    WFEEngineAPI.showError(ctx);
}
```

---

## 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)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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: <a href="#">CreateProcess</a> : page 8 – 21.
<b>userkey</b>	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)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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 – 21.

---

## 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);
```

**Java Syntax**

```
public static boolean setItemOwner
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String owner)
```

**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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>itemtype</b>	A valid item type. Item types are defined in the Workflow Builder.
<b>itemkey</b>	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.
<b>owner</b>	A valid role.

**Example** The following code excerpt shows an example of how to call *setItemOwner()* in a Java program. The example code is from the WFTTest.java program.

```
// set item owner
if (WFEngineAPI.setItemOwner(ctx, itemType, itemKey, owner))
    System.out.println("Set Item Owner: "+owner);
else
{
    System.out.println("Cannot set owner.");
}
```

```
WFEngineAPI.showError(ctx);  
}
```

---

## 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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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 – 21.

**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 #SYNCH as the itemkey to create a forced synchronous process. See: Synchronous, Asynchronous, and Forced Synchronous Processes: page 8 – 14.

**Example** The following code excerpt shows an example of how to call *startProcess()* in a Java program. The example code is from the *WFTTest.java* program.

```
// start a process
if (WFEngineAPI.startProcess(ctx, iType, iKey))
    System.out.println("Process Started successfully");
else
{
    System.out.println("launch failed");
    WFEngineAPI.showError(ctx);
}
```

---

## 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 – 6.
- 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 #SYNCH as the itemkey to create a forced synchronous process. See: Synchronous, Asynchronous, and Forced Synchronous Processes: page 8 – 14.
- 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.

<b>userkey</b>	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.
<b>owner</b>	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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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 – 21.
<b>process</b>	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, <code>&lt;parent_process_internal_name&gt;:&lt;label_name&gt;</code> . If this argument is null, the top level process for the item is suspended. This argument defaults to null.

**Example**    The following code excerpt shows an example of how to call *suspendProcess()* in a Java program. The example code is from the *WFTest.java* program.

```
// suspend, status should become SUSPEND
System.out.println("Suspend Process " + iType + "/" + iKey +
    " ...");
if (WFEEngineAPI.suspendProcess(ctx, iType, iKey, null))
    System.out.println("Seems to suspend successfully");
else
{
    System.out.println("suspend failed");
    WFEEngineAPI.showError(ctx);
}
```

---

## 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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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 – 21.
<b>process</b>	An optional argument that allows the selection of a particular subprocess for that item type. Provide the process activity's label name. If the process activity label name does not uniquely identify the subprocess you can precede the label name with the internal name of its parent process. For example, <code>&lt;parent_process_internal_name&gt;:&lt;label_name&gt;</code> . If this argument is null, the top level process for the item is resumed. This argument defaults to null.

**Example**    The following code excerpt shows an example of how to call `resumeProcess()` in a Java program. The example code is from the `WFTest.java` program.

```
// resume process and status should be ACTIVE
System.out.println("Resume Process " + iType + "/" + iKey +
    " ...");
if (WFEEngineAPI.resumeProcess(ctx, iType, iKey, null))
    System.out.println("Seems to resume successfully");
else
{
    System.out.println("resume failed");
    WFEEngineAPI.showError(ctx);
}
```

---

## 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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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 – 21.
<b>process</b>	An optional argument that allows the selection of a particular subprocess for that item type. Provide the process activity's label name. If the process activity label name does not uniquely identify the subprocess you can precede the label name with the internal name of its parent process. For example, <code>&lt;parent_process_internal_name&gt;:&lt;label_name&gt;</code> . 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".

**Example** The following code excerpt shows an example of how to call *abortProcess()* in a Java program. The example code is from the *WFTTest.java* program.

```
// abort process, should see status COMPLETE with result
// code force
System.out.println("Abort Process ..." + iType + "/" +
    iKey);
if (!WFEngineAPI.abortProcess(ctx, iType, iKey, pr, null))
{
    System.out.println("Seemed to have problem aborting...");
    WFEngineAPI.showError(ctx);
}
```

---

## 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)

<b>copy_itemtype</b>	A valid item type for the original process to be copied. The new process will have the same item type.
<b>copy_itemkey</b>	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.
<b>new_itemkey</b>	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.
<b>same_version</b>	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, <i>CreateForkProcess()</i> copies the item attributes and status of the original process to the new process. If you specify FALSE, <i>CreateForkProcess()</i> copies the item attributes of the original process to the new

process but does 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:** When you fork an item, Oracle Workflow automatically creates an item attribute called #FORKED\_FROM for the new item and sets the attribute to the item key of the original item. This attribute provides an audit trail for the forked item.

---

## 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 '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)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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 FALSE,  
process_stuck in boolean default FALSE);
```

**Description**    Runs a background engine for processing deferred activities, timed out activities, and stuck processes 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 or processes that become stuck 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 16 – 7.

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 – 45

### Arguments (input)

<b>itemtype</b>	A valid item type. If the item type is null the Workflow engine will run for all item types.
<b>minthreshold</b>	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.
<b>maxthreshold</b>	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 FALSE.
- process\_stuck** Specify TRUE or FALSE to indicate whether to run stuck processes. Defaults to FALSE.

### 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 Home *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.

---

## AddItemAttribute

**PL/SQL Syntax**

```
procedure AddItemAttr
(
  itemtype in varchar2,
  itemkey in varchar2,
  aname in varchar2,
  text_value in varchar2 default null,
  number_value in number default null,
  date_value in date default null);
```

**Java Syntax**

```
public static boolean addItemAttr
(
  WFContext wCtx,
  String itemType,
  String itemKey,
  String aName)

public static boolean addItemAttrText
(
  WFContext wCtx,
  String itemType,
  String itemKey,
  String aName,
  String aValue)

public static boolean addItemAttrNumber
(
  WFContext wCtx,
  String itemType,
  String itemKey,
  String aName,
  BigDecimal aValue)

public static boolean addItemAttrDate
(
  WFContext wCtx,
  String itemType,
  String itemKey,
  String aName,
  String aValue)
```

**Description** Adds a new item type attribute variable to the process. Although most item type attributes are defined at design time, you can create new attributes at runtime for a specific process. You can optionally set a default text, number, or date value for a new item type attribute when the attribute is created.

**Note:** If you are using Java, choose the correct method for your attribute type. To add an empty item type attribute, use *addItemAttr()*. When adding an item type attribute with a default value, use *addItemAttrText()* for all attribute types except number and date.



**Attention:** If you need to add large numbers of item type attributes at once, use the *AddItemAttributeArray* APIs rather than the *AddItemAttribute* APIs for improved performance. See: *AddItemAttributeArray*; page 8 – 46

## Arguments (input)

<b>wCtx</b>	Workflow context information. Required for the Java methods only. See: <i>Oracle Workflow Context</i> ; page 8 – 6.
<b>itemtype</b>	A valid item type.
<b>itemkey</b>	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: <i>CreateProcess</i> ; page 8 – 21.
<b>aname</b>	The internal name of the item type attribute.
<b>text_value</b>	The default text value for the item type attribute. Required for the PL/SQL procedure only. Defaults to null.
<b>number_value</b>	The default number value for the item type attribute. Required for the PL/SQL procedure only. Defaults to null.
<b>date_value</b>	The default date value for the item type attribute. Required for the PL/SQL procedure only. Defaults to null.
<b>aValue</b>	The default value for the item type attribute. Required for the <i>addItemAttrText()</i> , <i>addItemAttrNumber()</i> , and <i>addItemAttrDate()</i> Java methods only.

**Example** The following example shows how API calls can be simplified by using *addItemAttr()* to set the default value of a new item type attribute at the time of creation.

Using *AddItemAttr()* to create the new attribute and *SetItemAttrText()* to set the value of the attribute, the following calls are required:



```
AddItemAttr('ITYPE', 'IKEY', 'NEWCHAR_VAR');  
SetItemAttrText('ITYPE', 'IKEY', 'NEWCHAR_VAR',  
                'new text values');
```

Using *AddItemAttr()* both to create the new attribute and to set its value, only the following call is required:

```
AddItemAttr('ITYPE', 'IKEY', 'NEWCHAR_VAR',  
            'new text values');
```

---

## AddItemAttributeArray

```
PL/SQL Syntax  procedure AddItemAttrTextArray
                (itemtype in varchar2,
                 itemkey in varchar2,
                 aname in Wf_Engine.NameTabTyp,
                 avalue in Wf_Engine.TextTabTyp);

                procedure AddItemAttrNumberArray
                (itemtype in varchar2,
                 itemkey in varchar2,
                 aname in Wf_Engine.NameTabTyp,
                 avalue in Wf_Engine.NumTabTyp);

                procedure AddItemAttrDateArray
                (itemtype in varchar2,
                 itemkey in varchar2,
                 aname in Wf_Engine.NameTabTyp,
                 avalue in Wf_Engine.DateTabTyp);
```

**Description** Adds an array of new item type attributes to the process. Although most item type attributes are defined at design time, you can create new attributes at runtime for a specific process. Use the *AddItemAttributeArray* APIs rather than the *AddItemAttribute* APIs for improved performance when you need to add large numbers of item type attributes at once.

Use the correct procedure for your attribute type. All attribute types except number and date use *AddItemAttrTextArray*.

**Note:** The *AddItemAttributeArray* APIs use PL/SQL table composite datatypes defined in the WF\_ENGINE package. The following table shows the column datatype definition for each PL/SQL table type.

PL/SQL Table Type	Column Datatype Definition
NameTabTyp	Wf_Item_Attribute_Values.NAME%TYPE
TextTabTyp	Wf_Item_Attribute_Values.TEXT_VALUE%TYPE

Table 8 – 1 (Page 1 of 2)

PL/SQL Table Type	Column Datatype Definition
NumTabTyp	Wf_Item_Attribute_Values.NUMBER_VALUE%TYPE
DateTabTyp	Wf_Item_Attribute_Values.DATE_VALUE%TYPE

Table 8 – 1 (Page 2 of 2)

### Arguments (input)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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 – 21.
<b>aname</b>	An array of the internal names of the new item type attributes.
<b>avalue</b>	An array of the values for the new item type attributes.

---

## 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);

procedure SetItemAttrEvent
    (itemtype in varchar2,
     itemkey in varchar2,
     name in varchar2,
     event in wf_event_t);
```

**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, date, and event use *SetItemAttrText*.



**Attention:** If you need to set the values of large numbers of item type attributes at once, use the *SetItemAttributeArray* APIs rather than the *SetItemAttribute* APIs for improved performance. See: *SetItemAttributeArray*: page 8 – 53

### Arguments (input)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: <i>Oracle Workflow Context</i> : page 8 – 6.
<b>itemtype</b>	A valid item type.
<b>itemkey</b>	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: <i>CreateProcess</i> : page 8 – 21.

**Note:** You can pass #SYNCH as the itemkey to create a forced synchronous process. See: *Synchronous, Asynchronous, and Forced Synchronous Processes*: page 8 – 14.

<b>aname or name</b>	The internal name of the item type attribute.
<b>avalue or event</b>	The value for the item type attribute.

**Example 1** The following code excerpt shows an example of how to call *setItemAttrText()* in a Java program. The example code is from the *WFTest.java* program.

```
if (WFEngineAPI.setItemAttrText(ctx, itemType, itemKey,  
"REQUESTOR_USERNAME", owner))  
    System.out.println("Requestor: "+owner);  
else  
{  
    WFEngineAPI.showError(ctx);  
}
```

**Example 2** If an event message is stored within an item attribute of type event, you can access the event data CLOB within that event message by creating an item attribute of type URL for the event data. The following sample PL/SQL code shows how to set the value of the URL attribute to reference the event data.

```
l_eventdataurl := Wfa_html.base_url || 'Wf_Event_Html.  
EventDataContents?P_EventAttribute=EVENT_MESSAGE' || '&' ||  
'P_ItemType=' || itemtype || '&' || 'P_ItemKey=' || itemkey || '&' ||  
'p_mime_type=text/xml';  
  
WF_ENGINE.SetItemAttrText('<item_type>', '<item_key>',  
                          'EVENTDATAURL', l_eventdataurl);
```

If you have applied a stylesheet to the event data XML document to create HTML, set the `p_mime_type` parameter in the URL to `text/html` instead.

If you omit the `p_mime_type` parameter from the URL, the MIME type defaults to `text/xml`.

## See Also

Event Message Structure: page 8 – 242

---

## SetItemAttrDocument



**Attention:** Document management functionality is reserved for future use. This description of the SetItemAttrDocument API is provided for reference only.

**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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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 – 21.

**Note:** You can pass #SYNCH as the itemkey to create a forced synchronous process. See: Synchronous, Asynchronous, and Forced Synchronous Processes: page 8 – 14.

<b>aname</b>	The internal name of the item type attribute.
<b>documentid</b>	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.

*<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.



---

## SetItemAttributeArray

**PL/SQL Syntax**

```
procedure SetItemAttrTextArray
    (itemtype in varchar2,
     itemkey in varchar2,
     aname in Wf_Engine.NameTabTyp,
     avalue in Wf_Engine.TextTabTyp);

procedure SetItemAttrNumberArray
    (itemtype in varchar2,
     itemkey in varchar2,
     aname in Wf_Engine.NameTabTyp,
     avalue in Wf_Engine.NumTabTyp);

procedure SetItemAttrDateArray
    (itemtype in varchar2,
     itemkey in varchar2,
     aname in Wf_Engine.NameTabTyp,
     avalue in Wf_Engine.DateTabTyp);
```

**Description** Sets the values of an array of item type attributes in a process. Use the *SetItemAttributeArray* APIs rather than the *SetItemAttribute* APIs for improved performance when you need to set the values of large numbers of item type attributes at once.

Use the correct procedure for your attribute type. All attribute types except number and date use *SetItemAttrTextArray*.

**Note:** The *SetItemAttributeArray* APIs use PL/SQL table composite datatypes defined in the WF\_ENGINE package. The following table shows the column datatype definition for each PL/SQL table type.

PL/SQL Table Type	Column Datatype Definition
NameTabTyp	Wf_Item_Attribute_Values.NAME%TYPE
TextTabTyp	Wf_Item_Attribute_Values.TEXT_VALUE%TYPE
NumTabTyp	Wf_Item_Attribute_Values.NUMBER_VALUE%TYPE
DateTabTyp	Wf_Item_Attribute_Values.DATE_VALUE%TYPE

Table 8 – 2 (Page 1 of 1)

## Arguments (input)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: <i>CreateProcess</i> : page 8 – 21.
<b>aname</b>	An array of the internal names of the item type attributes.
<b>avalue</b>	An array of the values for the item type attributes.

**Example** The following example shows how using the *SetItemAttributeArray* APIs rather than the *SetItemAttribute* APIs can help reduce the number of calls to the database.

Using *SetItemAttrText()*:

```
SetItemAttrText('ITYPE', 'IKEY', 'VAR1', 'value1');
SetItemAttrText('ITYPE', 'IKEY', 'VAR2', 'value2');
SetItemAttrText('ITYPE', 'IKEY', 'VAR3', 'value3');
```

```
// Multiple calls to update the database.
```

Using *SetItemAttrTextArray()*:

```
declare
    varname    Wf_Engine.NameTabTyp;
    varval     Wf_Engine.TextTabTyp;
begin
    varname(1) := 'VAR1';
    varval(1)  := 'value1';
    varname(2) := 'VAR2';
    varval(2)  := 'value2';
    varname(3) := 'VAR3';
    varval(3)  := 'value3';
    Wf_Engine.SetItemAttrTextArray('ITYPE', 'IKEY', varname,
                                   varval);
exception
    when OTHERS then
        // handle your errors here
        raise;
end;
```

```
// Only one call to update the database.
```



---

## 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;

function GetItemAttrEvent
    (itemtype in varchar2,
     itemkey in varchar2,
     name in varchar2) return wf_event_t;
```

**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, date, and event use *GetItemAttrText*.

### Arguments (input)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: <i>CreateProcess</i> : page 8 – 21.
	<b>Note:</b> Pass #SYNCH as the itemkey to create a forced synchronous process. See: <i>Synchronous, Asynchronous, and Forced Synchronous Processes</i> : page 8 – 14.
<b>aname</b>	The internal name of an item type attribute, for <i>GetItemAttrText()</i> , <i>GetItemAttrNumber()</i> , and <i>GetItemAttrDate()</i> .
<b>name</b>	The internal name of an item type attribute, for <i>GetItemAttrEvent()</i> .

## See Also

Event Message Structure: page 8 – 242

---

## GetItemAttrDocument



**Attention:** Document management functionality is reserved for future use. This description of the GetItemAttrDocument API is provided for reference only.

### PL/SQL Syntax

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

### Description

Returns the document identifier for a DM 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.

*<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)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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: <a href="#">CreateProcess: page 8 – 21</a> .
	<b>Note:</b> Pass #SYNCH as the itemkey to create a forced synchronous process. See: <a href="#">Synchronous, Asynchronous, and Forced Synchronous Processes: page 8 – 14</a> .
<b>aname</b>	The internal name of the item type attribute.

---

## GetItemAttrClob

**PL/SQL Syntax**

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

**Description** Returns the value of an item type attribute in a process as a character large object (CLOB).

### Arguments (input)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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 – 21.
<b>aname</b>	The internal name of an item type attribute.



---

## getItemAttributes

**Java Syntax**

```
public static WFTwoDDataSource getItemAttributes  
    (WFContext wCtx,  
     String itemType,  
     String itemKey)
```

**Description** Returns a list of all the item attributes, their types, and their values for the specified item type instance as a two dimensional data object.

### Arguments (input)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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 – 21.

---

## 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)

<b>itemtype</b>	A valid item type.
<b>aname</b>	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)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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: <a href="#">CreateProcess</a> : page 8 – 21.
<b>actid</b>	The activity ID for a particular usage of an activity in a process definition. Also referred to as the activity ID of the node
<b>aname</b>	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;

function GetActivityAttrEvent
    (itemtype in varchar2,
     itemkey in varchar2,
     actid in number,
     name in varchar2) return wf_event_t;
```

**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)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: <i>CreateProcess</i> : page 8 – 21.

**Note:** Pass #SYNCH as the itemkey to create a forced synchronous process. See: Synchronous, Asynchronous, and Forced Synchronous Processes: page 8 – 14.

<b>actid</b>	The activity ID for a particular usage of an activity in a process definition. Also referred to as the activity ID of the node.
<b>aname</b>	The internal name of an activity attribute, for <i>GetActivityAttrText()</i> , <i>GetActivityAttrNumber()</i> , and <i>GetActivityAttrDate()</i> .
<b>name</b>	The internal name of an activity attribute, for <i>GetActivityAttrEvent()</i> .

## See Also

Event Message Structure: page 8 – 242

---

## GetActivityAttrClob

**PL/SQL Syntax**

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

**Description** Returns the value of an activity attribute in a process as a character large object (CLOB).

### Arguments (input)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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 – 21.
<b>actid</b>	The activity ID for a particular usage of an activity in a process definition. Also referred to as the activity ID of the node.
<b>aname</b>	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 – 69.

### Arguments (input)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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.
<b>activity</b>	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, <i>&lt;parent_process_internal_name&gt;:&lt;label_name&gt;</i> .

**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 create and start an item**—You can call *CompleteActivity()* for a 'Start' activity to implicitly create and start 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.

Use *CompleteActivity()* if you cannot use *CreateProcess()* and *StartProcess()* to start your process. For example, call *CompleteActivity()* if you need 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.

**Note:** Starting a process using *CompleteActivity()* differs from starting a process using *CreateProcess()* and *StartProcess()* in these ways:

- The 'Start' activity called with *CompleteActivity()* may or may not have incoming transitions. *StartProcess()* executes only 'Start' activities that do not have any incoming transitions.
- *CompleteActivity()* only completes the single 'Start' activity with which it is called. Other 'Start' activities in the process are not completed. *StartProcess()*, however, executes every activity in the process that is marked as a 'Start' activity and does not have any incoming transitions.

- *CompleteActivity()* does not execute the activity with which it is called; it simply marks the activity as complete. *StartProcess()* does execute the 'Start' activities with which it starts a process.
- When you use *CompleteActivity()* to start a new process, the item type of the activity being completed must either have a selector function defined to choose a root process, or have exactly one runnable process with the activity being completed marked as a 'Start' activity. You cannot explicitly specify a root process as you can with *StartProcess()*.

## Arguments (input)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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.
<b>activity</b>	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, <i>&lt;parent_process_internal_name&gt;:&lt;label_name&gt;</i> . This activity node must be marked as a 'Start' activity.
<b>result_code</b>	An optional activity completion result. Possible values are determined by the process activity's Result Type, or one of the engine standard results. See: <i>AbortProcess</i> : page 8 – 36.

**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');`



**result**

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 – 36.

---

## 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)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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.
<b>activity</b>	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, <i>&lt;parent_process_internal_name&gt;:&lt;label_name&gt;</i> .
<b>performer</b>	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.

---

## Event

**PL/SQL Syntax**    `procedure Event`

```
    (itemtype in varchar2,  
     itemkey in varchar2,  
     process_name in varchar2 default null,  
     event_message in wf_event_t);
```

**Description**    Receives an event from the Business Event System into a workflow process.

If the specified item key already exists, the event is received into that item. If the item key does not already exist, but the specified process includes an eligible Receive event activity marked as a Start activity, the Workflow Engine creates a new item running that process.

Within the workflow process that receives the event, the procedure searches for eligible Receive event activities. An activity is only eligible to receive an event if its event filter is either blank or set to that particular event. Additionally, the activity must have an appropriate status.

- An activity marked as a Start activity can only receive an event if it has never been executed.
- A normal activity can only receive an event if its activity status is NOTIFIED, meaning the process has transitioned to that activity and is waiting to receive the event.

For each eligible Receive event activity, *Event()* stores the event name, event key, and event message in the item type attributes specified in the event activity node, if they have been defined. Additionally, the procedure sets any parameters in the event message parameter list as item type attributes for the process, creating new item type attributes if a corresponding attribute does not already exist for any parameter. Then the Workflow Engine begins a thread of execution from the event activity.

If no eligible Receive event activity exists for a received event, the procedure returns an exception and an error message.

**Note:** If the event received by a Receive event activity was originally raised by a Raise event activity in another workflow process, the item type and item key for that process are included in the parameter list within the event message. In this case, the Workflow Engine automatically sets the specified process as the parent for the process that receives the event,

overriding any existing parent setting. See: SetItemParent: page 8 – 79.

## Arguments (input)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	A string that uniquely identifies the item within an item type. The item type and key together identify the process.
<b>process_name</b>	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, <i>&lt;parent_process_internal_name&gt;:&lt;label_name&gt;</i> . If this argument is null, the top level process for the item is started. This argument defaults to null.
<b>event_message</b>	The event message containing the details of the event.



---

## 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 – 10. 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)

<b>item_type</b>	A valid item type.
<b>item_key</b>	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.
<b>activity</b>	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_ENGINE.AbortProcess( )`.
- Purge the existing item by calling `WF_ENGINE.Items( )`.
- Revise the process.
- Recreate the item by calling `WF_ENGINE.CreateProcess( )`.
- Restart the revised process at the appropriate activity by calling `WF_ENGINE.HandleError( )`.

---

## SetItemParent

**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)

<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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.
<b>parent_itemtype</b>	A valid item type for the parent process.
<b>parent_itemkey</b>	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.
<b>parent_context</b>	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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>itemtype</b>	A valid item type.
<b>itemkey</b>	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.

**Example**    The following code excerpt shows an example of how to call *itemStatus()* in a Java program. The example code is from the WFTest.java program.

```
// get status and result for this item
dataSource = WFEngineAPI.itemStatus(ctx, itemType, iKey);
System.out.print("Status and result for " + itemType + "/" +
    iKey + " = ");
displayDataSource(ctx, dataSource);
```

---

## 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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>itemType</b>	A valid item type.
<b>itemKey</b>	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 – 21.
<b>process</b>	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 Function APIs

The WFFunctionAPI Java class is the abstract class from which the Java procedures for all external Java function activities are derived. This class contains methods for accessing item type and activity attributes, as well as the *execute()* method which forms the main entry point function of the external Java function activity being implemented.

The WFFunctionAPI class is stored in the oracle.apps.fnd.wf Java package. The following list shows the APIs available in this class.



**Attention:** Java is case-sensitive and all Java method names begin with a lower case letter to follow Java naming conventions.

- loadItemAttributes: page 8 – 83
- loadActivityAttributes: page 8 – 84
- getActivityAttr: page 8 – 85
- getItemAttr: page 8 – 87
- setItemAttrValue: page 8 – 88
- execute: page 8 – 89

### See Also

Standard API for Java Procedures Called by Function Activities: page 7 – 8

Function Activity: page 4 – 44

---

## loadItemAttributes

**Java Syntax**    `public void loadItemAttributes`  
                  `(WFContext pWCtx) throws SQLException`

**Description**    Retrieves the item attributes from the database for the item type from which the external Java function was called. The item attributes are not loaded by default due to the performance impact that could occur if the item type contains a large number of item attributes. You can use this method to load the item attributes explicitly before accessing them in your function.

If a database access error occurs, this method throws a `SQLException`.

### Arguments (input)

**pWCtx**            Workflow context information. See: Oracle Workflow Context: page 8 – 6.

---

## loadActivityAttributes

**Java Syntax**    `public void loadActivityAttributes`  
                  `(WFContext pWCtx,`  
                  `String itemType,`  
                  `String iKey,`  
                  `BigDecimal actid) throws SQLException`

**Description**    Retrieves the activity attributes from the database for the specified activity. This method is called by default when the function activity is instantiated and before the *execute()* function is called.

If a database access error occurs, this method throws a SQLException.

### Arguments (input)

<b>pWCtx</b>	Workflow context information. See: Oracle Workflow Context: page 8 – 6.
<b>iType</b>	A valid item type.
<b>iKey</b>	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 – 21.
<b>actid</b>	An activity instance ID.



---

## getActivityAttr

**Java Syntax**    `public WFAAttribute getActivityAttr  
                  (String aName)`

**Java Syntax**    `public WFAAttribute getActivityAttr  
                  (WFContext pWCtx,  
                  String aName) throws SQLException`

**Description**    There are two implementations of *getActivityAttr()*. These methods return the activity attribute information for the specified activity attribute.

- If you call *getActivityAttr(String aName)* with only the activity attribute name, this method returns the activity attribute value but does not attempt to resolve any reference to an item attribute. If an activity attribute does point to an item attribute, this method returns the internal name of the item attribute. With the item attribute name, you can then perform additional processing based on the item attribute.

For example, if you want to write information back to the item attribute, you can first use *getActivityAttr(String aName)* to retrieve the item attribute name. Then use *setItemAttrValue(WFContext pWCtx, WFAAttribute pAttr)* to set the item attribute value, which also becomes the activity attribute value. See: *setItemAttrValue*: page 8 – 88.

- If you call *getActivityAttr(WFContext pWCtx, String aName)* with both the Workflow context and the activity attribute name, this method returns the activity attribute, and if the activity attribute points to an item attribute, the method attempts to resolve the reference by retrieving the value of that item attribute. You can use *getActivityAttr(WFContext pWCtx, String aName)* when you want to obtain the actual activity attribute value, and you do not need to know which item attribute it references. This method attempts to resolve the reference within the previously loaded item attributes, or if the item attributes have not been loaded, the method calls *loadItemAttributes(WFContext pWCtx)* to load them. See: *loadItemAttributes*: page 8 – 83.

If a database access error occurs, this method throws a `SQLException`.

## Arguments (input)

<b>pWCtx</b>	Workflow context information. Required for the second method only. See: Oracle Workflow Context: page 8 – 6.
<b>aName</b>	The internal name of an activity attribute.



---

## setItemAttrValue

**Java Syntax**    `public void setItemAttrValue`  
                  `(WFContext pWCtx,`  
                  `WFAttribute pAttr)`  
                  throws `NumberFormatException, WFException`

**Description**    Sets the value of the specified item attribute in the database.

This method throws a `NumberFormatException` if it cannot convert the value to the appropriate format for an attribute of type number or date. The method throws a `WFException` if it encounters an error while setting an attribute of type document or text.

### Arguments (input)

<b>pWCtx</b>	Workflow context information. See: Oracle Workflow Context: page 8 – 6.
<b>pAttr</b>	The attribute information for an item attribute.




---

## Workflow Attribute APIs

The WFAttribute Java class contains descriptive information for an item or activity attribute, including the internal name of the attribute, attribute value, attribute data type, format information, and default value type. The attribute value is stored as an Object type. This class also contains methods for accessing the attribute information, which can be called by a Java application or the Java procedure for an external Java function activity.

The WFAttribute class is stored in the oracle.apps.fnd.wf Java package. The following list shows the APIs available in this class.

 **Attention:** Java is case-sensitive and all Java method names, except the constructor method names, begin with a lower case letter to follow Java naming conventions.

- WFAttribute: page 8 – 92
- value: page 8 – 93
- getName: page 8 – 94
- getValue: page 8 – 95
- getType: page 8 – 96
- getFormat: page 8 – 97
- getValueType: page 8 – 98
- toString: page 8 – 99
- compareTo: page 8 – 100

### See Also

Standard API for Java Procedures Called by Function Activities: page 7 – 8

---

### WFAttribute Class Constants

The WFAttribute class contains several constants. The following table shows the constants that can be used to represent the data type of an attribute.

Constant Variable Declaration	Constant Value
<code>public static final String TEXT</code>	"TEXT"
<code>public static final String NUMBER</code>	"NUMBER"
<code>public static final String DATE</code>	"DATE"
<code>public static final String LOOKUP</code>	"LOOKUP"
<code>public static final String FORM</code>	"FORM"
<code>public static final String URL</code>	"URL"
<code>public static final String DOCUMENT</code>	"DOCUMENT"
<code>public static final String ROLE</code>	"ROLE"
<code>public static final String EVENT</code>	"EVENT"

**Table 8 – 3 (Page 1 of 1)**

The following table shows the constants that can be used to represent the type of the default value for an attribute. The default value can be either a constant or, for an activity attribute, a reference to an item type attribute.

Constant Variable Declaration	Constant Value
<code>public static final String CONSTANT</code>	"CONSTANT"
<code>public static final String ITEMATTR</code>	"ITEMATTR"

**Table 8 – 4 (Page 1 of 1)**

---

## WFAttribute

**Java Syntax**    `public WFAttribute()`

**Java Syntax**    `public WFAttribute  
                  (String pName  
                  String pType,  
                  Object pValue,  
                  String pValueType)`

**Description**    There are two constructor methods for the WFAttribute class. The first constructor creates a new WFAttribute object. The second constructor creates a new WFAttribute object and initializes it with the specified attribute name, attribute type, value, and value type.

### Arguments (input)

<b>pName</b>	The internal name of an item or activity attribute. Required for the second method only.
<b>pType</b>	The data type of the attribute. Required for the second method only.
<b>pValue</b>	The attribute value. Required for the second method only.
<b>pValueType</b>	The type of the default value for the attribute. The default value can be either a constant or, for an activity attribute, a reference to an item type attribute. Required for the second method only.



---

## value

**Java Syntax**    `public void value`  
                  `(Object pValue)`

**Description**    Sets the value of the item or activity attribute within a `WFAttribute` object. The value must be cast to the `Object` type.



**Attention:** Using `value()` to set the attribute value within a `WFAttribute` object does not set the attribute value in the database. To set the value of an item attribute in the database, use `WFFunctionAPI.setItemAttrValue()`. See: `setItemAttrValue`: page 8 – 88.

### Arguments (input)

<b>pValue</b>	The attribute value.
---------------	----------------------

---

## getName

**Java Syntax**    `public String getName()`

**Description**    Returns the internal name of the item or activity attribute.

---

## getValue

**Java Syntax**    `public Object getValue()`

**Description**    Returns the value of the item or activity attribute as type Object.

---

## getType

**Java Syntax**    `public String getType()`

**Description**    Returns the data type of the item or activity attribute. See: Attribute Types: page 4 – 3.

---

## getFormat

**Java Syntax**    `public String getFormat()`

**Description**    Returns the format string for the item or activity attribute, such as the length for an attribute of type text or the format mask for an attribute of type number or date. See: [To Define an Item Type or Activity Attribute](#): page 4 – 8.

---

## getValueType

**Java Syntax**    `public String getValueType()`

**Description**    Returns the type of the default value for the item or activity attribute. The default value can be either a constant or, for an activity attribute, a reference to an item type attribute. See: To Define an Item Type or Activity Attribute: page 4 – 8.

---

## toString

**Java Syntax**    `public String toString()`

**Description**    Returns the internal name and the value of the item or activity attribute as a string in the following format:

`<name>=<value>`

This method overrides the *toString()* method in the Object class.





---

## 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 – 26.

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 – 102
- GET\_ERROR: page 8 – 103
- TOKEN: page 8 – 104
- RAISE: page 8 – 105
- CONTEXT: page 8 – 108
- TRANSLATE: page 8 – 110

### See Also

Standard API for PL/SQL Procedures Called by Function Activities:  
page 7 – 3

---

## CLEAR

**Syntax**    `procedure CLEAR;`

**Description**    Clears the error buffers.

### See Also

`GET_ERROR`: page 8 – 103

---

## 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 – 102

---

## 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)

<b>token_name</b>	Name of the token.
<b>token_value</b>	Value to substitute for the token.

### See Also

RAISE: page 8 – 105

CONTEXT: page 8 – 108



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 Oracle 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)

<b>pkg_name</b>	Name of the procedure package.
<b>proc_name</b>	Procedure or function name.
<b>arg1</b>	First IN argument.
<b>argn</b>	nth IN argument.

**Example 1**     /\*PL/SQL procedures called by function activities can use the WF\_CORE APIs to raise and catch errors the same way the Workflow Engine does.\*/

```
package My_Package is  
  
  procedure MySubFunction(  
    arg1 in varchar2,  
    arg2 in varchar2)  
  is  
    ...  
  begin  
    if (<error condition>) then  
      Wf_Core.Token('ARG1', arg1);  
      Wf_Core.Token('ARG2', arg2);  
      Wf_Core.Raise('ERROR_NAME');  
    end if;  
    ...  
  exception  
    when others then
```



```

        Wf_Core.Context('My_Package', 'MySubFunction', arg1,
arg2);
        raise;
end MySubFunction;
procedure MyFunction(
    itemtype in varchar2,
    itemkey in varchar2,
    actid in number,
    funcmode in varchar2,
    result out varchar2)
is
...
begin
    ...
    begin
        MySubFunction(arg1, arg2);
    exception
        when others then
            if (Wf_Core.Error_Name = 'ERROR_NAME') then
                -- This is an error I wish to ignore.
                Wf_Core.Clear;
            else
                raise;
            end if;
        end;
    ...
exception
    when others then
        Wf_Core.Context('My_Package', 'MyFunction', itemtype,
itemkey, to_char(actid), funcmode);
        raise;
end MyFunction;

```

## See Also

TOKEN: page 8 – 104

RAISE: page 8 – 105

---

## 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)

<b>tkn_name</b>	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 – 113
- Activities: page 8 – 114
- Notifications: page 8 – 115
- Total: page 8 – 116
- TotalPERM: page 8 – 117

- AdHocDirectory: page 8 – 118

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications, you can 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 – 119.

**Note:** If you are using the standalone version of Oracle Workflow available with Oracle9i Release 2, you can use the standalone Oracle Workflow Manager component available through Oracle Enterprise Manager to submit and manage Workflow purge database jobs. For more information, please refer to the Oracle Workflow Manager online help.

## See Also

Standard API for PL/SQL Procedures Called by Function Activities:  
page 7 – 3

Purging for Performance: page C – 8

---

## Items

**Syntax**    `procedure Items`  
`(itemtype in varchar2 default null,`  
`itemkey in varchar2 default null,`  
`enddate in date default sysdate,`  
`docommit in boolean default TRUE);`

**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)

<b>itemtype</b>	Item type to delete. Leave this argument null to delete all item types.
<b>itemkey</b>	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.
<b>enddate</b>	Specified date to delete up to.
<b>docommit</b>	Specify TRUE or FALSE to indicate whether to commit data while purging. If you want <i>Items()</i> to commit data as it purges to reduce rollback segments and improve performance, specify TRUE. If you do not want to perform automatic commits, specify FALSE. Defaults to TRUE.

---

## 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 that are 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)

<b>itemtype</b>	Item type associated with the activities you want to delete. Leave this argument null to delete activities for all item types.
<b>name</b>	Internal name of activity to delete. Leave this argument null to delete all activities for the specified item type.
<b>enddate</b>	Specified date to delete up to.



---

## Total

**Syntax**     procedure Total

```
                  (itemtype in varchar2 default null,  
                  itemkey in varchar2 default null,  
                  enddate in date default sysdate,  
                  doccommit in boolean default TRUE);
```

**Description**     Deletes all eligible obsolete runtime item type and activity data that is associated with the specified item type and has an END\_DATE less than or equal to the specified end date. *Total()* also deletes ad hoc roles and users that are no longer in use by calling *AdHocDirectory()*. See: *AdHocDirectory*: page 8 – 118.

Because *Total()* purges all activities and ad hoc role information, it is more costly in performance than *Items()*. If you want to purge a specific item key, use *Items()*. Use *Total()* as part of your routine maintenance during periods of low activity. See: *Items*: page 8 – 113.

### Arguments (input)

<b>itemtype</b>	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.
<b>itemkey</b>	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.
<b>enddate</b>	Specified date to delete up to.
<b>doccommit</b>	Specify TRUE or FALSE to indicate whether to commit data while purging. If you want <i>Total()</i> to commit data as it purges to reduce rollback segments and improve performance, specify TRUE. If you do not want to perform automatic commits, specify FALSE. Defaults to TRUE.



---

## TotalPERM

**Syntax**    `procedure TotalPERM`  
`(itemtype in varchar2 default null,`  
`itemkey in varchar2 default null,`  
`enddate in date default sysdate,`  
`docommit in boolean default TRUE);`

**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. *TotalPERM()* is similar to *Total()* except that *TotalPERM()* deletes only items with a persistence type of 'PERM'. See: Total: page 8 – 116.

### Arguments (input)

<b>itemtype</b>	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.
<b>itemkey</b>	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.
<b>enddate</b>	Specified date to delete up to.
<b>docommit</b>	Specify TRUE or FALSE to indicate whether to commit data while purging. If you want <i>TotalPERM()</i> to commit data as it purges to reduce rollback segments and improve performance, specify TRUE. If you do not want to perform automatic commits, specify FALSE. Defaults to TRUE.

---

## 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

If you are using the version of Oracle Workflow embedded in Oracle Applications, you can submit the Purge Obsolete Workflow Runtime Data concurrent program to purge obsolete item type runtime status information. Use the Submit Requests form in Oracle Applications to submit this concurrent program.

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager to submit and manage the Purge Obsolete Workflow Runtime Data concurrent program. For more information, please refer to the Oracle Applications Manager online help.

### ► 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:

<b>Item Type</b>	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.
<b>Item Key</b>	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.
<b>Age</b>	Minimum age of data to purge, in days, if the persistence type is set to 'TEMP'. The default is 0.

**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 – 123
- GetUserRoles: page 8 – 124
- GetRoleInfo: page 8 – 125
- GetRoleInfo2: page 8 – 126
- IsPerformer: page 8 – 127
- UserActive: page 8 – 128
- GetUserName: page 8 – 129
- GetRoleName: page 8 – 130
- GetRoleDisplayName: page 8 – 131
- SetAdHocUserStatus: page 8 – 132
- SetAdHocRoleStatus: page 8 – 133
- CreateAdHocUser: page 8 – 134
- CreateAdHocRole: page 8 – 136
- AddUsersToAdHocRole: page 8 – 138
- SetAdHocUserExpiration: page 8 – 139
- SetAdHocRoleExpiration: page 8 – 140
- SetAdHocUserAttr: page 8 – 141
- SetAdHocRoleAttr: page 8 – 142
- RemoveUsersFromAdHocRole: page 8 – 143



**Attention:** If you implement OID integration, you must maintain your users only through OID. You must not create ad hoc users in the WF\_LOCAL\_USERS table, because you risk discrepancies in your user information and unpredictable results if you use any tool other than OID to maintain users after integrating with OID. Consequently, if you implement OID integration, you must not use the *CreateAdHocUser()*, *SetAdHocUserStatus()*, *SetAdHocUserExpiration()*, or *SetAdHocUserAttr()* APIs in the WF\_DIRECTORY package.

You can still use ad hoc roles, however, since Workflow roles are not maintained through OID.

## **See Also**

Standard API for PL/SQL Procedures Called by Function Activities:  
page 7 – 3



---

## 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
- E-mail 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', 'MAILATTH', 'SUMMARY')
- Language
- Territory
- E-mail 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)

<b>user</b>	A valid username.
<b>role</b>	A valid role name.

---

## UserActive

**Syntax**

```
function UserActive
(username in varchar2)
return boolean;
```

**Description** Determines if a user currently has a status of 'ACTIVE' and is available to participate in a workflow. Returns TRUE if the user has a status of 'ACTIVE', otherwise it returns FALSE.

### Arguments (input)

**username** A valid username.

---

## 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'.



**Attention:** If you implement Oracle Internet Directory integration, you must maintain your users only through OID. You must not use the *SetAdHocUserStatus( )* API to update user information in the WF\_LOCAL\_USERS table, because you risk discrepancies in your user information and unpredictable results if you use any tool other than OID to maintain users after integrating with OID.

### Arguments (input)

<b>user_name</b>	The internal name of the user.
<b>status</b>	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)

<b>role_name</b>	The internal name of the role.
<b>status</b>	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.



**Attention:** If you implement Oracle Internet Directory integration, you must maintain your users only through OID. You must not use the *CreateAdHocUser()* API to create new users in the WF\_LOCAL\_USERS table, because you risk discrepancies in your user information and unpredictable results if you use any tool other than OID to maintain users after integrating with OID.

### Arguments (input)

<b>name</b>	An internal name for the user. The name must be 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.
<b>display_name</b>	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 a display name, the system generates one for you where the display name contains a prefix of '~WF_ADHOC-' followed by a sequence number.

<b>language</b>	The value of the database 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.
<b>territory</b>	The value of the database 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.
<b>description</b>	An optional description for the user.
<b>notification_preference</b>	Indicate how this user prefers to receive notifications: 'MAILTEXT', 'MAILHTML', 'MAILATTH', 'QUERY' or 'SUMMARY'. If null, the procedure sets the notification preference to 'MAILHTML'.
<b>email_address</b>	A optional electronic mail address for this user.
<b>fax</b>	An optional Fax number for the user.
<b>status</b>	The availability of the user to participate in a workflow process. The possible statuses are 'ACTIVE', 'EXTLEAVE', 'INACTIVE', and 'TMPLAVE'. If null, the procedure sets the status to 'ACTIVE'.
<b>expiration_date</b>	The date at which the user is no longer valid in the directory service. If null, the procedure defaults the expiration date to sysdate.

## See Also

Setting Up an Oracle Workflow Directory Service: page 2 – 21

---

## 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)

<b>role_name</b>	An internal name for the role. The name must be 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.
<b>role_display_name</b>	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 a display name, the system generates one for you where the display name contains a prefix of '~WF_ADHOC-' followed by a sequence number.
<b>language</b>	The value of the database 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.
<b>territory</b>	The value of the database 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.
<b>role_description</b>	An optional description for the role.
<b>notification_preference</b>	Indicate how this role receives notifications: ‘MAILTEXT’, ‘MAILHTML’, ‘MAILATTH’, ‘QUERY’ or ‘SUMMARY’. If null, the procedure sets the notification preference to ‘MAILHTML’.
<b>role_users</b>	Indicate the names of the users that belong to this role, using commas or spaces to delimit the list.
<b>email_address</b>	A optional electronic mail address for this role or a mail distribution list defined by your electronic mail system.
<b>fax</b>	An optional Fax number for the role.
<b>status</b>	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’.
<b>expiration_date</b>	The date at which the role is no longer valid in the directory service. If null, the procedure defaults the expiration date to sysdate.

## See Also

Setting Up an Oracle Workflow Directory Service: page 2 – 21

---

## AddUsersToAdHocRole

**Syntax**    `procedure AddUsersToAdHocRole`  
                  `(role_name in varchar2,`  
                  `role_users in varchar2);`

**Description**    Adds users to a existing ad hoc role.

### Arguments (input)

<b>role_name</b>	The internal name of the ad hoc role.
<b>role_users</b>	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.



**Attention:** If you implement Oracle Internet Directory integration, you must maintain your users only through OID. You must not use the *SetAdHocUserExpiration()* API to update user information in the WF\_LOCAL\_USERS table, because you risk discrepancies in your user information and unpredictable results if you use any tool other than OID to maintain users after integrating with OID.

### Arguments (input)

<b>user_name</b>	The internal name of the ad hoc user.
<b>expiration_date</b>	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)

<b>role_name</b>	The internal name of the ad hoc role.
<b>expiration_date</b>	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.



**Attention:** If you implement Oracle Internet Directory integration, you must maintain your users only through OID. You must not use the *SetAdHocUserAttr()* API to update user information in the WF\_LOCAL\_USERS table, because you risk discrepancies in your user information and unpredictable results if you use any tool other than OID to maintain users after integrating with OID.

### Arguments (input)

<b>user_name</b>	The internal name of the ad hoc user to update.
<b>display_name</b>	A new display name for the ad hoc user. If null, the display name is not updated.
<b>notification_preference</b>	A new notification preference of 'MAILTEXT', 'MAILHTML', 'MAILATTH', 'QUERY' or 'SUMMARY'. If null, the notification preference is not updated.
<b>language</b>	A new value of the database NLS_LANGUAGE initialization parameter for the ad hoc user. If null, the language setting is not updated.
<b>territory</b>	A new value of the database NLS_TERRITORY initialization parameter for the ad hoc user. If null, the territory setting is not updated.
<b>email_address</b>	A new valid electronic mail address for the ad hoc user. If null, the electronic mail address is not updated.
<b>fax</b>	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)

<b>role_name</b>	The internal name of the ad hoc role to update.
<b>display_name</b>	A new display name for the ad hoc role. If null, the display name is not updated.
<b>notification_preference</b>	A new notification preference of 'MAILTEXT', 'MAILHTML', 'QUERY' or 'SUMMARY'. If null, the notification preference is not updated.
<b>language</b>	A new value of the database NLS_LANGUAGE initialization parameter for the ad hoc role. If null, the language setting is not updated.
<b>territory</b>	A new value of the database NLS_TERRITORY initialization parameter for the ad hoc role. If null, the territory setting is not updated.
<b>email_address</b>	A new valid electronic mail address for the ad hoc role. If null, the electronic mail address is not updated.
<b>fax</b>	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)

<b>role_name</b>	The internal name of the ad hoc role.
<b>role_users</b>	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 LDAP APIs

Call the following APIs to synchronize local user information in your Workflow directory service with the users in an LDAP directory such as Oracle Internet Directory (OID). These APIs are defined in a PL/SQL package called WF\_LDAP.

- Synch\_changes: page 8 – 145
- Synch\_all: page 8 – 146
- Schedule\_changes: page 8 – 147

### See Also

Synchronizing Workflow Directory Services with Oracle Internet Directory: page 2 – 30

---

## Synch\_changes

**Syntax**     `function synch_changes`  
                  `return boolean;`

**Description**     Determines whether there have been any user changes to an LDAP directory since the last synchronization by querying the LDAP change log records; if there are any changes, including creation, modification, and deletion, *Synch\_changes( )* stores the user attribute information in an attribute cache and raises the `oracle.apps.wf.public.user.change` event to alert interested parties. The function connects to the LDAP directory specified in the Global Workflow Preferences. One event is raised for each changed user.

If the function completes successfully, it returns `TRUE`; otherwise, if it encounters an exception, it returns `FALSE`.

### See Also

Synchronizing Workflow Directory Services with Oracle Internet Directory: page 2 – 30

Setting Global User Preferences: page 2 – 14

User Entry Has Changed Event: page 14 – 15

---

## Synch\_all

**Syntax**     `function synch_all`  
                  `return boolean;`

**Description**     Retrieves all users from an LDAP directory, stores the user attribute information in an attribute cache, and raises the `oracle.apps.wf.public.user.change` event to alert interested parties. The function connects to the LDAP directory specified in the Global Workflow Preferences. One event is raised for each user.

Because *Synch\_all()* retrieves information for all users stored in the LDAP directory, you should use this function only once during setup, or as required for recovery or cleanup. Subsequently, you can use *Synch\_changes()* or *Schedule\_changes()* to retrieve only changed user information.

If the function completes successfully, it returns TRUE; otherwise, if it encounters an exception, it returns FALSE.

Run *Synch\_all()* to begin your Workflow directory service synchronization with Oracle Internet Directory if you implement OID integration.

### See Also

Synchronizing Workflow Directory Services with Oracle Internet Directory: page 2 – 30

Setting Global User Preferences: page 2 – 14

User Entry Has Changed Event: page 14 – 15

*Synch\_changes*: page 8 – 145

*Schedule\_changes*: page 8 – 147

---

## Schedule\_changes

**Syntax** procedure schedule\_changes  
(l\_day in pls\_integer default 0,  
l\_hour in pls\_integer default 0,  
l\_minute in pls\_integer default 10);

**Description** Runs the *Synch\_changes()* API repeatedly at the specified time interval to check for user changes in an LDAP directory and alert interested parties of any changes. The default interval is ten minutes. *Schedule\_changes()* submits a database job using the DBMS\_JOB utility to run *Synch\_changes()*.

Run *Schedule\_changes()* to maintain your Workflow directory service synchronization with Oracle Internet Directory if you implement OID integration.

### Arguments (input)

<b>l_day</b>	The number of days in the interval to specify how often you want to run the <i>Synch_changes()</i> API. The default value is zero.
<b>l_hour</b>	The number of hours in the interval to specify how often you want to run the <i>Synch_changes()</i> API. The default value is zero.
<b>l_minute</b>	The number of minutes in the interval to specify how often you want to run the <i>Synch_changes()</i> API. The default value is ten.

### See Also

Synch\_changes: page 8 – 145

Synchronizing Workflow Directory Services with Oracle Internet Directory: page 2 – 30

---

## 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 retrieve an access key or to generate a complete URL to access the various pages of the Workflow Monitor. All Workflow Monitor APIs are defined in the PL/SQL package called WF\_MONITOR.

- GetAccessKey: page 8 – 150
- GetDiagramURL: page 8 – 151
- GetEnvelopeURL: page 8 – 153
- GetAdvancedEnvelopeURL: page 8 – 155



**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.

---

## GetAccessKey

**Syntax**

```
function GetAccessKey
    (x_item_type varchar2,
     x_item_key  varchar2,
     x_admin_mode varchar2)
    return varchar2;
```

**Description** Retrieves the access key password that controls access to the Workflow Monitor. Each process instance has separate access keys for running the Workflow Monitor in 'ADMIN' mode or 'USER' mode.

### Arguments (input)

<b>x_item_type</b>	A valid item type.
<b>x_item_key</b>	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.
<b>x_admin_mode</b>	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.

---

## 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 – 14.

*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 your Web server. 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 – 14.

<b>x_item_type</b>	A valid item type.
<b>x_item_key</b>	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.
<b>x_admin_mode</b>	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 – 110

---

## 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 – 14.

`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 your Web server. 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 – 14.

**x\_item\_type** A valid item type.

<b>x_item_key</b>	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.
<b>x_admin_mode</b>	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 – 110

---

## 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 – 14.

`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 your Web server. 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 – 14.

<b>x_item_type</b>	A valid item type.
<b>x_item_key</b>	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.
<b>x_admin_mode</b>	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.
<b>x_options</b>	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 – 110



---

## 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 – 157
- WF\_NOTIFICATION\_ATTR\_RESP\_V: page 8 – 159
- WF\_RUNNABLE\_PROCESSES\_V: page 8 – 160
- WF\_ITEMS\_V: page 8 – 161

**Note:** These database views are public, meaning they are available for you to use for your custom data requirements. This description does not mean that any privileges for these views have been granted to PUBLIC.

---

### 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 – 61.

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 are established. A package of data on the queue is referred to as an event or a message.

**Note:** An event in this context is different from the business events associated with the Business Event System, and 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 Oracle Advanced Queuing concepts and terminology. Refer to the *Oracle Application Developer's Guide – Advanced Queuing* for more information on Advanced Queues.

**Note:** In a future release, this workflow Advanced Queues processing will be implemented within the Business Event System using a specialized queue handler to handle dequeue and enqueue operations.

### Queue APIs

- EnqueueInbound: page 8 – 165
- DequeueOutbound: page 8 – 167
- DequeueEventDetail: page 8 – 170
- PurgeEvent: page 8 – 172
- PurgeItemtype: page 8 – 173
- ProcessInboundQueue: page 8 – 174
- GetMessageHandle: page 8 – 175
- DequeueException: page 8 – 176

- Deferred\_queue: page 8 – 177
- Inbound\_queue: page 8 – 178
- Outbound\_queue: page 8 – 179

### 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 – 180
- CreateMsg: page 8 – 181
- WriteMsg: page 8 – 182
- SetMsgAttr: page 8 – 183
- SetMsgResult: page 8 – 184

### Payload Structure

Oracle Workflow queues use the datatype `system.wf_payload_t` to define the payload for any given message. The payload contains all the information that is required about the event. The following table lists the attributes of `system.wf_payload_t`.

Attribute Name	Datatype	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.

Table 8 – 5 (Page 1 of 2)

Attribute Name	Datatype	Description
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 – 5 (Page 2 of 2)**

## See Also

Standard API for PL/SQL Procedures Called by Function Activities:  
page 7 – 3

Advanced Queuing: *Oracle Application Developer's Guide – Advanced Queuing*



---

## 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,
     error_stack 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 external function activity as complete with the specified result when it processes the inbound queue. The result value is only effective for successful completion, however. If you specify an external program error in the `error_stack` parameter, Oracle Workflow marks the external function activity as complete with an `ERROR` status, overriding the result value. Additionally, if a corresponding error process is defined in the item type, Oracle Workflow launches that error process.

### Arguments (input)

<b>itemtype</b>	The item type of the event.
<b>itemkey</b>	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.
<b>actid</b>	The function activity instance ID that this event is associated with.
<b>result</b>	An optional activity completion result. Possible values are determined by the function activity's Result Type.
<b>attrlist</b>	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. Oracle Advanced Queues allow you to search a queue for messages based on a specific correlation value. If null, the Workflow Engine creates a correlation identifier based on the Workflow schema name and the item type.

**error\_stack**

Specify an optional external program error that will be placed on Oracle Workflow's internal error stack. You can specify any text value up to a maximum length of 200 characters.

---

## 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)

<b>dequeuemode</b>	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.
<b>navigation</b>	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. Oracle 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 Workflow schema name and 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);
    queue_name       varchar2(30);
    navigation_mode  number;
    end_of_queue     boolean;

begin
    queue_name := 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`

```
(dequeueemode 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)

<b>dequeueemode</b>	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.
<b>navigation</b>	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. Oracle 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 Workflow schema name and 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)

<b>queueName</b>	The name of the queue from which to purge the event.
<b>message_handle</b>	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)

<b>queueName</b>	The name of the queue from which to purge the events.
<b>itemType</b>	An optional item type of the events to purge.
<b>correlation</b>	Specify an optional correlation identifier for the message to be purged. Oracle Advanced Queues allow you to search a queue for messages based on a specific correlation value. You can use the Like comparison operator, <code>'%'</code> , to specify the identifier string. If null, the Workflow Engine creates a correlation identifier based on the Workflow schema name and 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 – 165.

### Arguments (input)

<b>itemtype</b>	An optional item type of the events to process.
<b>correlation</b>	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 Workflow schema name and 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)

<b>queueName</b>	The name of the queue from which to retrieve the message handle.
<b>itemType</b>	The item type of the message.
<b>itemKey</b>	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.
<b>actId</b>	The function activity instance ID that this message is associated with.
<b>correlation</b>	Specify an optional correlation identifier for the message. If the correlation is null, the Workflow Engine creates a correlation identifier based on the Workflow schema name and the item type.

---

## DequeueException

**Syntax**      `procedure DequeueException`  
                  `(queueName in varchar2);`

**Description**      Dequeues all messages from an exception queue and places the messages on the standard Business Event System WF\_ERROR queue with the error message 'Message Expired.' When the messages are dequeued from WF\_ERROR, a predefined subscription is triggered that launches the Default Event Error process.

### Arguments (input)

**queueName**              The name of the exception queue that has been enabled for dequeue.

### See Also

Default Event Error Process: page 6 – 34

---

## 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 – 163.



---

## 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 – 163.

### Arguments (input)

<b>itemtype</b>	The item type of the message.
<b>itemkey</b>	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.
<b>actid</b>	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 – 163.

### Arguments (input)

<b>itemtype</b>	The item type of the message.
<b>itemkey</b>	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.
<b>actid</b>	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 – 163.

### Arguments (input)

<b>itemtype</b>	The item type of the message.
<b>itemkey</b>	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.
<b>actid</b>	The function activity instance ID that this message is associated with.
<b>attrName</b>	The internal name of the item attribute you wish to append to the message.
<b>attrValue</b>	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 – 163.

### Arguments (input)

<b>itemtype</b>	The item type of the message.
<b>itemkey</b>	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.
<b>actid</b>	The function activity instance ID that this message is associated with.
<b>result</b>	The completion result for the message. Possible values are determined by the activity's Result Type.

---

## Document Management APIs



**Attention:** Document management functionality is reserved for future use. This description of Oracle Workflow document management APIs is provided for reference only.

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 – 186
- `get_launch_attach_url`: page 8 – 187
- `get_open_dm_display_window`: page 8 – 188
- `get_open_dm_attach_window`: page 8 – 189
- `set_document_id_html`: page 8 – 190

### See Also

Standard API for PL/SQL Procedures Called by Function Activities:  
page 7 – 3

---

## 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)

<b>username</b>	The username of the person accessing the document management system.
<b>document_identifier</b>	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 <i>GetItemAttrDocument</i> API. See: <i>GetItemAttrDocument</i> : page 8 – 59 and <i>SetItemAttrDocument</i> : page 8 – 51.
<b>display_icon</b>	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)

<b>username</b>	The username of the person accessing the document management system.
<b>callback_function</b>	The URL you would like to invoke after the user selects a document to attach. This callback function should be the <code>callback_url</code> syntax that is returned from the <code>set_document_id_html</code> API.
<b>display_icon</b>	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)

<b>frame_name</b>	The name of the HTML frame that you wish to interact with in the current UI.
<b>form_name</b>	The name of the HTML form that you wish to interact with in the current UI.
<b>document_id_field_name</b>	<p>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:</p> <pre>DM:&lt;node_id&gt;:&lt;document_id&gt;:&lt;version&gt;</pre> <p>&lt;nodeid&gt; is the node ID assigned to the document management system node as defined in the Document Management Nodes web page.</p> <p>&lt;documentid&gt; is the document ID of the document, as assigned by the document management system where the document resides.</p> <p>&lt;version&gt; is the version of the document. If a version is not specified, the latest version is assumed.</p>

**document\_name\_** The name of the HTML field in the current UI that  
**field\_name** 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 consists 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 – 195

If the performer role of a notification has a notification preference of MAILTEXT, MAILHTML, MAILATTN 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 – 48.

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 – 194.
- Forward the notification to another role. See: Forwarding a Notification: page 8 – 194.
- Transfer ownership of the notification to another role. See: Transferring a Notification: page 8 – 195.

- Ignore the notification and let it time out. See: Processing a Timed Out Notification: page 8 – 195.

## **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 – 13.

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 – 13.

*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 – 13.

*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 – 61.

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 then 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 – 5.



**Attention:** Java is case-sensitive and all Java method names begin with a lower case letter to follow Java naming conventions.

- Send: page 8 – 199—PL/SQL and Java
- SendGroup: page 8 – 203—PL/SQL
- Forward: page 8 – 205—PL/SQL and Java
- Transfer: page 8 – 207—PL/SQL and Java
- Cancel: page 8 – 209—PL/SQL and Java
- CancelGroup: page 8 – 210—PL/SQL
- Respond: page 8 – 211—PL/SQL and Java
- Responder: page 8 – 212—PL/SQL and Java
- VoteCount: page 8 – 213—PL/SQL and Java
- OpenNotificationsExist: page 8 – 214—PL/SQL and Java
- Close: page 8 – 215—PL/SQL and Java
- AddAttr: page 8 – 216—PL/SQL and Java
- SetAttribute: page 8 – 217—PL/SQL and Java
- GetAttrInfo: page 8 – 219—PL/SQL and Java
- GetInfo: page 8 – 220—PL/SQL and Java
- GetText: page 8 – 221—PL/SQL
- GetShortText: page 8 – 222—PL/SQL
- GetAttribute: page 8 – 223—PL/SQL and Java
- GetAttrDoc: page 8 – 225—PL/SQL and Java
- GetSubject: page 8 – 226—PL/SQL and Java
- GetBody: page 8 – 227—PL/SQL and Java

- GetShortBody: page 8 – 228—PL/SQL
- TestContext: page 8 – 229—PL/SQL
- AccessCheck: page 8 – 230—PL/SQL and Java
- WorkCount: page 8 – 231—PL/SQL and Java
- getNotifications: page 8 – 232—Java
- getNotificationAttributes: page 8 – 233—Java
- WriteToClob: page 8 – 234—PL/SQL

---

## 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 messageType,  
     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 or web-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 – 200.

## Arguments (input)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>role</b>	The role name assigned as the performer of the notification activity.
<b>msg_type or msgType</b>	The item type associated with the message.
<b>msg_name or messageName</b>	The message internal name.
<b>due_date or dueDate</b>	The date that a response is required. This optional due date is only for the recipient’s information; it has no effect on processing.
<b>callback</b>	The callback function name used for communication of SEND and RESPOND source message attributes.
<b>context</b>	Context information passed to the callback function.
<b>send_comment or sendComment</b>	A comment presented with the message.
<b>priority</b>	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)

<b>command</b>	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.
<b>context</b>	The context passed to <i>SEND( )</i> or <i>SendGroup( )</i> . The format is <i>&lt;itemtype&gt;:&lt;itemkey&gt;:&lt;activityid&gt;</i> .
<b>attr_name</b>	An attribute name to set/get if command is GET or SET.
<b>attr_type</b>	An attribute type if command is SET or GET.
<b>text_value</b>	Value of a text attribute if command is SET or value of text attribute returned if command is GET.
<b>number_value</b>	Value of a number attribute if command is SET or value of a number attribute returned if command is GET.
<b>date_value</b>	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 or web-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 – 200.

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 – 61.

### Arguments (input)

<b>role</b>	The role name assigned as the performer of the notification activity.
<b>msg_type</b>	The item type associated with the message.
<b>msg_name</b>	The message internal name.

<b>due_date</b>	The date that a response is required. This optional due date is only for the recipient's information; it has no effect on processing.
<b>callback</b>	The callback function name used for communication of SEND source message attributes.
<b>context</b>	Context information passed to the callback function.
<b>send_comment</b>	A comment presented with the message.
<b>priority</b>	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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	The notification id.
<b>new_role or newRole</b>	The role name of the person the note is reassigned to.
<b>forward_comment or comment</b>	An optional forwarding comment.

**Example** The following code excerpt shows an example of how to call *forward()* in a Java program. The example code is from the WFTest.java program.

```
// forward to MBEECH
System.out.println("Delegate Test");
count = WFNotificationAPI.workCount(ctx, "MBEECH");
System.out.println("There are " + count +
  " open notification(s) for" + " MBEECH");
System.out.println("Delegate nid " + myNid +
```

```
        " from BLEWIS to MBEECH");
WFNotificationAPI.forward(ctx, myNid, "MBEECH",
        "Matt, Please handle.");
count = WFNotificationAPI.workCount(ctx, "MBEECH");
System.out.println("There are " + count +
        " open notification(s) for" +
        " MBEECH after Delegate.");
```

---

## 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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	The notification id.
<b>new_role or newRole</b>	The role name of the person the note is transferred to.
<b>forward_comment or comment</b>	An optional comment to append to notification.

**Example** The following code excerpt shows an example of how to call *transfer()* in a Java program. The example code is from the WFTest.java program.

```
// transfer to MBEECH
System.out.println("Transfer Test");
System.out.println("Transfer nid " + myNid +
" from BLEWIS to MBEECH");
```

```
WFNotificationAPI.transfer(ctx, myNid, "MBEECH",  
    "Matt, You own it now.");  
count = WFNotificationAPI.workCount(ctx, "MBEECH");  
System.out.println("There are " + count +  
    " open notification(s) for" +  
    p" MBEECH after Transfer.");
```

---

## 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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	The notification id.
<b>cancel_comment or comment</b>	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 – 61.

### Arguments (input)

<b>gid</b>	The notification group id.
<b>cancel_comment</b>	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 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 who 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. The following table shows the value that is stored for each response mechanism.

Response Mechanism	Value Stored
Web	Web login username
E-mail	E-mail username as displayed in the mail response

Table 8 – 6 (Page 1 of 1)

### Arguments (input)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	The notification id
<b>comment</b>	An optional comment on the response
<b>responder</b>	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 – 211.

### Arguments (input)

**wCtx**                    Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.

**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 gid,
     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 – 61.

### Arguments (input)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>gid</b>	The notification group id.
<b>ResultCode</b>	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 – 61.

### Arguments (input)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>gid</b>	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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	The notification id.
<b>responder</b>	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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	The notification id.
<b>aname</b>	The attribute name.
<b>avalue</b>	The attribute value.

**Example**    The following code excerpt shows an example of how to call *addAttr()* in a Java program. The example code is from the *WFTest.java* program.

```
if (WFNotificationAPI.addAttr(ctx, myNid, myAttr) == false)
{
    System.out.println("Add attribute " + myAttr + " failed.");
}
```

---

## 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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	The notification id.
<b>aname</b>	The attribute name.
<b>avalue</b>	The attribute value.

**Example** The following code excerpt shows an example of how to call a *setAttribute* method in a Java program. The example code is from the WFTest.java program.

```
if (WFNotificationAPI.setAttrDate(ctx, myNid, myAttr, value)
    == false)
{
    System.out.println("set attribute " + myAttr + " to " +
        value + " failed.");
}
```

---

## 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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	The notification id.
<b>aname</b>	The attribute name.

**Example** The following code excerpt shows an example of how to call *getAttrInfo()* in a Java program. The example code is from the WFTest.java program.

```
dataSource = WFNotificationAPI.getAttrInfo(ctx, myNid,
    myAttr);
displayDataSource(ctx, dataSource);

// the first element is the attribute type
myAttrType = (String) dataSource.getData(0,0);
```

---

## 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
(
  WFCContext 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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	The notification id.

**Example** The following code excerpt shows an example of how to call *getInfo()* in a Java program. The example code is from the WFTest.java program.

```
// Notification Info
System.out.println("Notification Info for nid " + myNid);
dataSource = WFNotificationAPI.getInfo(ctx, myNid);
displayDataSource(ctx, dataSource);
```



---

## 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)

<b>some_text</b>	Text to be substituted.
<b>nid</b>	Notification ID of notification to use for token values.
<b>disptype</b>	The display type of the message body that you are token substituting the text into. Valid display types are: <ul style="list-style-type: none"><li>• <code>wf_notification.doc_text</code>, which returns <code>text/plain</code></li><li>• <code>wf_notification.doc_html</code>, which returns <code>text/html</code></li><li>• <code>wf_notification.doc_attach</code>, which returns null</li></ul> 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)

<b>some_text</b>	Text to be substituted.
<b>nid</b>	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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	The notification id.
<b>aname</b>	The message attribute name.

**Example** The following code excerpt shows an example of how to call the *getAttribute* methods in a Java program. The example code is from the *WFTTest.java* program.

```
// we get the value according to the type.
if (myAttrType == "DATE")
{
    value = WFNotificationAPI.getAttrDate(ctx, myNid, myAttr);
}
else if (myAttrType == "NUMBER")
{
    value = (WFNotificationAPI.getAttrNumber(ctx, myNid,
        myAttr)).toString();
}
else if (myAttrType == "DOCUMENT")
{
    value = WFNotificationAPI.getAttrDoc(ctx, myNid, myAttr,
        null);
}
else
    value = WFNotificationAPI.getAttrText(ctx, myNid, myAttr);

System.out.println(myAttr.toString() + " = '" + value +
    "'");
```

---

## 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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	The notification id.
<b>aname</b>	The message attribute name.
<b>disptype</b>	The display type of the document you wish to return. Valid display types are: <ul style="list-style-type: none"><li>• wf_notification.doc_text, which returns text/plain</li><li>• wf_notification.doc_html, which returns text/html</li><li>• wf_notification.doc_attach, which returns null</li></ul>

---

## 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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	The notification id.
<b>disptype</b>	The display type of the message body you wish to fetch. Valid display types are: <ul style="list-style-type: none"><li>• wf_notification.doc_text, which returns text/plain</li><li>• wf_notification.doc_html, which returns text/html</li><li>• wf_notification.doc_attach, which returns null</li></ul> 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.

### Arguments (input)

<b>nid</b>	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.

### Arguments (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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>access_str or accessString</b>	The access string, in the format: <i>nid/nkey</i> where <i>nid</i> is the notification ID and <i>nkey</i> 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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>username</b>	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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>itemType</b>	The internal name of the item type.
<b>itemKey</b>	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)

<b>wCtx</b>	Workflow context information. Required for the Java method only. See: Oracle Workflow Context: page 8 – 6.
<b>nid</b>	The notification ID.

**Example**     The following code excerpt shows an example of how to call *getNotificationAttributes()* in a Java program. The example code is from the WFTest.java program.

```
// List available Notification Attributes
System.out.println("List of Attributes for id " + myNid
    ":");
dataSource =
    WFNotificationAPI.getNotificationAttributes(ctx, myNid);
displayDataSource(ctx, dataSource);
```

---

## WriteToClob

**PL/SQL Syntax**    `procedure WriteToClob`  
`(clob_loc in out clob,`  
`msg_string in varchar2);`

**Description**    Appends a character string to the end of a character large object (CLOB). You can use this procedure to help build the CLOB for a PL/SQL CLOB document attribute for a notification.

### Arguments (input)

<b>clob_loc</b>	The CLOB to which the string should be added.
<b>msg_string</b>	A string of character data.

### See Also

To Define a Document Attribute: page 4 – 14

“PL/SQL CLOB” Documents: page 7 – 19

---

## Overview of the Oracle Workflow Business Event System

The Oracle Workflow Business Event System leverages the Oracle Advanced Queuing infrastructure to communicate business events between systems. When a significant business event occurs in an internet or intranet application on a system, it triggers event subscriptions that specify the processing to execute for that event.

Subscriptions can include the following types of processing:

- Executing custom code on the event information
- Sending event information to a workflow process
- Sending event information to named communication points called agents on the local system or external systems

The event information communicated by the Business Event System is called an event message. The event message includes header properties to identify the event as well as event data describing what occurred.

You define events, systems, agents, and subscriptions in the Event Manager. You can also define event activities in the Workflow Builder to include business events in your workflow processes.

### See Also

Managing Business Events: page 13 – 2

Event Activities: page 4 – 54

---

## Business Event System Datatypes

Oracle Workflow uses a number of abstract datatypes (ADTs) to model the structure and behavior of Business Event System data. These datatypes include the following:

- Agent structure: WF\_AGENT\_T
- Parameter structure: WF\_PARAMETER\_T
- Parameter list structure: WF\_PARAMETER\_LIST\_T
- Event message structure: WF\_EVENT\_T

The Business Event System datatypes are created by a script called `wftypes.sql`, which is located in the Oracle Workflow `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 Also

User-Defined Datatypes, *Oracle Concepts*



---

## Agent Structure

Oracle Workflow uses the object type `WF_AGENT_T` to store information about an agent in a form that can be referenced by an event message. The following table lists the attributes of the `WF_AGENT_T` datatype.

Attribute Name	Datatype	Description
NAME	VARCHAR2(30)	The name of the agent.
SYSTEM	VARCHAR2(30)	The system where the agent is located.

Table 8 – 7 (Page 1 of 1)

The `WF_AGENT_T` object type also includes the following methods, which you can use to retrieve and set the values of its attributes.

- `getName`
- `getSystem`
- `setName`
- `setSystem`

---

### `getName`

**PL/SQL Syntax** MEMBER FUNCTION `getName`  
`return varchar2`

**Description** Returns the value of the NAME attribute in a `WF_AGENT_T` object.

---

### `getSystem`

**PL/SQL Syntax** MEMBER FUNCTION `getSystem`  
`return varchar2`

**Description** Returns the value of the SYSTEM attribute in a `WF_AGENT_T` object.



---

## Parameter Structure

Oracle Workflow uses the object type `WF_PARAMETER_T` to store a parameter name and value pair in a form that can be included in an event message parameter list. `WF_PARAMETER_T` allows custom values to be added to the `WF_EVENT_T` event message object. The following table lists the attributes of the `WF_PARAMETER_T` datatype.

Attribute Name	Datatype	Description
NAME	VARCHAR2(30)	The parameter name.
VALUE	VARCHAR2(2000)	The parameter value.

Table 8 – 8 (Page 1 of 1)

The `WF_PARAMETER_T` object type also includes the following methods, which you can use to retrieve and set the values of its attributes.

- `getName`
- `getValue`
- `setName`
- `setValue`

---

### `getName`

**PL/SQL Syntax**    `MEMBER FUNCTION getName`

`return varchar2`

**Description**    Returns the value of the NAME attribute in a `WF_PARAMETER_T` object.

---

### `getValue`

**PL/SQL Syntax**    `MEMBER FUNCTION getValue`

`return varchar2`

**Description** Returns the value of the VALUE attribute in a WF\_PARAMETER\_T object.

---

## setName

**PL/SQL Syntax** MEMBER PROCEDURE setName  
(pName in varchar2)

**Description** Sets the value of the NAME attribute in a WF\_PARAMETER\_T object.

**Arguments (input)**

**pName** The value for the NAME attribute.

---

## setValue

**PL/SQL Syntax** MEMBER PROCEDURE setValue  
(pValue in varchar2)

**Description** Sets the value of the VALUE attribute in a WF\_PARAMETER\_T object.

**Arguments (input)**

**pValue** The value for the VALUE attribute.

---

## Parameter List Structure

Oracle Workflow uses the named varying array (varray) `WF_PARAMETER_LIST_T` to store a list of parameters in a form that can be included in an event message. `WF_PARAMETER_LIST_T` allows custom values to be added to the `WF_EVENT_T` event message object. The `WF_PARAMETER_LIST_T` datatype can include up to 100 parameter name and value pairs. A description of this datatype is as follows:

### **`WF_PARAMETER_LIST_T`**

- Maximum size: 100
- Element datatype: `WF_PARAMETER_T`

## Event Message Structure

Oracle Workflow uses the object type `WF_EVENT_T` to store event messages. This datatype contains all the header properties of an event message as well as the event data payload, in a serialized form that is suitable for transmission outside the system.

`WF_EVENT_T` defines the event message structure that the Business Event System and the Workflow Engine use to represent a business event. Internally, the Business Event System and the Workflow Engine can only communicate events in this format.

The standard queues that Oracle Workflow provides for the Business Event System, called `WF_IN`, `WF_OUT`, `WF_DEFERRED`, and `WF_ERROR`, all use `WF_EVENT_T` as their payload type. If you want to use queues with an alternate payload type, including any existing queues you already have defined on your system, you must create a queue handler to translate between the standard Workflow `WF_EVENT_T` structure and your custom payload type. See: [Setting Up Queues: page 2 – 97](#) and [Standard APIs for a Queue Handler: page 7 – 23](#).

The following table lists the attributes of the `WF_EVENT_T` datatype.

Attribute Name	Datatype	Description
PRIORITY	NUMBER	The priority with which the message recipient should dequeue the message. A smaller number indicates a higher priority. For example, 1 represents a high priority, 50 represents a normal priority, and 99 represents a low priority.
SEND_DATE	DATE	<p>The date and time when the message is available for dequeuing. The send date can be set to the system date to indicate that the message is immediately available for dequeuing, or to a future date to indicate future availability.</p> <p>If the send date is set to a future date when an event is raised, the event message is placed on the <code>WF_DEFERRED</code> queue, and subscription processing does not begin until the specified date. If the send date is set to a future date when an event is sent to an agent, the event message is propagated to that agent's queue, but does not become available for the consumer to dequeue until the specified date.</p>

Attribute Name	Datatype	Description
RECEIVE_DATE	DATE	The date and time when the message is dequeued by an agent listener.
CORRELATION_ID	VARCHAR2(240)	A correlation identifier that associates this message with other messages. This attribute is initially blank but can be set by a function. If a value is set for the correlation ID, then that value is used as the item key if the event is sent to a workflow process.
PARAMETER_LIST	WF_PARAMETER_LIST_T	A list of additional parameter name and value pairs.
EVENT_NAME	VARCHAR2(240)	The internal name of the event.
EVENT_KEY	VARCHAR2(240)	The string that uniquely identifies the instance of the event.
EVENT_DATA	CLOB	A set of additional details describing what occurred in the event. The event data can be structured as an XML document.
FROM_AGENT	WF_AGENT_T	The agent from which the event is sent. For locally raised events, this attribute is initially null.
TO_AGENT	WF_AGENT_T	The agent to which the event should be sent (the message recipient).
ERROR_SUBSCRIPTION	RAW(16)	If an error occurs while processing this event, this is the subscription that was being executed when the error was encountered.
ERROR_MESSAGE	VARCHAR2(4000)	An error message that the Event Manager generates if an error occurs while processing this event.
ERROR_STACK	VARCHAR2(4000)	An error stack of arguments that the Event Manager generates if an error occurs while processing this event. The error stack provides context information to help you locate the source of an error.

Table 8 – 9 (Page 2 of 2)

The WF\_EVENT\_T object type also includes the following methods, which you can use to retrieve and set the values of its attributes.

- Initialize: page 8 – 245
- getPriority: page 8 – 245
- getSendDate: page 8 – 245

- getReceiveDate: page 8 – 246
- getCorrelationID: page 8 – 246
- getParameterList: page 8 – 246
- getEventName: page 8 – 246
- getEventKey: page 8 – 247
- getEventData: page 8 – 247
- getFromAgent: page 8 – 247
- getToAgent: page 8 – 247
- getErrorSubscription: page 8 – 247
- getErrorMessage: page 8 – 248
- getErrorStack: page 8 – 248
- setPriority: page 8 – 248
- setSendDate: page 8 – 248
- setReceiveDate: page 8 – 249
- setCorrelationID: page 8 – 249
- setParameterList: page 8 – 249
- setEventName: page 8 – 250
- setEventKey: page 8 – 250
- setEventData: page 8 – 250
- setFromAgent: page 8 – 251
- setToAgent: page 8 – 251
- setErrorSubscription: page 8 – 251
- setErrorMessage: page 8 – 251
- setErrorStack: page 8 – 252
- Content: page 8 – 252
- Address: page 8 – 253
- AddParameterToList: page 8 – 253
- GetValueForParameter: page 8 – 253

**Note:** You can set the values of the `EVENT_NAME`, `EVENT_KEY`, and `EVENT_DATA` attributes individually using the `setEventName`, `setEventKey`, and `setEventData` methods, or



you can use the Content method to set all three event content attributes at once. See: Content: page 8 – 252.

Similarly, you can set the values of the FROM\_AGENT, TO\_AGENT, PRIORITY, and SEND\_\_DATE attributes individually using the setFromAgent, setToAgent, setPriority, and setSendDate methods, or you can use the Address method to set all four address attributes at once. See: Address: page 8 – 253.

---

## Initialize

**PL/SQL Syntax**    `STATIC PROCEDURE initialize`  
`(new_wf_event_t in out wf_event_t)`

**Description**    Initializes a new WF\_EVENT\_T object by setting the PRIORITY attribute to 0, initializing the EVENT\_DATA attribute to EMPTY using the *Empty\_CLOB()* function, and setting all other attributes to NULL.



**Attention:** You must call the Initialize method before you can perform any further manipulation on a new WF\_EVENT\_T object.

### Arguments (input)

**new\_wf\_event\_t**    The WF\_EVENT\_T object to initialize.

---

## getPriority

**PL/SQL Syntax**    `MEMBER FUNCTION getPriority`  
`return number`

**Description**    Returns the value of the PRIORITY attribute in a WF\_EVENT\_T object.

---

## getSendDate

**PL/SQL Syntax**    `MEMBER FUNCTION getSendDate`  
`return date`

**Description** Returns the value of the SEND\_DATE attribute in a WF\_EVENT\_T object.

---

## getReceiveDate

**PL/SQL Syntax** MEMBER FUNCTION getReceiveDate  
return date

**Description** Returns the value of the RECEIVE\_DATE attribute in a WF\_EVENT\_T object.

---

## getCorrelationID

**PL/SQL Syntax** MEMBER FUNCTION getCorrelationID  
return varchar2

**Description** Returns the value of the CORRELATION\_ID attribute in a WF\_EVENT\_T object.

---

## getParameterList

**PL/SQL Syntax** MEMBER FUNCTION getParameterList  
return wf\_parameter\_list\_t

**Description** Returns the value of the PARAMETER\_LIST attribute in a WF\_EVENT\_T object.

---

## getEventName

**PL/SQL Syntax** MEMBER FUNCTION getEventName  
return varchar2

**Description** Returns the value of the EVENT\_NAME attribute in a WF\_EVENT\_T object.

---

## getEventKey

**PL/SQL Syntax** MEMBER FUNCTION getEventKey  
return varchar2

**Description** Returns the value of the EVENT\_KEY attribute in a WF\_EVENT\_T object.

---

## getEventData

**PL/SQL Syntax** MEMBER FUNCTION getEventData  
return clob

**Description** Returns the value of the EVENT\_DATA attribute in a WF\_EVENT\_T object.

---

## getFromAgent

**PL/SQL Syntax** MEMBER FUNCTION getFromAgent  
return wf\_agent\_t

**Description** Returns the value of the FROM\_AGENT attribute in a WF\_EVENT\_T object.

---

## getToAgent

**PL/SQL Syntax** MEMBER FUNCTION getToAgent  
return wf\_agent\_t

**Description** Returns the value of the TO\_AGENT attribute in a WF\_EVENT\_T object.

---

## getErrorSubscription

**PL/SQL Syntax** MEMBER FUNCTION getErrorSubscription

return raw

**Description** Returns the value of the ERROR\_SUBSCRIPTION attribute in a WF\_EVENT\_T object.

---

## getErrorMessage

**PL/SQL Syntax** MEMBER FUNCTION getErrorMessage  
return varchar2

**Description** Returns the value of the ERROR\_MESSAGE attribute in a WF\_EVENT\_T object.

---

## getErrorStack

**PL/SQL Syntax** MEMBER FUNCTION getErrorStack  
return varchar2

**Description** Returns the value of the ERROR\_STACK attribute in a WF\_EVENT\_T object.

---

## setPriority

**PL/SQL Syntax** MEMBER PROCEDURE setPriority  
(pPriority in number)

**Description** Sets the value of the PRIORITY attribute in a WF\_EVENT\_T object.

### Arguments (input)

**pPriority** The value for the PRIORITY attribute.

---

## setSendDate

**PL/SQL Syntax** MEMBER PROCEDURE setSendDate

(pSendDate in date default sysdate)

**Description** Sets the value of the SEND\_DATE attribute in a WF\_EVENT\_T object.

**Arguments (input)**

**pSendDate** The value for the SEND\_DATE attribute.

---

## setReceiveDate

**PL/SQL Syntax** MEMBER PROCEDURE setReceiveDate  
(pReceiveDate in date default sysdate)

**Description** Sets the value of the RECEIVE\_DATE attribute in a WF\_EVENT\_T object.

**Arguments (input)**

**pReceiveDate** The value for the RECEIVE\_DATE attribute.

---

## setCorrelationID

**PL/SQL Syntax** MEMBER PROCEDURE setCorrelationID  
(pCorrelationID in varchar2)

**Description** Sets the value of the CORRELATION\_ID attribute in a WF\_EVENT\_T object.

**Arguments (input)**

**pCorrelationID** The value for the CORRELATION\_ID attribute.

---

## setParameterList

**PL/SQL Syntax** MEMBER PROCEDURE setParameterList  
(pParameterList in wf\_parameter\_list\_t)

**Description** Sets the value of the PARAMETER\_LIST attribute in a WF\_EVENT\_T object.

### Arguments (input)

**pParameterList** The value for the PARAMETER\_LIST attribute.

---

## setEventName

**PL/SQL Syntax** MEMBER PROCEDURE setEventName  
(pEventName in varchar2)

**Description** Sets the value of the EVENT\_NAME attribute in a WF\_EVENT\_T object.

### Arguments (input)

**pEventName** The value for the EVENT\_NAME attribute.

---

## setEventKey

**PL/SQL Syntax** MEMBER PROCEDURE setEventKey  
(pEventKey in varchar2)

**Description** Sets the value of the EVENT\_KEY attribute in a WF\_EVENT\_T object.

### Arguments (input)

**pEventKey** The value for the EVENT\_KEY attribute.

---

## setEventData

**PL/SQL Syntax** MEMBER PROCEDURE setEventData  
(pEventData in clob)

**Description** Sets the value of the EVENT\_DATA attribute in a WF\_EVENT\_T object.

### Arguments (input)

**pEventData** The value for the EVENT\_DATA attribute.



(pErrorMessage in varchar2)

**Description** Sets the value of the ERROR\_MESSAGE attribute in a WF\_EVENT\_T object.

**Arguments (input)**

**pErrorMessage** The value for the ERROR\_MESSAGE attribute.

---

## setErrorStack

**PL/SQL Syntax** MEMBER PROCEDURE setErrorStack  
(pErrorStack in varchar2)

**Description** Sets the value of the ERROR\_STACK attribute in a WF\_EVENT\_T object.

**Arguments (input)**

**pErrorStack** The value for the ERROR\_STACK attribute.

---

## Content

**PL/SQL Syntax** MEMBER PROCEDURE Content  
(pName in varchar2,  
pKey in varchar2,  
pData in clob)

**Description** Sets the values of all the event content attributes in a WF\_EVENT\_T object, including EVENT\_NAME, EVENT\_KEY, and EVENT\_DATA.

**Arguments (input)**

**pName** The value for the EVENT\_NAME attribute.  
**pKey** The value for the EVENT\_KEY attribute.  
**pData** The value for the EVENT\_DATA attribute.



---

## Address

**PL/SQL Syntax**    MEMBER PROCEDURE Address

```
(pOutAgent in wf_agent_t,  
pToAgent in wf_agent_t,  
pPriority in number,  
pSendDate in date)
```

**Description**    Sets the values of the all address attributes in a WF\_EVENT\_T object, including FROM\_AGENT, TO\_AGENT, PRIORITY, and SEND\_DATE.

### Arguments (input)

<b>pOutAgent</b>	The value for the FROM_AGENT attribute.
<b>pToAgent</b>	The value for the TO_AGENT attribute.
<b>pPriority</b>	The value for the PRIORITY attribute.
<b>pSendDate</b>	The value for the SEND_DATE attribute.

---

## AddParameterToList

**PL/SQL Syntax**    MEMBER PROCEDURE AddParameterToList

```
(pName in varchar2,  
pValue in varchar2)
```

**Description**    Adds a new parameter name and value pair to the list stored in the PARAMETER\_LIST attribute of a WF\_EVENT\_T object. If a parameter with the specified name already exists in the parameter list, then the previous value of that parameter is overwritten with the specified value.

### Arguments (input)

<b>pName</b>	The parameter name.
<b>pValue</b>	The parameter value.

---

## GetValueForParameter

**PL/SQL Syntax**    MEMBER FUNCTION GetValueForParameter

(pName in varchar2) return varchar2

**Description** Returns the value of the specified parameter from the list stored in the PARAMETER\_LIST attribute of a WF\_EVENT\_T object. This method begins at the end of the parameter list and searches backwards through the list. If no parameter with the specified name is found in the parameter list, then the GetValueForParameter method returns NULL.

**Arguments (input)**

**pName** The parameter name.

---

## Example for Using Abstract Datatypes

The following example shows some ways to use abstract datatype methods in a SQL script, including:

- Initializing a new event message structure with the Initialize method



**Attention:** You must call the Initialize method before you can perform any further manipulation on a new WF\_EVENT\_T object.

- Initializing a CLOB locator
- Writing a text variable into a CLOB variable
- Setting the content attributes of the event message structure with the Content method
- Setting the address attributes of the event message structure with the Address method

The example code is from the script wfeventnq.sql, which enqueues an event message on a queue using an override agent. See: Wfeventnq.sql: page 16 – 10.

```
declare
l_overrideagent varchar2(30) := '&overrideagent';
l_overridesystem varchar2(30) := '&overridesystem';
l_fromagent varchar2(30) := '&fromagent';
l_fromsystem varchar2(30) := '&fromsystem';
l_toagent varchar2(30) := '&toagent';
l_tosystem varchar2(30) := '&tosystem';
l_eventname varchar2(100) := '&eventname';
l_eventkey varchar2(100) := '&eventkey';
l_msg varchar2(200) := '&message';
l_clob clob;
l_overrideagent_t wf_agent_t;
l_toagent_t wf_agent_t;
l_fromagent_t wf_agent_t;
l_event_t wf_event_t;

begin

    /*You must call wf_event_t.initialize before you can manipulate
    a new wf_event_t object.*/
    wf_event_t.initialize(l_event_t);
```

```

l_overrideagent_t := wf_agent_t(l_overrideagent,
                               l_overridesystem);
l_toagent_t := wf_agent_t(l_toagent, l_tosystem);
l_fromagent_t := wf_agent_t(l_fromagent, l_fromsystem);

if l_msg is null then
    l_event_t.Content(l_eventname, l_eventkey, null);
else
    dbms_lob.createtemporary(l_clob, FALSE, DBMS_LOB.CALL);
    dbms_lob.write(l_clob, length(l_msg), 1, l_msg);
    l_event_t.Content(l_eventname, l_eventkey, l_clob);
end if;

l_event_t.Address(l_fromagent_t, l_toagent_t, 50, sysdate);

wf_event.enqueue(l_event_t, l_overrideagent_t);

end;
```

## Mapping Between WF\_EVENT\_T and OMBAQ\_TEXT\_MSG

If you use Oracle8i, you can optionally implement Oracle Message Broker (OMB) to propagate event messages between systems. OMB queues require messages to be stored in a structure defined by a Java Message Service abstract datatype called OMBAQ\_TEXT\_MSG.

Oracle Workflow provides a queue handler called WF\_EVENT\_OMB\_QH which you can use to translate between the standard Workflow WF\_EVENT\_T message structure and the OMBAQ\_TEXT\_MSG structure. See: Setting Up the WF\_EVENT\_OMB\_QH Queue Handler: page 2 – 100 and Agents: page 13 – 22.

Among other attributes, the OMBAQ\_TEXT\_MSG datatype contains an attribute called TEXT\_LOB, which contains the message payload in CLOB format, and another attribute called HEADER, whose datatype is another ADT called OMBAQ\_HEADER.

OMBAQ\_HEADER in turn contains an attribute called PROPERTIES, whose datatype is a third ADT, a named varying array called OMBAQ\_PROPERTIES. The maximum size of OMBAQ\_PROPERTIES is 1000, and the datatype of its elements is a fourth ADT, OMBAQ\_PROPERTY.

The following table shows how the attributes of the WF\_EVENT\_T message structure are mapped to the attributes within the OMBAQ\_TEXT\_MSG structure.

WF_EVENT_T	OMBAQ_TEXT_MSG
WF_EVENT_T.PRIORITY	ombaq_properties(5).str_value [name = PRIORITY]
WF_EVENT_T.SEND_DATE	ombaq_properties(6).str_value [name = SENDDATE]
WF_EVENT_T.RECEIVE_DATE	ombaq_properties(7).str_value [name = RECEIVEDATE]
WF_EVENT_T.CORRELATION_ID	ombaq_properties(8).str_value [name = CORRELATIONID]
WF_EVENT_T.EVENT_NAME	ombaq_properties(9).str_value [name = EVENTNAME]
WF_EVENT_T.EVENT_KEY	ombaq_properties(10).str_value [name = EVENTKEY]

Table 8 – 10 (Page 1 of 2)

WF_EVENT_T	OMBAQ_TEXT_MSG
WF_EVENT_T.EVENT_DATA	text_lob (CLOB)
WF_EVENT_T.FROM_AGENT.NAME	ombaq_properties(1).str_value [name = FROMAGENTNAME]
WF_EVENT_T.FROM_AGENT.SYSTEM	ombaq_properties(2).str_value [name = FROMAGENTSYSTEM]
WF_EVENT_T.TO_AGENT.NAME	ombaq_properties(3).str_value [name = TOAGENTNAME]
WF_EVENT_T.TO_AGENT.SYSTEM	ombaq_properties(4).str_value [name = TOAGENTSYSTEM]
WF_EVENT_T.ERROR_SUBSCRIPTION	ombaq_properties(11).str_value [name = ERRORSUBSCRIPTION]
WF_EVENT_T.ERROR_MESSAGE	ombaq_properties(12).str_value [name = ERRORMESSAGE1]
WF_EVENT_T.ERROR_MESSAGE	ombaq_properties(13).str_value [name = ERRORMESSAGE2]
WF_EVENT_T.ERROR_STACK	ombaq_properties(14).str_value [name = ERRORSTACK1]
WF_EVENT_T.ERROR_STACK	ombaq_properties(15).str_value [name = ERRORSTACK2]
WF_EVENT_T.PARAMETER_LIST	ombaq_properties(16).str_value [name = <first_parameter_name>]
...	...
WF_EVENT_T.PARAMETER_LIST	ombaq_properties(115).str_value [name = <hundredth_parameter_name>]

Table 8 – 10 (Page 2 of 2)

**Note:** You can use any names you choose for the parameters in the parameter list for an event, except the reserved words that are used for the other event properties. The reserved words are:

PRIORITY, SENDDATE, RECEIVEDATE, CORRELATIONID,  
EVENTNAME, EVENTKEY, FROMAGENTNAME,  
FROMAGENTSYSTEM, TOAGENTNAME, TOAGENTSYSTEM,  
ERRORSUBSCRIPTION, ERRORMESSAGE1, ERRORMESSAGE2,  
ERRORSTACK1, ERRORSTACK2

**Note:** Oracle Message Broker and the OMBAQ\_TEXT\_MSG datatype are no longer used in Oracle9i. In Oracle9i, you can use the Messaging Gateway and Internet access features of Oracle Advanced Queuing to propagate event messages, in place of Oracle Message Broker.

## See Also

Oracle AQ Driver ADTs, *Oracle Message Broker Administration Guide*

---

## Event APIs

The Event APIs can be called by an application program or a workflow process in the runtime phase to communicate with the Business Event System and manage events. These APIs are defined in a PL/SQL package called WF\_EVENT.

- Raise: page 8 – 261
- Send: page 8 – 265
- NewAgent: page 8 – 267
- Test: page 8 – 268
- Enqueue: page 8 – 269
- Listen: page 8 – 270

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications, you can also use the "Workflow Agent Listener" concurrent program to listen for inbound event messages. See: Workflow Agent Listener: page 8 – 272.

- SetErrorInfo: page 8 – 273
- SetDispatchMode: page 8 – 274
- AddParameterToList: page 8 – 275
- AddParameterToListPos: page 8 – 276
- GetValueForParameter: page 8 – 277
- GetValueForParameterPos: page 8 – 278



---

## Raise

**PL/SQL Syntax**    procedure Raise

```
(p_event_name in varchar2,  
 p_event_key in varchar2,  
 p_event_data in clob default NULL,  
 p_parameters in wf_parameter_list_t default NULL,  
 p_send_date in date default NULL);
```

**Description**    Raises a local event to the Event Manager. *Raise()* creates a WF\_EVENT\_T structure for this event instance and sets the specified event name, event key, event data, parameter list, and send date into the structure.

The event data can be passed to the Event Manager within the call to the *Raise()* API, or the Event Manager can obtain the event data itself by calling the Generate function for the event, after first checking whether the event data is required by a subscription. If the event data is not already available in your application, you can improve performance by allowing the Event Manager to run the Generate function and generate the event data only when subscriptions exist that require that data, rather than always generating the event data from your application at runtime. See: Events: page 13 – 4 and Standard API for an Event Data Generate Function: page 7 – 21.

The send date can optionally be set to indicate when the event should become available for subscription processing. If the send date is null, *Raise()* sets the send date to the current system date. You can defer an event by setting the send date to a date later than the system date. In this case, the Event Manager places the event message on the standard WF\_DEFERRED queue, where it remains in a WAIT state until the send date. When the send date arrives, the event message becomes available for dequeuing and will be dequeued the next time an agent listener runs on the WF\_DEFERRED queue.

**Note:** If an event is deferred when it is raised, the event retains its original Local source type when it is dequeued from the WF\_DEFERRED queue.

When an event is raised and is not deferred, or when an event that was deferred is dequeued from the WF\_DEFERRED queue, the Event Manager begins subscription processing for the event. The Event Manager searches for and executes any active subscriptions by the local system to that event with a source type of Local, and also any active subscriptions by the local system to the Any event with a source type of Local. If no active subscriptions exist for the event that was raised

(apart from subscriptions to the Any event), then Oracle Workflow executes any active subscriptions by the local system to the Unexpected event with a source type of Local.

**Note:** The Event Manager does not raise an error if the event is not defined.

The Event Manager checks each subscription before executing it to determine whether the subscription requires the event data. If the event data is required but is not already provided, the Event Manager calls the Generate function for the event to produce the event data. If the event data is required but no Generate function is defined for the event, Oracle Workflow creates a default set of event data using the event name and event key.

**Note:** Any exceptions raised during *Raise()* processing are not trapped, but instead are exposed to the code that called the *Raise()* procedure. This behavior enables you to use subscriptions and their rule functions to perform validation, with the same results as if the validation logic were coded inline.

## Arguments (input)

<b>p_event_name</b>	The internal name of the event.
<b>p_event_key</b>	A string generated when the event occurs within a program or application. The event key uniquely identifies a specific instance of the event.
<b>p_event_data</b>	An optional set of information about the event that describes what occurred. The Event Manager checks each subscription before executing it to determine whether the subscription requires the event data. If the event data is required but is not already provided, the Event Manager calls the Generate function for the event to produce the event data. See: Events: page 13 – 4 and Standard API for an Event Data Generate Function: page 7 – 21.
<b>p_parameters</b>	An optional list of additional parameter name and value pairs.
<b>p_send_date</b>	An optional date to indicate when the event should become available for subscription processing.

**Example**

```
declare
    l_xmldocument varchar2(32000);
```

```

        l_eventdata clob;
        l_parameter_list wf_parameter_list_t;
        l_message varchar2(10);

begin

    /*
    ** If the complete event data is easily available, we can
    ** optionally test if any subscriptions to this event
    ** require it (rule data = Message).
    */

        l_message := wf_event.test('<EVENT_NAME>');

    /*
    ** If we do require a message, and we have the message now,
    ** set it; else we can just rely on the Event Generate
    ** Function callback code. Then Raise the Event with the
    ** required parameters.
    */

    if l_message = 'MESSAGE' then
        if l_xmldocument is not null then
            dbms_lob.createtemporary(l_eventdata, FALSE,
                DBMS_LOB.CALL);
            dbms_lob.write(l_eventdata, length(l_xmldocument), 1 ,
                l_xmldocument);
            -- Raise the Event with the message
            wf_event.raise( p_event_name => '<EVENT_NAME>',
                p_event_key   => '<EVENT_KEY>',
                p_event_data => l_eventdata,
                p_parameters => l_parameter_list);
        else
            -- Raise the Event without the message
            wf_event.raise( p_event_name => '<EVENT_NAME>',
                p_event_key   => '<EVENT_KEY>',
                p_parameters => l_parameter_list);
        end if;
    elsif
        l_message = 'KEY' then
        -- Raise the Event
        wf_event.raise( p_event_name => <EVENT_NAME>,
            p_event_key   => <EVENT_KEY>,
            p_parameters => l_parameter_list);
    end if;
end if;

```

```
        end if;

    /*
    ** Up to your own custom code to commit the transaction
    */

        commit;

    /*
    ** Up to your own custom code to handle any major exceptions
    */

    exception
    when others then
    null;
    end;
```

## See Also

Any Event: page 14 – 10

Unexpected Event: page 14 – 12

---

## Send

**PL/SQL Syntax**      procedure Send  
                              (p\_event in out wf\_event\_t);

**Description**      Sends an event message from one agent to another. If the event message contains both a From Agent and a To Agent, the message is placed on the outbound queue of the From Agent and then asynchronously delivered to the To Agent by AQ propagation, or whichever type of propagation is implemented for the agents' protocol.

If the event message contains a To Agent but no specified From Agent, the message is sent from the default outbound agent that matches the queue type of the To Agent.

If the event message contains a From Agent but no specified To Agent, the event message is placed on the From Agent's queue without a specified recipient.

- You can omit the To Agent if the From Agent uses a multi-consumer queue with a subscriber list. (The standard Workflow queue handlers work only with multi-consumer queues.) In this case, the queue's subscriber list determines which consumers can dequeue the message. If no subscriber list is defined for that queue, however, the event message is placed on the WF\_ERROR queue for error handling.

**Note:** The subscriber list for a multi-consumer queue in Oracle Advanced Queuing is different from event subscriptions in the Oracle Workflow Business Event System. For more information, see: *Subscription and Recipient Lists, Oracle Application Developer's Guide – Advanced Queuing*.

- You can also omit the To Agent if the From Agent uses a single-consumer queue for which you have defined a custom queue handler. For a single-consumer queue, no specified consumer is required.

The send date within the event message indicates when the message should become available for the consumer to dequeue. If the send date is blank, the *Send()* procedure resets the value to the current system date, meaning the message is immediately available for dequeuing as soon as it is propagated. If the send date is a future date, the message is marked with a delay time corresponding to that date and does not become available for dequeuing until the delay time has passed. For more information, see: *Time Specification: Delay, Oracle Application Developer's Guide – Advanced Queuing*.

**Note:** If you want to use the send date to determine when a message becomes available for dequeuing on the To Agent, you should set the send date during subscription processing before *Send()* is called.

*Send()* returns the final event message that was sent, including any properties set by the procedure.

### Arguments (input)

<b>p_event</b>	The event message.
----------------	--------------------

---

## NewAgent

**PL/SQL Syntax**

```
function NewAgent  
    (p_agent_guid in raw) return wf_agent_t;
```

**Description** Creates a WF\_AGENT\_T structure for the specified agent and sets the agent's system and name into the structure. See: Agent Structure: page 8 – 237.

### Arguments (input)

**p\_agent\_guid** The globally unique identifier of the agent.

---

## Test

### PL/SQL Syntax

```
function Test  
    (p_event_name in varchar2) return varchar2;
```

### Description

Tests whether the specified event is enabled and whether there are any enabled subscriptions by the local system referencing the event, or referencing an enabled event group that contains the event. *Test()* returns the most costly data requirement among these subscriptions, using the following result codes:

- NONE—No enabled local subscriptions reference the event, or the event does not exist.
- KEY—At least one enabled local subscription references the event, but all such subscriptions require only the event key.
- MESSAGE—At least one enabled local subscription on the event requires the complete event data.

### Arguments (input)

**p\_event\_name**      The internal name of the event.







Wfagtlst.sql: page 16 – 6

Standard APIs for a Queue Handler: page 7 – 23

---

## Workflow Agent Listener Concurrent Program

If you are using the version of Oracle Workflow embedded in Oracle Applications, you can submit the Listen procedure as a concurrent program to listen for inbound event messages. Use the Submit Requests form in Oracle Applications to submit the Workflow Agent Listener.

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager to submit and manage the Workflow Agent Listener concurrent program. For more information, please refer to the Oracle Applications Manager online help.

### ► To Listen for Inbound Event Messages

1. Navigate to the Submit Requests form in Oracle Applications to submit the Workflow Agent Listener 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 "Workflow Agent Listener" and its short name is FNDWFLST. See: *Overview of Concurrent Programs and Requests, Oracle Applications System Administrator's Guide*.
2. Submit the Workflow Agent Listener concurrent program as a request. See: *Submitting a Request, Oracle Applications User's Guide*.
3. In the Parameters window, enter the name of the agent that you want to monitor for inbound event messages.
4. Choose OK to close the Parameters window.
5. When you finish modifying the print and run options to define the schedule for this request, choose Submit to submit the request.

### See Also

Listen: page 8 – 270

---

## SetErrorInfo

**PL/SQL Syntax**    `procedure SetErrorInfo`  
                          `(p_event in out wf_event_t,`  
                          `p_type in varchar2);`

**Description**    Retrieves error information from the error stack and sets it into the event message. The error message and error stack are set into the corresponding attributes of the event message. The error name and error type are added to the PARAMETER\_LIST attribute of the event message.

### Arguments (input)

<b>p_event</b>	The event message.
<b>p_type</b>	The error type, either 'ERROR' or 'WARNING'.



---

## AddParameterToList

**PL/SQL Syntax**

```
procedure AddParameterToList  
    (p_name in varchar2,  
    p_value in varchar2,  
    p_parameterlist in out wf_parameter_list_t);
```

**Description** Adds the specified parameter name and value pair to the end of the specified parameter list varray. If the varray is null, *AddParameterToList()* initializes it with the new parameter.

### Arguments (input)

<b>p_name</b>	The parameter name.
<b>p_value</b>	The parameter value.
<b>p_parameterlist</b>	The parameter list.

---

## AddParameterToListPos

**PL/SQL Syntax**    `procedure AddParameterToListPos`  
                          `(p_name in varchar2,`  
                          `p_value in varchar2,`  
                          `p_position out integer,`  
                          `p_parameterlist in out wf_parameter_list_t);`

**Description**    Adds the specified parameter name and value pair to the end of the specified parameter list varray. If the varray is null, *AddParameterToListPos()* initializes it with the new parameter. The procedure also returns the index for the position at which the parameter is stored within the varray.

### Arguments (input)

<b>p_name</b>	The parameter name.
<b>p_value</b>	The parameter value.
<b>p_parameterlist</b>	The parameter list.



---

## GetValueForParameter

**PL/SQL Syntax**

```
function GetValueForParameter
    (p_name in varchar2,
     p_parameterlist in wf_parameter_list_t)
    return varchar2;
```

**Description** Retrieves the value of the specified parameter from the specified parameter list varray. *GetValueForParameter()* begins at the end of the parameter list and searches backwards through the list.

### Arguments (input)

<b>p_name</b>	The parameter name.
<b>p_parameterlist</b>	The parameter list.

---

## GetValueForParameterPos

**PL/SQL Syntax**

```
function GetValueForParameterPos
    (p_position in integer,
    p_parameterlist in wf_parameter_list_t)
    return varchar2;
```

**Description** Retrieves the value of the parameter stored at the specified position in the specified parameter list varray.

### Arguments (input)

<b>p_position</b>	The index representing the position of the parameter within the parameter list.
<b>p_parameterlist</b>	The parameter list.

---

## Event Subscription Rule Function APIs

The event subscription rule function APIs provide standard rule functions that you can assign to event subscriptions. A rule function specifies the processing that Oracle Workflow performs when the subscription's triggering event occurs.

Oracle Workflow provides a standard `Default_Rule` function to perform basic subscription processing. This function is executed by default if no other rule function is specified for the subscription. The default rule function includes the following actions:

- Sending the event message to a workflow process, if specified in the subscription definition
- Sending the event message to an agent, if specified in the subscription definition

Oracle Workflow also provides some other standard rule functions that you can use. The `Log`, `Error`, `Warning`, and `Success` functions can be used for testing and debugging your application. The `Workflow_Protocol` function passes the event message to a workflow process to be sent to an agent. The `Error_Rule` function performs the same processing as the `Default_Rule` function, but also raises an exception. The `Workflow_Protocol` and `Error_Rule` functions are used in predefined Oracle Workflow event subscriptions. The `SetParametersIntoParameterList` function adds the subscription parameters to the parameter list within the event message.

These rule function APIs are defined in a PL/SQL package called `WF_RULE`.

- `Default_Rule`: page 8 – 281
- `Log`: page 8 – 283
- `Error`: page 8 – 284
- `Warning`: page 8 – 285
- `Success`: page 8 – 286
- `Workflow_Protocol`: page 8 – 287
- `Error_Rule`: page 8 – 288
- `SetParametersIntoParameterList`: page 8 – 289

### See Also

Event Subscriptions: page 13 – 34



---

## Default\_Rule

**PL/SQL Syntax**

```
function Default_Rule
    (p_subscription_guid in raw,
     p_event in out wf_event_t) return varchar2;
```

**Description** Performs default subscription processing when no rule function is specified for an event subscription. The default processing includes:

- Sending the event message to a workflow process, if specified in the subscription definition
- Sending the event message to an agent, if specified in the subscription definition

If either of these operations raises an exception, *Default\_Rule()* traps the exception, stores the error information in the event message, and returns the status code ERROR. Otherwise, *Default\_Rule()* returns the status code SUCCESS.

**Note:** If the event message is being sent to the Default Event Error workflow process, *Default\_Rule()* generates a new correlation ID to use as the item key for the process in order to ensure that the item key is unique.

If you want to run a custom rule function on the event message before it is sent, you can define one subscription with a low phase number that uses the custom rule function, and then define another subscription with a higher phase number that uses the default rule function to send the event.

For example, follow these steps:

1. Define a subscription to the relevant event with your custom rule function and a phase of 10.
2. Define another subscription to the event with the rule function *WF\_EVENT.Default\_Rule* and a phase of 20, and specify the workflow or agent to which you want to send the event.
3. Raise the event to trigger the subscriptions. The subscription with the lower phase number will be executed first and will run your custom rule function on the event message. When the event is passed to the second subscription, the modified event message will be sent to the workflow or agent you specified.

You can also call *Default\_Rule()* to add the default send processing within a custom rule function. If you enter a rule function other than *Default\_Rule()* for a subscription, Oracle Workflow does not automatically send the event message to the workflow and agent

specified in the subscription. Instead, if you want to send the message from the same subscription, you must explicitly include the send processing in your custom rule function, which you can optionally accomplish by calling *Default\_Rule()*. See: Standard API for an Event Subscription Rule Function: page 7 – 25.

**Note:** You may find it advantageous to define multiple subscriptions to an event with simple rule functions that you can reuse, rather than creating complex specialized rule functions that cannot be reused.

### Arguments (input)

<b>p_subscription_guid</b>	The globally unique identifier of the subscription.
<b>p_event</b>	The event message.

---

## Log

**PL/SQL Syntax**

```
function Log
    (p_subscription_guid in raw,
     p_event in out wf_event_t) return varchar2;
```

**Description** Logs the contents of the specified event message using *DBMS\_OUTPUT.put\_line* and returns the status code *SUCCESS*. Use this function to output the contents of an event message to a SQL\*Plus session for testing and debugging purposes.

For example, if you want to test a custom rule function that modifies the event message, you can use *Log()* to show the event message both before and after your custom rule function is run. Define three subscriptions to the relevant event as follows:

- Define the first subscription with a phase of 10 and the rule function *WF\_RULE.Log*.
- Define the second subscription with a phase of 20 and your custom rule function.
- Define the third subscription with a phase of 30 and the rule function *WF\_RULE.Log*.

Next, connect to SQL\*Plus. Execute the following command:

```
set serveroutput on size 100000
```

Then raise the event using *WF\_EVENT.Raise*. As the Event Manager executes your subscriptions to the event in phase order, you should see the contents of the event message both before and after your custom rule function is run.

**Note:** You should not assign *Log()* as the rule function for any enabled subscriptions in a production instance of Oracle Workflow. This function should be used for debugging only.

### Arguments (input)

<b>p_subscription_guid</b>	The globally unique identifier of the subscription.
<b>p_event</b>	The event message.

---

## Error

**PL/SQL Syntax**     `function Error`

```
(p_subscription_guid in raw,  
 p_event in out wf_event_t) return varchar2;
```

**Description**     Returns the status code ERROR. Additionally, when you assign this function as the rule function for a subscription, you must enter a text string representing the internal name of an error message in the Parameters field for the subscription. When the subscription is executed, *Error()* will set that error message into the event message using *setErrorMessage()*. See: *setErrorMessage*: page 8 – 251.

The text string you enter in the Parameters field must be a valid name of an Oracle Workflow error message. The names of the error messages provided by Oracle Workflow are stored in the NAME column of the WF\_RESOURCES table for messages with a type of WFERR.

You can use *Error()* as a subscription rule function if you want to send the system administrator an error notification with one of the predefined Workflow error messages whenever a particular event is raised.

For example, define a subscription to the relevant event with the rule function *WF\_RULE.Error* and enter *WFSQL\_ARGS* in the Parameters field. Then raise the event to trigger the subscription. Because *Error()* returns the status code ERROR, the Event Manager places the event message on the WF\_ERROR queue and subscription processing for the event is halted. When the listener runs on the WF\_ERROR queue, an error notification will be sent to the system administrator with the message "Invalid value(s) passed for arguments," which is the display name of the *WFSQL\_ARGS* error message.

**Note:** *Error()* does not raise any exception to the calling application when it completes normally.

### Arguments (input)

<b>p_subscription_guid</b>	The globally unique identifier of the subscription.
<b>p_event</b>	The event message.



---

## Warning

**PL/SQL Syntax**     `function Warning`  
`(p_subscription_guid in raw,`  
`p_event in out wf_event_t) return varchar2;`

**Description**     Returns the status code WARNING. Additionally, when you assign this function as the rule function for a subscription, you must enter a text string representing the internal name of an error message in the Parameters field for the subscription. When the subscription is executed, *Warning()* will set that error message into the event message using *setErrorMessage()*. See: *setErrorMessage*: page 8 – 251.

The text string you enter in the Parameters field must be a valid name of an Oracle Workflow error message. The names of the error messages provided by Oracle Workflow are stored in the NAME column of the WF\_RESOURCES table for messages with a type of WFERR.

You can use *Warning()* as a subscription rule function if you want to send the system administrator a warning notification with one of the predefined Workflow error messages whenever a particular event is raised.

For example, define a subscription to the relevant event with the rule function *WF\_RULE.Warning* and enter *WFSQL\_ARGS* in the Parameters field. Then raise the event to trigger the subscription. Because *Warning()* returns the status code WARNING, the Event Manager places the event message on the WF\_ERROR queue, but subscription processing for the event still continues. When the listener runs on the WF\_ERROR queue, a warning notification will be sent to the system administrator with the message "Invalid value(s) passed for arguments," which is the display name of the *WFSQL\_ARGS* error message.

**Note:** *Warning()* does not raise any exception to the calling application when it completes normally.

### Arguments (input)

<b>p_subscription_guid</b>	The globally unique identifier of the subscription.
<b>p_event</b>	The event message.

---

## Success

**PL/SQL Syntax**     `function Success`  
`(p_subscription_guid in raw,`  
`p_event in out wf_event_t) return varchar2;`

**Description**     Returns the status code SUCCESS. This function removes the event message from the queue but executes no other code except returning the SUCCESS status code to the calling subscription.

You can use Success for testing and debugging purposes while developing code for use with the Business Event System. For example, if you are trying to debug multiple subscriptions to the same event, you can modify one of the subscriptions by replacing its rule function with *WF\_RULE.Success*, leaving all other details for the subscription intact. When the subscription is executed, it will return SUCCESS but not perform any other subscription processing. This method can help you isolate a problem subscription.

*Success()* is analogous to the WF\_STANDARD.Noop procedure used in the standard Noop activity.

### Arguments (input)

<b>p_subscription_guid</b>	The globally unique identifier of the subscription.
<b>p_event</b>	The event message.

---

## Workflow\_Protocol

**PL/SQL Syntax**

```
function Workflow_Protocol
    (p_subscription_guid in raw,
     p_event in out wf_event_t) return varchar2;
```

**Description** Sends the event message to the workflow process specified in the subscription, which will in turn send the event message to the inbound agent specified in the subscription.

**Note:** *Workflow\_Protocol()* does not itself send the event message to the inbound agent. This function only sends the event message to the workflow process, where you can model the processing that you want to send the event message on to the specified agent.

If the subscription also specifies an outbound agent, the workflow process places the event message on that agent's queue for propagation to the inbound agent. Otherwise, a default outbound agent will be selected.

If the subscription parameters include the parameter name and value pair ACKREQ=Y, then the workflow process waits to receive an acknowledgement after sending the event message.

If the workflow process raises an exception, *Workflow\_Protocol()* stores the error information in the event message and returns the status code ERROR. Otherwise, *Workflow\_Protocol()* returns the status code SUCCESS.

*Workflow\_Protocol()* is used as the rule function in several predefined subscriptions to Workflow Send Protocol and Event System Demonstration events. See: Workflow Send Protocol: page 14 – 17 and Event System Demonstration: page 15 – 63.

### Arguments (input)

<b>p_subscription_guid</b>	The globally unique identifier of the subscription.
<b>p_event</b>	The event message.

---

## Error\_Rule

**PL/SQL Syntax**

```
function Error_Rule
    (p_subscription_guid in raw,
     p_event in out wf_event_t) return varchar2;
```

**Description** Performs the same subscription processing as *Default\_Rule()*, including:

- Sending the event message to a workflow process, if specified in the subscription definition
- Sending the event message to an agent, if specified in the subscription definition

However, if either of these operations encounters an exception, *Error\_Rule()* reraises the exception so that the event is not placed back onto the WF\_ERROR queue. Otherwise, *Error\_Rule()* returns the status code SUCCESS.

*Error\_Rule()* is used as the rule function for the predefined subscriptions to the Unexpected event and to the Any event with the Error source type. The predefined subscriptions specify that the event should be sent to the Default Event Error process in the System: Error item type.

You can also use this rule function with your own error subscriptions. Enter *WF\_RULE.Error* as the rule function for your error subscription and specify the workflow item type and process that you want the subscription to launch.

### Arguments (input)

<b>p_subscription_guid</b>	The globally unique identifier of the subscription.
<b>p_event</b>	The event message.

### See Also

Unexpected Event: page 14 – 12

Any Event: page 14 – 10

---

## SetParametersIntoParameterList

**PL/SQL Syntax**

```
function SetParametersIntoParameterList
    (p_subscription_guid in raw,
     p_event in out wf_event_t) return varchar2;
```

**Description** Sets the parameter name and value pairs from the subscription parameters into the PARAMETER\_LIST attribute of the event message, except for any parameter named ITEMKEY or CORRELATION\_ID. For a parameter with one of these names, the function sets the CORRELATION\_ID attribute of the event message to the parameter value.

If these operations raise an exception, *SetParametersIntoParameterList()* stores the error information in the event message and returns the status code ERROR. Otherwise, *SetParametersIntoParameterList()* returns the status code SUCCESS.

You can use *SetParametersIntoParameterList()* as the rule function for a subscription with a lower phase number, to add predefined parameters from the subscription into the event message. Then subsequent subscriptions with higher phase numbers can access those parameters within the event message.

### Arguments (input)

<b>p_subscription_guid</b>	The globally unique identifier of the subscription.
<b>p_event</b>	The event message.

### See Also

Event Message Structure: page 8 – 242

---

## Event Function APIs

The Event Function APIs provide utility functions that can be called by an application program, the Event Manager, or a workflow process in the runtime phase to communicate with the Business Event System and manage events. These APIs are defined in a PL/SQL package called `WF_EVENT_FUNCTIONS_PKG`.

- Parameters: page 8 – 291
- SubscriptionParameters: page 8 – 293
- AddCorrelation: page 8 – 294
- Generate: page 8 – 296
- Receive: page 8 – 298

---

## Parameters

**PL/SQL Syntax**

```
function Parameters
    (p_string in varchar2,
     p_numvalues in number,
     p_separator in varchar2) return t_parameters;
```

**Description** Parses a string of text that contains the specified number of parameters delimited by the specified separator. *Parameters()* returns the parsed parameters in a varray using the T\_PARAMETERS composite datatype, which is defined in the WF\_EVENT\_FUNCTIONS\_PKG package. The following table describes the T\_PARAMETERS datatype:

Datatype Name	Element Datatype Definition
T_PARAMETERS	VARCHAR2(240)

Table 8 – 11 (Page 1 of 1)

*Parameters()* is a generic utility that you can call in Generate functions when the event key is a concatenation of values separated by a known character. Use this function to separate the event key into its component values.

### Arguments (input)

<b>p_string</b>	A text string containing concatenated parameters.
<b>p_numvalues</b>	The number of parameters contained in the string.
<b>p_separator</b>	The separator used to delimit the parameters in the string.

### Example

```
set serveroutput on

declare
l_parameters wf_event_functions_pkg.t_parameters;
begin
-- Initialize the datatype
l_parameters := wf_event_functions_pkg.t_parameters(1,2);

l_parameters :=
wf_event_functions_pkg.parameters('1111/2222',2,'/');
dbms_output.put_line('Value 1:' || l_parameters(1));
dbms_output.put_line('Value 2:' || l_parameters(2));
```

```
end;  
/
```



---

## SubscriptionParameters

**PL/SQL Syntax**

```
function SubscriptionParameters
    (p_string in varchar2,
     p_key in varchar2) return varchar2;
```

**Description** Returns the value for the specified parameter from a text string containing the parameters defined for an event subscription. The parameter name and value pairs in the text string should be separated by spaces and should appear in the following format:

```
<name1>=<value1> <name2>=<value2> ... <nameN>=<valueN>
```

SubscriptionParameters() searches the text string for the specified parameter name and returns the value assigned to that name. For instance, you can call this function in a subscription rule function to retrieve the value of a subscription parameter, and then code different behavior for the rule function based on that value.

### Arguments (input)

<b>p_string</b>	A text string containing the parameters defined for an event subscription.
<b>p_key</b>	The name of the parameter whose value should be returned.

**Example** In the following example, *SubscriptionParameters()* is used to assign the value of the ITEMKEY subscription parameter to the `l_function` program variable. The example code is from the `AddCorrelation` function, which adds a correlation ID to an event message during subscription processing. See: `AddCorrelation`: page 8 – 294.

```
...
--
-- This is where we will do some logic to determine
-- if there is a parameter
--
l_function := wf_event_functions_pkg.SubscriptionParameters
    (l_parameters, 'ITEMKEY');
...

```

---

## AddCorrelation

**PL/SQL Syntax**

```
function AddCorrelation
    (p_subscription_guid in raw,
     p_event in out wf_event_t) return varchar2;
```

**Description** Adds a correlation ID to an event message during subscription processing. *AddCorrelation()* searches the subscription parameters for a parameter named ITEMKEY that specifies a custom function to generate a correlation ID for the event message. The function must be specified in the Parameters field for the subscription in the following format:

```
ITEMKEY=<package_name.function_name>
```

*AddCorrelation()* uses *SubscriptionParameters()* to search for and retrieve the value of the ITEMKEY parameter. See: *SubscriptionParameters*: page 8 – 293.

If a custom correlation ID function is specified with the ITEMKEY parameter, then *AddCorrelation()* runs that function and sets the correlation ID to the value returned by the function. Otherwise, *AddCorrelation()* sets the correlation ID to the system date. If the event message is then sent to a workflow process, the Workflow Engine uses that correlation ID as the item key to identify the process instance.

If *AddCorrelation()* encounters an exception, the function returns the status code ERROR. Otherwise, *AddCorrelation()* returns the status code SUCCESS.

*AddCorrelation()* is defined according the standard API for an event subscription rule function. You can use *AddCorrelation()* as the rule function for a subscription with a low phase number to add a correlation ID to an event, and then use a subscription with a higher phase number to perform any further processing.

For example, follow these steps:

1. Define a subscription to the relevant event with the rule function *WF\_EVENT\_FUNCTIONS\_PKG.AddCorrelation* and a phase of 10. Enter the parameter name and value pair *ITEMKEY=<package\_name.function\_name>* in the Parameters field for the subscription, replacing *<package\_name.function\_name>* with the package and function that will generate the correlation ID.
2. Define another subscription to the event with a phase of 20, and specify the processing you want to perform by entering a custom rule function or a workflow item type and process. or both.

3. Raise the event to trigger the subscriptions. The subscription with the lower phase number will be executed first and will add a correlation ID to the event message. When the event is passed to the second subscription, that correlation ID will be used as the item key.

You can also call *AddCorrelation()* within a custom rule function to add a correlation ID during your custom processing. See: Standard API for an Event Subscription Rule Function: page 7 – 25.

**Note:** You may find it advantageous to define multiple subscriptions to an event with simple rule functions that you can reuse, rather than creating complex specialized rule functions that cannot be reused.

### Arguments (input)

<b>p_subscription_guid</b>	The globally unique identifier of the subscription.
<b>p_event</b>	The event message.

---

## Generate

**PL/SQL Syntax**     `function Generate`  
`(p_event_name in varchar2,`  
`p_event_key in varchar2) return clob;`

**Description**     Generates the event data for events in the Seed event group. This event data contains Business Event System object definitions which can be used to replicate the objects from one system to another.

The Seed event group includes the following events:

- oracle.apps.wf.event.event.create
- oracle.apps.wf.event.event.update
- oracle.apps.wf.event.event.delete
- oracle.apps.wf.event.group.create
- oracle.apps.wf.event.group.update
- oracle.apps.wf.event.group.delete
- oracle.apps.wf.event.system.create
- oracle.apps.wf.event.system.update
- oracle.apps.wf.event.system.delete
- oracle.apps.wf.event.agent.create
- oracle.apps.wf.event.agent.update
- oracle.apps.wf.event.agent.delete
- oracle.apps.wf.event.subscription.create
- oracle.apps.wf.event.subscription.update
- oracle.apps.wf.event.subscription.delete
- oracle.apps.wf.event.all.sync

For the event, event group, system, agent, and subscription definition events, *WF\_EVENT\_FUNCTIONS\_PKG.Generate()* calls the Generate APIs associated with the corresponding tables to produce the event data XML document. For the Synchronize Event Systems event, *WF\_EVENT\_FUNCTIONS\_PKG.Generate()* produces an XML document containing all the event, event group, system, agent, and subscription definitions from the Event Manager on the local system.

## Arguments (input)

<b>p_event_name</b>	The internal name of the event.
<b>p_event_key</b>	A string generated when the event occurs within a program or application. The event key uniquely identifies a specific instance of the event.

## See Also

WF\_EVENTS\_PKG.Generate: page 8 – 303

WF\_EVENT\_GROUPS\_PKG.Generate: page 8 – 306

WF\_SYSTEMS\_PKG.Generate: page 8 – 309

WF\_AGENTS\_PKG.Generate: page 8 – 312

WF\_EVENT\_SUBSCRIPTIONS\_PKG.Generate: page 8 – 315

Predefined Workflow Events: page 14 – 2

---

## Receive

**PL/SQL Syntax**    `function Receive`

```
(p_subscription_guid in raw,  
 p_event in out wf_event_t) return varchar2;
```

**Description**    Receives Business Event System object definitions during subscription processing and loads the definitions into the appropriate Business Event System tables. This function completes the replication of the objects from one system to another.

*WF\_EVENT\_FUNCTIONS\_PKG.Receive()* is defined according to the standard API for an event subscription rule function. Oracle Workflow uses *WF\_EVENT\_FUNCTIONS\_PKG.Receive()* as the rule function for two predefined subscriptions, one that is triggered when the System Signup event is raised locally, and one that is triggered when any of the events in the Seed event group is received from an external source.

The Seed event group includes the following events:

- oracle.apps.wf.event.event.create
- oracle.apps.wf.event.event.update
- oracle.apps.wf.event.event.delete
- oracle.apps.wf.event.group.create
- oracle.apps.wf.event.group.update
- oracle.apps.wf.event.group.delete
- oracle.apps.wf.event.system.create
- oracle.apps.wf.event.system.update
- oracle.apps.wf.event.system.delete
- oracle.apps.wf.event.agent.create
- oracle.apps.wf.event.agent.update
- oracle.apps.wf.event.agent.delete
- oracle.apps.wf.event.subscription.create
- oracle.apps.wf.event.subscription.update
- oracle.apps.wf.event.subscription.delete
- oracle.apps.wf.event.all.sync

*WF\_EVENT\_FUNCTIONS\_PKG.Receive()* parses the event data XML document from the event message that was received and then loads the Business Event System object definitions into the appropriate tables.

**Note:** For the event, event group, system, agent, and subscription definition events, *WF\_EVENT\_FUNCTIONS\_PKG.Receive()* calls the Receive APIs associated with the corresponding tables to parse the XML document and load the definition into the table.

### Arguments (input)

<b>p_subscription_guid</b>	The globally unique identifier of the subscription.
<b>p_event</b>	The event message.

### See Also

*WF\_EVENTS\_PKG.Receive*: page 8 – 304

*WF\_EVENT\_GROUPS\_PKG.Receive*: page 8 – 307

*WF\_SYSTEMS\_PKG.Receive*: page 8 – 310

*WF\_AGENTS\_PKG.Receive*: page 8 – 313

*WF\_EVENT\_SUBSCRIPTIONS\_PKG.Receive*: page 8 – 316

Predefined Workflow Events: page 14 – 2

---

## Business Event System Replication APIs

You can call the following APIs to replicate Business Event System data across your systems. The replication APIs are stored in the following PL/SQL packages, each of which corresponds to a Business Event System table. Oracle Workflow provides both a Generate function and a Receive function for each table.

- WF\_EVENTS\_PKG
  - WF\_EVENTS\_PKG.Generate: page 8 – 303
  - WF\_EVENTS\_PKG.Receive: page 8 – 304
- WF\_EVENT\_GROUPS\_PKG
  - WF\_EVENT\_GROUPS\_PKG.Generate: page 8 – 306
  - WF\_EVENT\_GROUPS\_PKG.Receive: page 8 – 307
- WF\_SYSTEMS\_PKG
  - WF\_SYSTEMS\_PKG.Generate: page 8 – 309
  - WF\_SYSTEMS\_PKG.Receive: page 8 – 310
- WF\_AGENTS\_PKG
  - WF\_AGENTS\_PKG.Generate: page 8 – 312
  - WF\_AGENTS\_PKG.Receive: page 8 – 313
- WF\_EVENT\_SUBSCRIPTIONS\_PKG
  - WF\_EVENT\_SUBSCRIPTIONS\_PKG.Generate: page 8 – 315
  - WF\_EVENT\_SUBSCRIPTIONS\_PKG.Receive: page 8 – 316

Each Generate API produces an XML message containing the complete information from the appropriate table for the specified Business Event System object definition. The corresponding Receive API parses the XML message and loads the row into the appropriate table.

Oracle Workflow uses these APIs during the automated replication of Business Event System data. The Generate APIs are called by *WF\_EVENT\_FUNCTIONS\_PKG.Generate()*, while the Receive APIs are called by *WF\_EVENT\_FUNCTIONS\_PKG.Receive()*. See: Generate: page 8 – 296 and Receive: page 8 – 298.

---

### Document Type Definitions

The document type definitions (DTDs) for the Workflow table XML messages are defined under the master tag *WF\_TABLE\_DATA*. Beneath the master tag, each DTD has a tag identifying the Workflow



table name to which it applies, and beneath that, a version tag as well as tags for each column in the table. The following example shows how the DTDs are structured:

```
<WF_TABLE_DATA>                <- masterTagName
  <WF_TABLE_NAME>                <- m_table_name
    <VERSION></VERSION>          <- m_package_version
      <COL1></COL1>
      <COL2></COL2>
    </WF_TABLE_NAME>
  </WF_TABLE_DATA>
```

The Business Event System replication APIs use the following DTDs:

- WF\_EVENTS DTD: page 8 – 302
- WF\_EVENT\_GROUPS DTD: page 8 – 305
- WF\_SYSTEMS DTD: page 8 – 308
- WF\_AGENTS DTD: page 8 – 311
- WF\_EVENT\_SUBSCRIPTIONS DTD: page 8 – 314

---

## WF\_EVENTS Document Type Definition

The following document type definition (DTD) describes the required structure for an XML message that contains the complete information for an event definition in the WF\_EVENTS table.

```
<WF_TABLE_DATA>
  <WF_EVENTS>
    <VERSION></VERSION>
    <GUID></GUID>
    <NAME></NAME>
    <STATUS></STATUS>
    <GENERATE_FUNCTION></GENERATE_FUNCTION>
    <OWNER_NAME></OWNER_NAME>
    <OWNER_TAG></OWNER_TAG>
    <DISPLAY_NAME></DISPLAY_NAME>
    <DESCRIPTION></DESCRIPTION>
  </WF_EVENTS>
</WF_TABLE_DATA>
```



---

## WF\_EVENTS\_PKG.Receive

**PL/SQL Syntax**    procedure Receive

                          (x\_message in varchar2);

**Description**    Receives an XML message containing the complete information for an event definition and loads the information into the WF\_EVENTS table.

### Arguments (input)

<b>x_message</b>	An XML message containing the complete information for an event definition.
------------------	---

---

## WF\_EVENT\_GROUPS Document Type Definition

The following document type definition (DTD) describes the required structure for an XML message that contains the complete information for an event group member definition in the WF\_EVENT\_GROUPS table.

**Note:** Event group header information is defined in the WF\_EVENTS table, similarly to an individual event. Only the event group member definitions are stored in the WF\_EVENT\_GROUPS table.

```
<WF_TABLE_DATA>
  <WF_EVENT_GROUPS>
    <VERSION></VERSION>
    <GROUP_GUID></GROUP_GUID>
    <MEMBER_GUID></MEMBER_GUID>
  </WF_EVENT_GROUPS>
</WF_TABLE_DATA>
```

---

## WF\_EVENT\_GROUPS\_PKG.Generate

**PL/SQL Syntax**    function Generate  
                              (x\_group\_guid in raw,  
                              x\_member\_guid in raw)  
                              return varchar2;

**Description**    Generates an XML message containing the complete information from the WF\_EVENT\_GROUPS table for the specified event group member definition.

### Arguments (input)

<b>x_group_guid</b>	The globally unique identifier of the event group.
<b>x_member_guid</b>	The globally unique identifier of the individual member event.

---

## WF\_EVENT\_GROUPS\_PKG.Receive

**PL/SQL Syntax**    procedure Receive

(x\_message in varchar2);

**Description**    Receives an XML message containing the complete information for an event group member definition and loads the information into the WF\_EVENT\_GROUPS table.

**Arguments (input)**

<b>x_message</b>	An XML message containing the complete information for an event group member definition.
------------------	--

---

## WF\_SYSTEMS Document Type Definition

The following document type definition (DTD) describes the required structure for an XML message that contains the complete information for a system definition in the WF\_SYSTEMS table.

```
<WF_TABLE_DATA>
  <WF_SYSTEMS>
    <VERSION></VERSION>
    <GUID></GUID>
    <NAME></NAME>
    <MASTER_GUID></MASTER_GUID>
    <DISPLAY_NAME></DISPLAY_NAME>
    <DESCRIPTION></DESCRIPTION>
  </WF_SYSTEMS>
</WF_TABLE_DATA>
```







---

## WF\_AGENTS Document Type Definition

The following document type definition (DTD) describes the required structure for an XML message that contains the complete information for an agent definition in the WF\_AGENTS table.

```
<WF_TABLE_DATA>
  <WF_AGENTS>
    <VERSION></VERSION>
    <GUID></GUID>
    <NAME></NAME>
    <SYSTEM_GUID></SYSTEM_GUID>
    <PROTOCOL></PROTOCOL>
    <ADDRESS></ADDRESS>
    <QUEUE_HANDLER></QUEUE_HANDLER>
    <QUEUE_NAME></QUEUE_NAME>
    <DIRECTION></DIRECTION>
    <STATUS></STATUS>
    <DISPLAY_NAME></DISPLAY_NAME>
    <DESCRIPTION></DESCRIPTION>
  </WF_AGENTS>
</WF_TABLE_DATA>
```





---

## WF\_EVENT\_SUBSCRIPTIONS Document Type Definition

The following document type definition (DTD) describes the required structure for an XML message that contains the complete information for an event subscription definition in the WF\_EVENT\_SUBSCRIPTIONS table.

```
<WF_TABLE_DATA>
  <WF_EVENT_SUBSCRIPTIONS>
    <VERSION></VERSION>
    <GUID></GUID>
    <SYSTEM_GUID></SYSTEM_GUID>
    <SOURCE_TYPE></SOURCE_TYPE>
    <SOURCE_AGENT_GUID></SOURCE_AGENT_GUID>
    <EVENT_FILTER_GUID></EVENT_FILTER_GUID>
    <PHASE></PHASE>
    <STATUS></STATUS>
    <RULE_DATA></RULE_DATA>
    <OUT_AGENT_GUID></OUT_AGENT_GUID>
    <TO_AGENT_GUID></TO_AGENT_GUID>
    <PRIORITY></PRIORITY>
    <RULE_FUNCTION></RULE_FUNCTION>
    <WF_PROCESS_NAME></WF_PROCESS_NAME>
    <PARAMETERS></PARAMETERS>
    <OWNER_NAME></OWNER_NAME>
    <DESCRIPTION></DESCRIPTION>
  </WF_EVENT_SUBSCRIPTIONS>
</WF_TABLE_DATA>
```







# Index

## Symbols

&#NID, 4-12, 4-13, 4-15, 4-32

#FROM\_ROLE attribute, 4-25

#HIDE\_REASSIGN attribute, 4-25

## A

AbortProcess(), 8-36

Access Level, 2-102  
default, 2-105

Access level indicator, 4-17

Access property page, 4-17

Access protection

*See also* Access level; Protection level  
preserving customizations, 4-18

AccessCheck(), 8-230

ACCOUNT parameter, 2-59

Acknowledge Ping event, 14-8

ACTID, 7-5, 7-14

Actions, for subscriptions, 13-37

Activities, 3-10, 4-42

accessing from different data stores, 5-7, 6-2

Concurrent Manager, 6-22

copy, 4-60

cost, 4-47

create, 4-48, 4-50, 4-54, 4-57

deferred, 4-47

effective date, 4-59

error process, 4-59

event, 4-42, 4-45

External Java functions, 2-86

for an error process, 6-26

function, 4-42, 4-44

icons, 2-85, 4-49, 4-52, 4-54, 4-58, 4-62

in a loop, 4-60

in the Buyer: Advanced Shipment Notice  
process, 15-84

in the Buyer: Receive Supplier Invoicing  
process, 15-86

in the Buyer: Receive Supplier PO  
Acknowledgement process, 15-81

in the Buyer: Send PO to Supplier process,  
15-78

in the Buyer: Top Level PO process, 15-75

in the Detail Ping process, 13-82

in the Master Ping process, 13-80

in the Notify Approver subprocess, 15-21

in the Requisition process, 15-15

in the Supplier: Advanced Shipment Notice  
process, 15-99

in the Supplier: Credit Check process, 15-95

in the Supplier: Get Order Details process,  
15-92

in the Supplier: Send Supplier Invoice  
process, 15-101

in the Supplier: Stock Check process, 15-97

in the Supplier: Top Level Order process,  
15-88

in the Workflow Event Protocol process,  
14-21

joining branches, 5-4

notification, 4-42, 4-43

optional details, 4-59

process, 4-42, 4-46

processing cost, 8-9

result type, 4-48, 4-52, 4-58

- Standard, 4–42, 6–2
- statuses, 8–3
- System: Error, 4–42
- timing out, 5–10
- version number, 4–60
- Activities( ), 8–114
- Activity attributes
  - See also* Function activity attributes
  - setting values for, 5–12
- Activity nodes
  - in the Buyer: Advanced Shipment Notice process, 15–84
  - in the Buyer: Receive Supplier Invoicing process, 15–86
  - in the Buyer: Receive Supplier PO Acknowledgement process, 15–81
  - in the Buyer: Send PO to Supplier process, 15–78
  - in the Buyer: Top Level PO process, 15–75
  - in the Detail Ping process, 13–82
  - in the Master Ping process, 13–80
  - in the Notify Approver subprocess, 15–21
  - in the Requisition process, 15–15
  - in the Supplier: Advanced Shipment Notice process, 15–99
  - in the Supplier: Credit Check process, 15–95
  - in the Supplier: Get Order Details process, 15–92
  - in the Supplier: Send Supplier Invoice process, 15–101
  - in the Supplier: Stock Check process, 15–97
  - in the Supplier: Top Level Order process, 15–88
  - in the Workflow Event Protocol process, 14–21
- Ad hoc users and roles, 5–24
  - APIs, 8–121
- AddAttr( ), 8–216
- AddCorrelation( ), 8–294
- AddItemAttr( ), 8–43
- addItemAttrDate( ), 8–43
- AddItemAttrDateArray( ), 8–46
- addItemAttrNumber( ), 8–43
- AddItemAttrNumberArray( ), 8–46
- addItemAttrText( ), 8–43
- AddItemAttrTextArray( ), 8–46
- AddParameterToList, 8–253
- AddParameterToList( ), 8–275
- AddParameterToListPos( ), 8–276
- Address, 8–253
- AddUsersToAdHocRole( ), 8–138
- AdHocDirectory( ), 8–118
- Administrator privileges, 2–16
- Advanced Queues integration, 8–162
- Advanced Queuing, 13–2
- Agent, datatype, 8–237
- Agent Created event, 14–4
- Agent Deleted event, 14–5
- Agent Updated event, 14–4
- Agents, 13–22
  - defining, 13–29
  - deleting, 13–33
  - direction, 13–22
  - finding, 13–32
  - pinging, 13–77
  - protocol, 13–23
  - queue handlers, 13–25
  - queues, 13–24
  - scheduling listeners, 13–56
  - scheduling propagations, 13–61
  - updating, 13–33
- Agents web page, 13–29, 13–33
- ALLOW\_FORWARDED\_RESPONSE
  - parameter, 2–61
- And activity, 6–2
- Any event, 14–10
- Any transitions, 5–2
- APIs, 8–3
- AQ message payload, 8–163
- Arrows, 5–2
- Assign activity, 6–14
- AssignActivity( ), 8–74
- Asynchronous processes, 8–14, C – 2
- Attribute, token substitution, 4–41
- Attribute types
  - attribute, 4–11
  - date, 4–10, 4–35
  - document, 4–11, 4–14, 4–36

- event, 4–11, 4–36
- form, 4–10, 4–13, 4–36
- lookup, 4–10, 4–35
- number, 4–10, 4–35
- role, 4–11, 4–36
- text, 4–10, 4–35
- URL, 4–10, 4–12, 4–35
- Attribute–type attributes, 4–4
- Attributes
  - copy, 4–16
  - type, 4–3, 4–10, 4–35
- AUTOCLOSE\_FYI parameter, 2–61
- Automatic Notification Handler, 10–25
- Automatic replication, of Event Manager objects, 13–71
- Automatic responses, 10–25
- Automatic routing, 10–25

## B

- B2B Advanced Shipment Notice event, 15–106
- B2B Invoice event, 15–107
- B2B Purchase Order Acknowledgement event, 15–105
- B2B Purchase Order event, 15–102
- Background engine, scripts, 16–7
- Background Engines
  - about, 2–43
  - scripts, 16–6
  - starting, 2–44
  - submitting, 2–45
- Background( ), 8–41
- BeginActivity( ), 8–67
- Block activity, 6–5
- Business Event System, 1–3
  - checking setup, 13–53
  - managing business events, 13–2
  - overview, 8–235
  - Ping/Acknowledge example, 13–77
  - predefined events, 14–2
  - setting up, 2–96
- Business Event System Replication APIs, 8–300

- Business events, 13–4
  - in Workflow processes, 8–17
- Buyer Workbench, web page, 15–67
- Buyer: Advanced Shipment Notice process, summary, 15–83
- Buyer: Receive Supplier Invoicing process, summary, 15–85
- Buyer: Receive Supplier PO Acknowledgement process, summary, 15–80
- Buyer: Send PO to Supplier process, summary, 15–78
- Buyer: Top Level PO process, summary, 15–73

## C

- Callback functions, 7–13
  - command, 7–15
  - for item types, 4–5
- Cancel( ), 8–209
- CancelGroup( ), 8–210
- Check Setup web page, 13–53, 13–58, 13–62
- Checking
  - activity versions, 16–17
  - background engines, 16–7
  - directory service data model, 16–10
  - foreign/primary key references, 16–13
  - workflow data model, 16–16
- CLEAR( ), 8–102
- ClearMsgStack( ), 8–180
- Close( ), 8–215
- Compare Date activity, 6–3
- Compare Event Property activity, 6–16
- Compare Execution Time activity, 6–3
- Compare Number activity, 6–3
- Compare Text activity, 6–3
- compareTo( ), 8–100
- Comparison activities, 6–3
- CompleteActivity( ), 8–69
- CompleteActivityInternalName( ), 8–72
- Concurrent Manager activities, 6–22
- Concurrent Manager Functions item type, 6–22
- Concurrent program
  - FNDWFLST, 16–4

- FNDWFPR, 16–5
- Concurrent programs
  - Notification Mailer, 2–48, 2–56
  - Purge Obsolete Workflow Runtime Data, 8–119
  - Workflow Agent Listener, 8–272
  - Workflow Background Process, 2–45
  - Workflow Definitions Loader, 2–109
  - Workflow Resource Generator, 8–105
- CONNECT parameter, 2–58
- Constants, WFAttribute class, 8–90
- Content, 8–252
- Content–attached checkbox, 4–37
- CONTEXT(), 8–108
- Continue Flow activity, 6–12
- Coordinating master/detail activities, 6–11
- Cost threshold, 4–47
- CreateAdHocRole(), 8–136
- CreateAdHocUser(), 8–134
- CreateForkProcess(), 8–38
- CreateMsg(), 8–181
- CreateProcess(), 8–21
- Custom logos, in web pages, 2–84
- Customization Level, 2–105
  - for activities, 4–8, 4–11, 4–20, 4–32, 4–50, 4–53, 4–56, 4–58, 4–63

## D

- Data types, wf\_payload\_t, 8–163
- Database links
  - checking, 13–55
  - creating, 2–96
- Datatypes
  - example, 8–255
  - for the Business Event System, 8–236
  - WF\_AGENT\_T, 8–237
  - WF\_EVENT\_T, 8–242
  - WF\_PARAMETER\_LIST\_T, 8–241
  - WF\_PARAMETER\_T, 8–239
- Date–type attributes, 4–3
- DBA Studio, 2–96
- DEBUG parameter, 2–62

- Default Error Process, 6–28
- Default Event Error Process, 6–34
- Default transitions, 5–2
- DEFAULT\_ERROR, 6–28
- DEFAULT\_EVENT\_ERROR, 6–34
- Default\_Rule(), 8–281
- Defer Thread activity, 6–6
- Deferred activities, 2–43, 4–47
  - performance, C – 7
- Deferred processing
  - for event subscriptions, 13–41
  - for workflow processes, 2–43, 8–9, C – 7
- DeferredQueue function, 8–177
- Delete
  - all workflow data, 16–14
  - data for an item type, 16–15
  - item type attributes, 16–14
  - runtime data for an item type, 16–15, C – 9
  - workflow status information, 16–15
- Demonstration, directory service, 15–7
- Dequeue, queue handler, 7–24
- DequeueEventDetail(), 8–170
- DequeueException(), 8–176
- DequeueOutbound(), 8–167
- Detail Notification web page, 10–19
- Detail Ping process, summary, 13–81
- Detail process, 6–12
- Detail Survey process
  - activities, 15–47
  - summary, 15–46
- Diagram arrows, 5–2
- Direct Response e–mail, 10–3
- DIRECT\_RESPONSE parameter, 2–60
- Directory repository, 2–21
- Directory Service
  - in Navigator tree, 3–4
  - view from Builder, 5–26
- Directory services, 2–21
  - checking the data model, 2–27, 16–10
  - integrating with local workflow users, 2–28
  - integrating with native Oracle users, 2–27
  - integrating with Oracle HR, 2–27
  - synchronization, 2–30, 8–144

- Directory Services APIs, 8–121
- DISCARD parameter, 2–63
- Dispatch mode, 13–43
- Document integration, 4–3, 4–11, 4–36, 7–17
- Document Management, item type, 15–49
- Document Management APIs, 8–185
- Document management integration, 4–3, 4–6
- Document message attributes, attached vs embedded, 4–37
- Document Review process, 15–49
  - activities, 15–52
  - summary, 15–50
- Document Type Definitions
  - Business Event System, 8–300
  - WF\_AGENTS, 8–311
  - WF\_EVENT\_GROUPS, 8–305
  - WF\_EVENT\_SUBSCRIPTIONS, 8–314
  - WF\_EVENTS, 8–302
  - WF\_SYSTEMS, 8–308
- Document-type attributes, 4–3
- Documents, 4–6
- Dynamic priority, 5–11
- Dynamic timeouts, 5–10

## E

- E-mail notifications, 1–5, 2–48
  - and HTML attachments, 2–4
  - example direct response instructions, 10–7
  - modifying mail templates, 2–69
  - requirements, 2–4
  - summaries, 10–24
  - templates for, 2–48, 10–3
  - with HTML attachments, 10–2
- Edit menu, A – 3
- Effective date, 3–16
- Effective dates, 3–14, 3–16, 4–59, 8–11
- Effectivity, dates of, 3–7
- END activities, 5–4
- End Activity, 6–8
- Engine thresholds, 2–47
- Enqueue, queue handler, 7–23

- Enqueue( ), 8–269
- EnqueueInbound( ), 8–165
- Environment variables
  - WF\_ACCESS\_LEVEL, 2–102, 2–106
  - WF\_RESOURCES, 2–42
- Error activities, 6–26
- Error Check process, 15–54
  - activities, 15–57
  - summary, 15–56
- Error handling
  - for event subscriptions, 13–44
  - for process activities, 8–77
  - for workflow processes, 8–10
- Error process, 4–59, 6–26
- Error( ), 8–284
- Error\_Rule( ), 8–288
- Errored activities, retrying, 16–13
- Event activities, 4–45
  - create, 4–54
  - Workflow Engine, 8–17
- Event activity attributes, 4–56
- Event activity details, 5–12
- Event APIs, 8–260
- Event Created event, 14–2
- Event data, 7–21, 13–5, 13–36
- Event data URL, 8–50
- Event Deleted event, 14–3
- Event Function APIs, 8–290
- Event Group Creation event, 14–3
- Event Group Deleted event, 14–3
- Event Group Updated event, 14–3
- Event groups, 13–4
  - defining, 13–8
- Event Manager, 13–3
- Event messages
  - datatype, 8–242
  - enqueueing, 16–10
- Event nodes, 5–12
- Event Rule APIs, 8–279
- Event subscriptions, 13–34
  - rule functions, 7–25
- Event Subscriptions web page, 13–52
- Event System Demonstration
  - data model, 15–64

- initiating, 15–66
- overview, 15–63
- setting up, 15–66
- Event System Demonstration process,
  - installing, 15–64
- Event System Local Queues web page, 13–73
- Event Updated event, 14–2
- Event(), 8–75
- Event-type attributes, 4–4
- Events, 13–4
  - defining, 13–5
  - deleting, 13–15
  - finding, 13–14
  - predefined, 14–2
  - raising, 13–4, 13–65
  - sending to agents, 13–39
  - sending to workflow processes, 13–38
  - updating, 13–15
- Events web page, 13–5, 13–8, 13–15, 13–45
- Events: Buyer Workbench, web page, 15–67
- Events: Track Order, web page, 15–69
- Example function activity
  - Select Approver, 15–26
  - Verify Authority, 15–29
- Example process
  - Event System Demonstration, 15–63
  - Requisition, 15–5
- Execute Concurrent Program activity, 6–22
- execute(), 8–89
- External document integration, 4–6
- External Java function activities, 2–86, 8–5, 8–82

## F

- FAILCOMMAND parameter, 2–62
- File menu, A – 2
- Find Agent web page, 13–32
- Find Event web page, 13–14, 13–50
- Find Notifications web page, 10–15
- Find System web page, 13–19
- FND\_FNDWFNFIAS, 11–8

- FND\_FNDWFNFIAS, 11–8
- FND\_FNDWFNOT, 10–14
- FNDWFLST, 8–272
  - concurrent program, 16–4
- FNDWFPR, 8–119
  - concurrent program, 16–5
- Fonts
  - modifying, 5–21
  - setting, 5–21
- Forced synchronous processes, 8–14, C – 2
- Form-type attributes, 4–3
- FORWARD mode, 8–13
- Forward(), 8–194, 8–205
- Frame target, URL attributes, 4–37
- FROM parameter, 2–59
- FROM\_ROLE attribute, 4–25
- FUNCMODE, 7–5, 7–6
- Function activities, 4–44
  - create, 4–50
  - standard Java API, 7–8
  - standard PL/SQL API, 7–3
- Function activity attributes, 4–8, 4–53
- Functions, 3–10
  - See also* PL/SQL procedures
- Future-dated events, 13–41

## G

- Generate function, 13–5
- Generate()
  - WF\_AGENTS\_PKG, 8–312
  - WF\_EVENT\_FUNCTIONS\_PKG, 8–296
  - WF\_EVENT\_GROUPS\_PKG, 8–306
  - WF\_EVENT\_SUBSCRIPTIONS\_PKG, 8–315
  - WF\_EVENTS\_PKG, 8–303
  - WF\_SYSTEMS\_PKG, 8–309
- Get Event Property activity, 6–15
- Get Monitor URL activity, 6–14
- GET\_ERROR(), 8–103
- get\_launch\_attach\_url(), 8–187
- get\_launch\_document\_url(), 8–186
- get\_open\_dm\_select\_window(), 8–188, 8–189
- get\_pref(), 8–148
- GetAccessKey(), 8–150

- getActivityAttr(), 8–85
- GetActivityAttrClob(), 8–66
- GetActivityAttrDate(), 8–64
- GetActivityAttrEvent(), 8–64
- GetActivityAttrInfo(), 8–63
- GetActivityAttrNumber(), 8–64
- GetActivityAttrText(), 8–64
- GetActivityLabel(), 8–25
- GetAdvancedEnvelopeURL(), 8–155
- GetAttrDate(), 8–223
- GetAttrDoc(), 8–225
- GetAttrInfo(), 8–219
- GetAttrNumber(), 8–223
- GetAttrText(), 8–223
- GetBody(), 8–227
- getCorrelationID, 8–246
- GetDiagramURL(), 8–151
- GetEnvelopeURL(), 8–153
- getErrorMessage, 8–248
- getErrorStack, 8–248
- getErrorSubscription, 8–247
- getEventData, 8–247
- getEventKey, 8–247
- getEventName, 8–246
- getFormat(), 8–97
- getFromAgent, 8–247
- GetInfo(), 8–220
- getItemAttr(), 8–87
- GetItemAttrClob(), 8–60
- GetItemAttrDate(), 8–57
- GetItemAttrDocument(), 8–59
- GetItemAttrEvent(), 8–57
- getItemAttributes(), 8–61
- GetItemAttrInfo(), 8–62
- GetItemAttrNumber(), 8–57
- GetItemAttrText(), 8–57
- getItemTypes(), 8–56
- GetItemUserKey(), 8–24
- GetMessageHandle(), 8–175
- getName
  - WF\_AGENT\_T, 8–237

- WF\_PARAMETER\_T, 8–239
- WFAttribute, 8–94
- getNotificationAttributes(), 8–233
- getNotifications(), 8–232
- getParameterList, 8–246
- getPriority, 8–245
- getProcessStatus(), 8–81
- getReceiveDate, 8–246
- GetRoleDisplayName(), 8–131
- GetRoleInfo(), 8–125
- GetRoleInfo2(), 8–126
- GetRoleName(), 8–130
- GetRoleUsers(), 8–123
- getSendDate, 8–245
- GetShortBody(), 8–228
- GetShortText(), 8–222
- GetSubject(), 8–226
- getSystem, 8–237
- GetText(), 8–221
- getToAgent, 8–247
- getType(), 8–96
- GetUserName(), 8–129
- GetUserRoles(), 8–124
- getValue
  - WF\_PARAMETER\_T, 8–239
  - WFAttribute, 8–95
- GetValueForParameter, 8–253
- GetValueForParameter(), 8–277
- GetValueForParameterPos(), 8–278
- getValueType(), 8–98
- Global Preferences, web page, 2–14
- Global variables, 4–2

## H

- HandleError(), 8–77
- Hardware requirements, 2–2
- Help menu, A – 6
- Hidden item types, 3–4
- HIDE\_REASSIGN attribute, 4–25
- Home page, 9–2
- HTML\_MAIL\_TEMPLATE parameter, 2–65

HTMLAGENT parameter, 2–62

## I

Icons, 2–85

viewing, 4–49, 4–52, 4–54, 4–58

IDLE parameter, 2–61

InboundQueue function, 8–178

Init.ora parameters, 13–54

Initialize, 8–245

Initiating a workflow process, 15–8, 15–36,  
15–66

Internal names

updating activity, 16–7

updating activity attributes, 16–7

updating item attributes, 16–8

updating item types, 16–8

updating lookup codes, 16–8

updating lookup types, 16–9

updating message attributes, 16–9

updating messages, 16–9

IsPerformer(), 8–127

Item attributes, external document integration,  
4–6

Item type attributes, 4–2, 4–8, 8–12

arrays, 8–13

Event System Demonstration, 15–71

performance, C – 3

Requisition, 15–12

Workflow Send Protocol, 14–18

Item types, 3–9, 4–2

callback function, 4–5

Concurrent Manager Functions, 6–22

context reset, 7–13

copy, 4–15

creation, 4–7

Event System Demonstration, 15–71

loading, 3–12, 3–13

persistence type, 4–4, C – 8

Requisition, 15–12

saving, 3–12

selector functions, 4–5, 7–13

Standard, 6–2

System: Error, 6–26

System: Mailer, 2–69

Workflow Agent Ping/Acknowledge, 13–78

Workflow Send Protocol, 14–18

ITEMKEY, 7–4, 7–14

Items(), 8–113

ItemStatus(), 8–80

ITEMTYPE, 7–4, 7–14

## J

Java API, for function activities, 7–8

Java APIs, 8–5

Java Function Activity Agent, 2–86

starting, 2–86

stopping, 2–95, 16–11

Java interface, 8–5

Java monitor tool, 11–2

Java Runtime Environment, 2–5

JavaScript, support in a Web browser, 2–4

Joining activities, 5–4

## L

Launch Process activity, 6–6

LaunchProcess(), 8–30

LDAP, 2–30

LDAP APIs, 2–34, 8–144

List of values, in a web interface, 10–24, 13–22

Listen(), 8–270

Listeners

deleting, 13–61

for inbound agents, 13–56

running, 16–6

scheduling, 13–58

updating, 13–61

Load balancing, 6–9

loadActivityAttributes(), 8–84

Loader program. *See* Workflow Definitions  
Loader

Loading item types, 3–13

loadItemAttributes(), 8–83

Local system, 13–17

LOG parameter, 2–62



- Log( ), 8–283
- Login Server, 2–32
- Lookup codes, copy, 4–22
- Lookup types, 3–9, 4–19
  - copy, 4–22
  - creation, 4–20
- Lookup–type attributes, 4–3
- Loop Counter activity, 6–7
- Loop Reset, 5–3
- Loops, 4–60, 7–6, 8–10

## M

- MAPI-compliant mail application, 2–55
- Master Ping Process, summary, 13–79
- Master process, 6–12
- Master/copy systems, 13–72
- Master/Detail coordination activities, 6–11
  - notes on usage, 6–13
- Menus, Oracle Workflow Builder, A – 2
- Message attributes, 4–23, 4–24, 4–33, 4–34, 15–32
  - #FROM\_ROLE, 4–25
  - #HIDE\_REASSIGN, 4–25
  - for Workflow Cancelled Mail message, 2–78
  - for Workflow Closed Mail message, 2–81
  - for Workflow Invalid Mail message, 2–79
  - for Workflow Open FYI Mail message, 2–77
  - for Workflow Open Mail (Direct) message, 2–73
  - for Workflow Open Mail (Templated) message, 2–70
  - for Workflow Open Mail for Outlook Express message, 2–75
  - for Workflow Summary Mail message, 2–82
  - for Workflow URL Attachment message, 2–77
  - for Workflow Warning Mail message, 2–82
- formatted table, 4–26
- performance, C – 5
- Respond, 4–25, 4–35, 4–39
- Send, 4–24, 4–35
- source, 4–24, 4–35

- Message function, WF\_NOTIFICATION(), 4–26
- Message propagation, setting up, 13–53
- Message templates, for e–mail notifications, 2–69
- Messages, 3–9
  - body, 4–31, 15–31
  - copy, 4–41
  - creation, 4–29
  - overriding default priority, 5–11
  - subject, 4–30, 15–31
  - viewing, 15–32
- Messages window, 4–23
- MIME support, 2–49
- Mod\_osso, 2–32
- Monitoring
  - Workflow Monitor, 11–2
  - workitems, 1–5
- Multi-consumer queues, 13–40
- Multilingual support, 16–5, 16–12

## N

- Naming conventions, PL/SQL stored procedures, 15–15
- Navigator Toolbar, A – 7
- Navigator tree, finding objects in, 3–6
- NewAgent( ), 8–267
- NLS codeset, 2–65
- NLS support
  - in a web session, 2–39
  - in e–mail notifications, 2–39
  - in Oracle Workflow Builder, 2–38
- Node activities, dynamic priority, 5–11
- NODE parameter, 2–59
- Nodes
  - adding to a process, 5–6
  - start and end, 5–8
- NOOP activity, 6–7
- Notification, status, 16–12
- Notification access keys, 10–3
- Notification activities, 4–43
  - coupling with custom functions, 4–49, 8–13
  - create, 4–48

- Notify Requisition Approval Required, 15–31
  - Notification APIs, 8–192, 8–197
  - Notification functions, 4–49, 8–13
  - Notification history, 4–26
  - Notification ID token, 4–12, 4–13, 4–15, 4–32
  - Notification IDs, 10–3
  - Notification Mailer
    - about, 2–48
    - configuration file, 2–58
    - MIME support, 2–49
    - notification preference, 2–49
    - required folders, 2–63
    - response processing, 2–67
    - script to restart, 2–67
    - shutdown, 2–48
    - starting, 2–56
    - starting for MAPI-compliant applications, 2–57
    - starting for UNIX Sendmail, 2–55
  - Notification method, 10–2
  - Notification preference, 9–8
  - Notification preferences, 2–20, 2–49
  - Notification summaries, via e-mail, 10–24
  - Notification System, 2–48, 8–192
  - Notification templates, for e-mail notifications, 2–69
  - Notification Web page, 1–5
    - reassigning notifications, 10–22
  - Notifications, 10–2
    - dependence on directory services, 10–2
    - forwarding, 8–194
    - hiding the Reassign button, 4–25
    - HTML-formatted e-mail, 10–9
    - identifying the responder, 8–211
    - load balancing, 6–9
    - plain text e-mail using direct response, 10–6
    - plain text e-mail using templated response, 10–5
    - plain text e-mail with attachments, 10–11
    - reassign in Notification Web page, 10–22
    - reassign via e-mail, 10–12
    - responding with Notification Web page, 10–22
    - setting the From Role, 4–25
    - timed out, 8–195
    - transferring, 8–195
    - via e-mail, 2–48, 10–2
    - via Notification Web page, 10–13
  - Notifications Worklist. *See* Worklist web page
  - Notifications(), 8–115
  - Notify activity, 6–9
  - Notify Approver, example notification activities, 15–31
  - Notify Approver subprocess, summary, 15–19
  - Notify Requisition Approval Required, 15–31
  - Number-type attributes, 4–3
- ## O
- OMBAQ\_TEXT\_MSG, 8–257
  - On Revisit, 8–11
  - OpenNotificationsExist(), 8–214
  - Or activity, 6–2
  - Oracle Advanced Queues integration, 8–162
  - Oracle Advanced Queuing, 13–2
  - Oracle Applications Manager, 1–5
  - Oracle DBA Studio, 2–96
  - Oracle HTTP Server, 2–32
    - identifying the Workflow web agent, 2–17
    - Workflow server requirements, 2–4
  - Oracle Internet Directory, 2–30
  - Oracle Message Broker, 2–100
  - Oracle Net Services, 2–2
  - Oracle Workflow, implementation issues, 2–6
  - Oracle Workflow Builder, 1–3
    - Loader functionality, 3–15
    - overview, 3–2
    - requirements, 2–2
    - save modes, 3–15, 4–17
    - starting from command line, 3–17
  - Oracle Workflow home page, 9–2
  - Oracle Workflow Manager, 1–5
  - Oracle Workflow views, 8–157
  - Oracle9i Application Server, Workflow server requirements, 2–4
  - Oracle9iAS Single Sign-On, 2–32

OutboundQueue function, 8–179

## P

Parameter, datatype, 8–239

Parameter list, datatype, 8–241

Parameters( ), 8–291

Partitioning Workflow tables, 2–12, C – 8

Payload, for Advanced Queues messages, 8–163

Performance

- concepts, C – 2

- deferred activities, C – 7

- item attributes, C – 3

- message attributes, C – 5

- partitioning Workflow tables, C – 8

- purging, C – 8

- subprocesses, C – 5

- synchronous and asynchronous workflows, C – 2

Periodic Alert, item type, 15–54

Persistence, 4–4, C – 8

Phase numbers, 13–36, 13–42

Ping Agent event, 14–8

Pinging agents, 13–77

PL/SQL, 1–4

- document, 7–17

PL/SQL APIs

- for a 'PL/SQL CLOB' document, 7–17

- for a 'PL/SQL' document, 7–17

- for a Queue Handler, 7–23

- for a selector or callback function, 7–13

- for an Event Data Generate Function, 7–21

- for an Event Subscription Rule Function, 7–25

- for function activities, 7–3

PL/SQL CLOB, document, 7–17, 7–19

PL/SQL documents, 4–6

PL/SQL stored procedures

- creating, 15–15

- naming conventions, 15–15

- scripts, 15–15

Post-notification functions, 4–43, 8–13

Predefined events, 14–2

Preferred notification method, 10–2

Preserving customizations, for an activity, 4–18

Process activities, 4–46

- create, 4–57

Process definition, modifying, 3–11

Process diagram

- adding nodes, 5–6

- drawing, 5–2, 5–6

PROCESS parameter, 2–63

Process rollback, 8–77

Process window, 5–2

- editing, 5–2

Process Window Toolbar, A – 8

Processes

- activity transitions, 5–2

- copying to clipboard, 5–20

- creation, 3–7

- editing, 3–10, 3–12

- loops, 7–6, 8–10

- overview, 5–19

- printing, 5–20

- starting, 5–4

- verify, 5–21

ProcessInboundQueue( ), 8–174

Product Survey, web page, 15–36

Product Survey item type, 15–38

Product Survey process, 15–34

- initiating, 15–36

- installing, 15–35

Propagation, setting up, 13–53

Propagations

- deleting, 13–65

- for outbound agents, 13–61

- scheduling, 13–62

- updating, 13–65

Protection level, 2–103

- reset, 16–12

Protection level locking. *See* Access protection

Protocols, 13–23

Purge

- outbound notification message queue, 16–12

- performance, C – 8

- runtime data, 16–5

- Workflow Purge APIs, 8–111
- Purge Obsolete Workflow Runtime Data concurrent program, 8–119
- PurgeEvent(), 8–172
- PurgeItemType(), 8–173

## Q

- Queue handlers, 7–23, 13–25
  - WF\_EVENT\_OMB\_QH, 2–100
- Queue tables, 2–97
- Queues
  - assigned to agents, 13–24
  - checking, 13–56
  - reviewing, 13–72
  - setting up, 2–97

## R

- Raise Event web page, 13–66
- RAISE(), 8–105
- Raise(), 8–261
- Raising events, 13–4, 13–65
- Reassign notifications
  - hiding the Reassign button, 4–25
  - in Notification Web page, 10–22
  - via e-mail, 10–12
- Reassign web page, 10–22
- Receive date, for event messages, 8–270
- Receive()
  - WF\_AGENTS\_PKG, 8–313
  - WF\_EVENT\_FUNCTIONS\_PKG, 8–298
  - WF\_EVENT\_GROUPS\_PKG, 8–307
  - WF\_EVENT\_SUBSCRIPTIONS\_PKG, 8–316
  - WF\_EVENTS\_PKG, 8–304
  - WF\_SYSTEMS\_PKG, 8–310
- RemoveUsersFromAdHocRole, 8–143
- Replication APIs, Business Event System, 8–300
- REPLYTO parameter, 2–62
- Requirements, hardware and software, 2–2
- Requisition, data model, 15–6

- Requisition Demonstration, web page, 15–8
- Requisition process, 15–5
  - example function activities, 15–26
  - initiating, 15–8
  - installing, 15–6
  - summary, 15–13
- Reset process. *See* Rollback
- RESET\_FAILED parameter, 2–65
- RESET-NLS parameter, 2–65
- Respond attributes, 2–70, 2–72, 2–75, 2–79
- RESPOND mode, 8–13
- Respond to notification
  - HTML-formatted e-mail, 10–9
  - plain text e-mail using direct response, 10–6
  - plain text e-mail using templated response, 10–5
  - plain text e-mail with attachments, 10–11
  - via Notification Web page, 10–13
- Respond(), 8–194, 8–211
- Responder, 8–211
- Responder(), 8–212
- Response methods, direct vs. templated, 2–60
- Response processing, by Notification Mailer, 2–67
- Responses, processing, 8–194
- RESULT, 7–5, 7–15
- Result type
  - for activities, 4–48, 4–52, 4–58
  - for voting activities, 4–62
- ResumeProcess(), 8–34
- Retry Error, 6–32
- RETRY\_ONLY, 6–32
- Role
  - administrator, 2–16
  - property page, 5–26
- Role Resolution activity, 6–9
- Role-type attributes, 4–4
- Roles, 5–24
  - ad hoc, 5–24
  - loading into the Workflow Builder, 5–25
  - tab page, 5–24
  - view from Builder, 5–26
- Rollback, of process, 8–77
- Routing, automatic, 10–25

- Routing rules
  - deleting, 10–32
  - for a role, 10–27
  - listing, 10–26
  - overriding, 10–31
  - updating, 10–32
- Rule functions, 7–25
  - for event subscriptions, 13–37
- Runtime data, C – 8

## S

- Sample workflow processes, 15–2
- Savepoints, 7–3, 7–8, 8–4
- Schedule\_changes(), 8–147
- Seed event group, 14–6
- Select Approver function activity, 15–26
- Selector functions, 4–5, 7–13
- Send date, for event messages, 8–265
- Send(), 8–192, 8–199, 8–265
- SEND\_ACCESS\_KEY parameter, 2–66
- SendGroup(), 8–192, 8–203
- Set Event Property activity, 6–15
- set\_document\_id\_html(), 8–190
- SetAdHocRoleAttr(), 8–142
- SetAdHocRoleExpiration(), 8–140
- SetAdHocRoleStatus(), 8–133
- SetAdHocUserAttr(), 8–141
- SetAdHocUserExpiration(), 8–139
- SetAdHocUserStatus(), 8–132
- SetAttrDate(), 8–217
- SetAttrNumber(), 8–217
- SetAttrText(), 8–217
- setCorrelationID, 8–249
- SetDispatchMode(), 8–274
- SetErrorInfo(), 8–273
- setErrorMessage, 8–251
- setErrorStack, 8–252
- setErrorSubscription, 8–251
- setEventData, 8–250
- setEventKey, 8–250
- setEventName, 8–250
- setFromAgent, 8–251
- SetItemAttrDate(), 8–48
- SetItemAttrDateArray(), 8–53
- SetItemAttrDocument(), 8–51
- SetItemAttrEvent(), 8–48
- SetItemAttrNumber(), 8–48
- SetItemAttrNumberArray(), 8–53
- SetItemAttrText(), 8–48
- SetItemAttrTextArray(), 8–53
- setItemAttrValue(), 8–88
- SetItemOwner(), 8–26
- SetItemParent API, 6–12
- SetItemParent(), 8–79
- SetItemUserKey(), 8–23
- SetMsgAttr(), 8–183
- SetMsgResult(), 8–184
- setName
  - WF\_AGENT\_T, 8–238
  - WF\_PARAMETER\_T, 8–240
- setParameterList, 8–249
- SetParametersIntoParameterList(), 8–289
- setPriority, 8–248
- setReceiveDate, 8–249
- setSendDate, 8–248
- setSystem, 8–238
- setToAgent, 8–251
- setValue, 8–240
- Shortcuts, 5–22
- Shutdown files, 2–62
- SHUTDOWN parameter, 2–62
- Single sign-on, 2–30, 2–32
- Single-consumer queues, 13–40
- Software requirements, 2–2
- Source types, 13–35
- Standard activities, 6–2
- Standard APIs
  - for “PL/SQL CLOB” documents, 7–17, 7–19
  - for “PL/SQL” documents, 7–17
  - for a Queue Handler, 7–23

- for an Event Data Generate Function, 7–21
- for an Event Subscription Rule Function, 7–25
- for function activities, 7–3, 7–8
- for selector/callback functions, 7–13
- Standard error process, 6–26
- Standard item type, 6–2
- START activities, 5–4
- Start activity, 6–8
- StartForkProcess(), 8–40
- StartProcess function, for sample Requisition process, 15–23
- StartProcess(), 8–28
- Status report
  - developer, 16–16
  - end user, 16–16
- Stuck processes, 2–43
- Submit Concurrent Program activity, 6–23
- Subprocesses
  - performance, C – 5
  - timing out, 5–10
- Subscription Created event, 14–5
- Subscription Deleted event, 14–5
- Subscription Updated event, 14–5
- SubscriptionParameters(), 8–293
- Subscriptions, 13–34
  - deferring, 13–41
  - defining, 13–45
  - deleting, 13–52
  - finding, 13–50
  - predefined, 14–2
  - updating, 13–52
- Success(), 8–286
- SUMMARYONLY parameter, 2–59
- Supplier: Advanced Shipment Notice process, summary, 15–98
- Supplier: Credit Check process, summary, 15–94
- Supplier: Get Order Details process, summary, 15–91
- Supplier: Send Supplier Invoice process, summary, 15–100
- Supplier: Stock Check process, summary, 15–96
- Supplier: Top Level Order process, summary, 15–87
- Survey–Master/Detail process
  - activities, 15–44
  - summary, 15–42
- Survey–Single Process, activities, 15–41
- Survey–Single process, summary, 15–39
- SuspendProcess(), 8–32
- Synch\_all(), 8–146
- Synch\_changes(), 8–145
- Synchronization, with Oracle Internet Directory, 2–30, 2–34, 8–144
- Synchronize Event Systems event, 14–5
- Synchronous processes, 8–14, C – 2
- System Created event, 14–4
- System Deleted event, 14–4
- System identifier, 13–68
- System Identifier web page, 13–68
- System integration, 13–2
- System Signup event, 14–9
- System Signup web page, 13–69
- System Updated event, 14–4
- System: Error item type, 6–26
- System: Mailer item type, 2–69
- Systems, 13–17
  - defining, 13–18
  - deleting, 13–21
  - finding, 13–19
  - local, 13–17
  - master/copy, 13–72
  - signing up, 13–67, 13–69
  - synchronizing, 13–70
  - updating, 13–21
- Systems web page, 13–18, 13–21

## T

- Tag files, 2–63
- TAGFILE parameter, 2–63
- TCP/IP drivers, 2–2
- Templated Response e–mail, 10–3
- Test harness, 12–2
- Test(), 8–268

TEST\_ADDRESS parameter, 2–62  
TestContext(), 8–229  
Text-type attributes, 4–3  
Timed out processes, 2–43, C – 7  
Timeout transitions, 5–2, 5–3  
Timeouts, 5–10  
    dynamic, 5–10  
Token substitution  
    attributes, 4–41  
    of document-type message attributes, 4–14  
TOKEN(), 8–104  
Toolbars, Oracle Workflow Builder, A – 7  
toString(), 8–99  
Total(), 8–116  
TotalPERM(), 8–117  
TRANSFER mode, 8–13  
Transfer(), 8–195, 8–207  
Transitions, 5–2  
    Any, 5–2  
    creating, 5–18  
    Default, 5–2  
    editing, 5–18  
    Timeout, 5–2  
TRANSLATE(), 8–110  
Translation, 2–38

## U

Unexpected event, 14–12  
UNIX Sendmail, 2–55  
UNPROCESS parameter, 2–63  
Upgrading workflow definitions, 8–12  
URL attributes, frame target, 4–37  
URL message attributes, attached vs  
    embedded, 4–37  
URL-type attributes, 4–3  
URLs  
    for event data, 8–50  
    for Event System Demonstration web pages,  
        15–67, 15–69  
    for Find Notifications Routing Rules web  
        page, 10–27

    for Find Notifications web page, 10–13  
    for Find Processes web page, 11–8  
    for Notifications Routing Rules web page,  
        10–26  
    for Oracle Workflow home page, 9–2  
    for Product Survey web page, 15–36  
    for Requisition Demonstration web page,  
        15–10  
    for the Workflow Monitor, 11–7  
    for Worklist web page, 10–13  
User Defined Alert Action process  
    activities, 15–61  
    summary, 15–60  
User Entry Has Changed event, 14–15  
User Preferences, web page, 9–6  
User preferences, 2–14  
    document management home, 2–20, 9–8  
    language and territory, 2–19, 9–7  
    notification preference, 2–20, 9–8  
User-defined datatypes, for the Business Event  
    System, 8–236  
UserActive(), 8–128  
Users, ad hoc, 5–24

## V

Vacation forwarding, 10–25  
value(), 8–93  
Verify Authority function activity, 15–29  
Version, 8–11, 16–16  
    of Oracle Workflow, 2–9  
Version compatibility, 2–9  
Version number, for activities, 4–60  
Versioning, 3–7  
View menu, A – 4  
View notifications  
    e-mail summary, 10–24  
    electronic mail, 10–2  
    Notification Web page, 10–13  
    web browser, 10–12  
Views, Oracle Workflow, 8–157  
Vote Yes/No activity, 6–10  
VoteCount(), 8–213  
Voting activities  
    processing, 8–195

result type, 4–62  
Voting activity, 4–61

## W

Wait activity, 6–4  
Wait for Concurrent Program activity, 6–24  
Wait for Flow activity, 6–12  
Warning(), 8–285  
Web agent, for Oracle Workflow, 2–17  
Web home page, 9–2  
Web notifications, requirements, 2–4  
WF\_ACCESS\_LEVEL, 2–102, 2–106  
WF\_AGENT\_T, 8–237  
WF\_AGENTS Document Type Definition, 8–311  
WF\_AGENTS\_PKG.Generate, 8–312  
WF\_AGENTS\_PKG.Receive, 8–313  
WF\_DEFERRED agent, 13–25  
WF\_DEFERRED queue, 2–97  
WF\_ENGINE.BACKGROUND, 2–44  
WF\_ERROR agent, 13–25  
WF\_ERROR queue, 2–97  
WF\_ERROR\_QH, 13–25  
WF\_EVENT\_FUNCTIONS\_PKG.Generate(), 8–296  
WF\_EVENT\_FUNCTIONS\_PKG.Receive(), 8–298  
WF\_EVENT\_GROUPS Document Type Definition, 8–305  
WF\_EVENT\_GROUPS\_PKG.Generate, 8–306  
WF\_EVENT\_GROUPS\_PKG.Receive, 8–307  
WF\_EVENT\_OMB\_QH, 13–25  
    attribute mapping, 8–257  
    setting up, 2–100  
WF\_EVENT\_QH, 13–25  
WF\_EVENT\_SUBSCRIPTIONS Document Type Definition, 8–314  
WF\_EVENT\_SUBSCRIPTIONS\_PKG.Generate, 8–315  
WF\_EVENT\_SUBSCRIPTIONS\_PKG.Receive, 8–316

WF\_EVENT\_T, 8–242  
    mapping attributes to OMBAQ\_TEXT\_MSG, 8–257  
WF\_EVENTS Document Type Definition, 8–302  
WF\_EVENTS\_PKG.Generate, 8–303  
WF\_EVENTS\_PKG.Receive, 8–304  
WF\_IN agent, 13–25  
WF\_IN queue, 2–97  
WF\_ITEM\_ACTIVITY\_STATUSES\_V, 8–157  
WF\_ITEMS\_V, 8–161  
WF\_LANGUAGES view, 2–38  
WF\_LOCAL\_\* tables, 2–21  
WF\_NOTIFICATION() message function, 4–26  
WF\_NOTIFICATION\_ATTR\_RESP\_V, 8–159  
WF\_OUT agent, 13–25  
WF\_OUT queue, 2–97  
WF\_PARAMETER\_LIST\_T, 8–241  
WF\_PARAMETER\_T, 8–239  
wf\_payload\_t, 8–163  
WF\_PURGE, 8–111  
WF\_REQDEMO.SelectApprover, 15–26  
WF\_REQDEMO.StartProcess, 15–8  
WF\_REQDEMO.VerifyAuthority, 15–18, 15–29  
WF\_RESOURCES, environment variable, 2–42  
WF\_ROLES, view, 2–24  
WF\_RUNNABLE\_PROCESSES\_V, 8–160  
WF\_SYSTEMS Document Type Definition, 8–308  
WF\_SYSTEMS\_PKG.Generate, 8–309  
WF\_SYSTEMS\_PKG.Receive, 8–310  
WF\_USER\_ROLES, view, 2–25  
WF\_USERS, view, 2–22  
Wfagtlst.sql, 16–6  
WFAAttribute class, 8–90  
WFAAttribute(), 8–92  
Wfbkg.sql, 16–6  
Wfbkgchk.sql, 16–7  
Wfchact.sql, 16–7  
Wfchacta.sql, 16–7  
Wfchita.sql, 16–8  
Wfchitt.sql, 16–8



Wfchluc.sql, 16–8  
 Wfchlut.sql, 16–9  
 Wfchmsg.sql, 16–9  
 Wfchmsga.sql, 16–9  
 Wfdirchk.sql, 16–10  
 wfdircsv.sql, 2–28  
 wfdirhrv.sql, 2–27  
 wfdirouv.sql, 2–27  
 wfevquec.sql, 2–99  
 wfevqued.sql, 2–99  
 Wfevtenq.sql, 16–10  
 WFFunctionAPI class, 8–82  
 wfjvlsnr.bat, 2–87  
 wfjvlsnr.csh, 2–87  
 Wfjvstop.sql, 16–11  
 WFLOAD, 2–109  
 wflod, 2–108  
 wfmail.cfg, 2–58  
 Wfmqupd.sql, 16–12  
 WFNLADD.sql, 16–5  
 WFNLENA.sql, 16–12  
 Wfntfsh.sql, 16–12  
 Wfprot.sql, 16–12  
 Wfqclean.sql, 16–13  
 Wfquhndob.pls, 2–100  
 Wfquhndos.pls, 2–100  
 Wfrefchk.sql, 16–13  
 wfresgen, 8–105  
 Wfretry.sql, 16–13  
 Wfrmall.sql, 16–14  
 Wfrmita.sql, 16–14  
 Wfrmitms.sql, 16–15  
 Wfrmitt.sql, 16–15  
 Wfrmtime.sql, 16–15, C – 9  
 Wfrun.sql, 16–15  
 WFRUND.SQL, 15–8  
 Wfstat.sql, 16–16  
 Wfstatus.sql, 16–16  
 Wfststdchk.sql, 16–16  
 Wftypes.sql, 8–236  
 Wfupart.sql, 2–12  
 Wfupartb.sql, 2–12  
 Wfver.sql, 16–16  
 Wfverchk.sql, 16–17  
 Wfverupd.sql, 16–17  
 wfxload, 2–114, 2–117  
 wfxload.bat, 2–114, 2–117  
 Windows menu, A – 6  
 WorkCount( ), 8–231  
 Workflow administrator, 2–16  
 Workflow Agent Listener, 16–4  
 Workflow Agent Listener concurrent program, 8–272  
 Workflow Agent Ping/ Acknowledge, 13–77  
   item type, 13–78  
   item type attributes, 13–78  
 Workflow Builder menus, A – 2  
 Workflow Cancelled Mail message template, 2–78  
 Workflow Closed Mail message template, 2–81  
 Workflow Core APIs, 8–101  
 Workflow definitions  
   loading, 1–4  
   source control, 3–12  
   testing, 12–2  
   transferring, 2–107  
 Workflow Definitions Loader, 1–4, 2–107, 2–108  
   concurrent program, 2–109  
 Workflow Demonstrations home page, 15–2  
 Workflow Designer. *See* Oracle Workflow Builder  
 Workflow diagrams, displaying, 15–3  
 Workflow Directory Service APIs, 8–121  
 Workflow Engine, 1–3  
   calling after activity completion, 8–8  
   calling for activity initiation, 8–3  
   CANCEL mode, 8–11  
   core APIs, 8–101, 8–111  
   cost threshold, 4–47  
   deferred activities, 8–9  
   directory services, 8–121  
   error processing, 8–10  
   Java APIs, 8–5, 8–19  
   looping, 8–10

- master/detail processes, 8–79
- PL/SQL APIs, 8–19
- RUN mode, 8–11
- threshold cost, 2–47, 8–9
- Workflow Engine APIs, 8–3
- Workflow Event Protocol process, summary, 14–20
- Workflow Invalid Mail message template, 2–79
- Workflow LDAP APIs, 2–34, 8–144
- Workflow Monitor, 11–2
  - Administration buttons, 11–6
  - Detail Tab window, 11–4
  - Process Diagram window, 11–3
  - Process title, 11–3
  - setup, 2–84
- Workflow Monitor APIs, 8–149
- Workflow Notification APIs. *See* Notification APIs
- Workflow Open Mail (Direct) message template, 2–71
- Workflow Open Mail (Templated) message template, 2–69
- Workflow Open Mail for Outlook Express message template, 2–74
- Workflow Open Mail message template, 2–76
- Workflow Preferences API, 8–148
- Workflow processes
  - creating and starting, 16–15
  - monitoring, 11–2
  - samples, 15–2
- Workflow Purge APIs, 8–111
- Workflow Queue APIs, 8–162
- Workflow queues, cleaning, 16–13
- Workflow Resource Generator, 8–105
  - concurrent program, 8–106

- Workflow roles, 2–21
- Workflow Send Protocol
  - item type, 14–18
  - sample workflow process, 14–17
- Workflow Send Protocol Acknowledgement event, 14–26
- Workflow Send Protocol event, 14–24
- Workflow Server, requirements, 2–3
- Workflow Summary Mail message template, 2–82
- Workflow URL Attachment message template, 2–77
- Workflow users, 2–21
- Workflow Views, 8–157
- Workflow Warning Mail message template, 2–82
- Workflow web pages, modifying template, 2–84
- Workflow XML Loader, 2–112
- Workflow\_Protocol( ), 8–287
- Workitems. *See* Items
- Worklist web page, 10–17
- WriteMsg( ), 8–182
- WriteToClob( ), 8–234

## X

- XML Compare Tag Value (Date) activity, 6–19
- XML Compare Tag Value (Number) activity, 6–19
- XML Compare Tag Value (Text) activity, 6–19
- XML Compare Tag Value activities, 6–19
- XML Get Tag Value activity, 6–18
- XML Transform activity, 6–20

# Reader's Comment Form

## Oracle Workflow Guide Volume 1, Release 2.6.2 A95276-03

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.



# Oracle<sup>®</sup> Workflow Guide

RELEASE 2.6.2  
VOLUME 2  
March 2002

ORACLE<sup>®</sup>

Oracle Workflow Guide Volume 2, Release 2.6.2

The part number for this volume is A95277-03. To reorder this book, please use the set part number, A95265-03.

**Copyright © 1996, 2002 Oracle Corporation. All rights reserved.**

Primary Authors: Siu Chang, Clara Jaeckel

Major Contributors: George Buzsaki, John Cordes, Mark Craig, Kevin Hudson, George Kellner, David Lam, Jin Liu, Kenneth Ma, 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 laws. Reverse engineering, disassembly or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. 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 NOTICE**

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 Express, JInitiator, Oracle8, Oracle8i, Oracle9i, Oracle*MetaLink*, Oracle Press, Oracle Store, PL/SQL, and SQL\*Plus are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.



# Contents

## VOLUME 1

- Preface** ..... **xix**
- Audience for This Guide ..... xx
- How To Use This Guide ..... xx
- Documentation Accessibility ..... xxi
- Other Information Sources ..... xxii
- Online Documentation ..... xxii
- Related User’s Guides ..... xxiii
- Guides Related to All Products ..... xxiii
- User Guides Related to This Product ..... xxiv
- Installation and System Administration ..... xxv
- Other Implementation Documentation ..... xxvi
- Training and Support ..... xxviii
- Do Not Use Database Tools to Modify Oracle  
    Applications Data ..... xxix
- About Oracle ..... xxix
- Your Feedback ..... xxx

## Chapter 1

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

<b>Chapter 2</b>	<b>Setting Up Oracle Workflow</b> . . . . .	<b>2 – 1</b>
	Oracle Workflow Hardware and Software Requirements . . . . .	2 – 2
	Overview of Setting Up . . . . .	2 – 6
	Overview of Required Setup Steps for the Standalone Version of Oracle Workflow . . . . .	2 – 6
	Overview of Required Setup Steps for the Version of Oracle Workflow Embedded in Oracle Applications . . . . .	2 – 7
	Optional Setup Steps . . . . .	2 – 7
	Other Workflow Features . . . . .	2 – 8
	Identifying the Version of Your Oracle Workflow Server . . . . .	2 – 9
	Setup Flowchart . . . . .	2 – 10
	Setup Checklist . . . . .	2 – 11
	Setup Steps . . . . .	2 – 12
	Overview of Oracle Workflow Access Protection . . . . .	2 – 101
	Setting Up a Default Access Level . . . . .	2 – 105
	Using the Workflow Definitions Loader . . . . .	2 – 107
	Using the Workflow XML Loader . . . . .	2 – 112
<b>Chapter 3</b>	<b>Defining a Workflow Process</b> . . . . .	<b>3 – 1</b>
	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
	Using Oracle Workflow Builder with Different Server Versions . . . . .	3 – 21
	Item Type Definition Web Page . . . . .	3 – 24
<b>Chapter 4</b>	<b>Defining Workflow Process Components</b> . . . . .	<b>4 – 1</b>
	Workflow Process Components . . . . .	4 – 2
	Item Types . . . . .	4 – 2
	Allowing Access to an Object . . . . .	4 – 17
	Lookup Types . . . . .	4 – 19
	Messages . . . . .	4 – 23
	Activities . . . . .	4 – 42
	Voting Activity . . . . .	4 – 61
	Deleting Objects in Oracle Workflow Builder . . . . .	4 – 68
	Modifying Objects in Oracle Workflow Builder . . . . .	4 – 69
	Workflow Objects That Support Versioning . . . . .	4 – 70



	Workflow Objects That Do Not Support Versioning . . . . .	4–71
<b>Chapter 5</b>	<b>Defining a Workflow Process Diagram . . . . .</b>	<b>5–1</b>
	Process Window . . . . .	5–2
	Modifying Fonts in Oracle Workflow Builder . . . . .	5–21
	Creating a Shortcut Icon for a Workflow Process . . . . .	5–22
	Roles . . . . .	5–24
<b>Chapter 6</b>	<b>Predefined Workflow Activities . . . . .</b>	<b>6–1</b>
	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 . . . . .	6–8
	End Activity . . . . .	6–8
	Role Resolution Activity . . . . .	6–9
	Notify Activity . . . . .	6–9
	Vote Yes/No Activity . . . . .	6–10
	Master/Detail Coordination Activities . . . . .	6–11
	Wait for Flow Activity . . . . .	6–12
	Continue Flow Activity . . . . .	6–12
	Assign Activity . . . . .	6–14
	Get Monitor URL Activity . . . . .	6–14
	Get Event Property Activity . . . . .	6–15
	Set Event Property Activity . . . . .	6–15
	Compare Event Property Activity . . . . .	6–16
	XML Get Tag Value Activity . . . . .	6–18
	XML Compare Tag Value Activities . . . . .	6–19
	XML Transform Activity . . . . .	6–20
	Concurrent Manager Standard Activities . . . . .	6–22
	Execute Concurrent Program Activity . . . . .	6–22
	Submit Concurrent Program Activity . . . . .	6–23
	Wait for Concurrent Program Activity . . . . .	6–24
	Default Error Process . . . . .	6–26

System: Error Item Type and Item Attributes . . . . .	6 – 27
Default Error Process . . . . .	6 – 28
Retry-only Process . . . . .	6 – 32
Default Event Error Process . . . . .	6 – 34

## Chapter 7

<b>Defining Procedures and Functions for Oracle Workflow . . . .</b>	<b>7 – 1</b>
Defining Procedures and Functions for Oracle Workflow . . . . .	7 – 2
Standard API for PL/SQL Procedures Called by Function Activities . . . . .	7 – 3
Standard API for Java Procedures Called by Function Activities . . . . .	7 – 8
Standard API for an Item Type Selector or Callback Function . .	7 – 13
Standard APIs for “PL/SQL” and “PL/SQL CLOB” Documents . . . . .	7 – 17
“PL/SQL” Documents . . . . .	7 – 17
“PL/SQL CLOB” Documents . . . . .	7 – 19
Standard API for an Event Data Generate Function . . . . .	7 – 21
Standard APIs for a Queue Handler . . . . .	7 – 23
Enqueue . . . . .	7 – 23
Dequeue . . . . .	7 – 24
Standard API for an Event Subscription Rule Function . . . . .	7 – 25

## Chapter 8

<b>Oracle Workflow APIs . . . . .</b>	<b>8 – 1</b>
Oracle Workflow Procedures and Functions . . . . .	8 – 2
Overview of the Workflow Engine . . . . .	8 – 3
Oracle Workflow Java Interface . . . . .	8 – 5
Additional Workflow Engine Features . . . . .	8 – 8
Workflow Engine APIs . . . . .	8 – 19
CreateProcess . . . . .	8 – 21
SetItemUserKey . . . . .	8 – 23
GetItemUserKey . . . . .	8 – 24
GetActivityLabel . . . . .	8 – 25
SetItemOwner . . . . .	8 – 26
StartProcess . . . . .	8 – 28
LaunchProcess . . . . .	8 – 30
SuspendProcess . . . . .	8 – 32
ResumeProcess . . . . .	8 – 34
AbortProcess . . . . .	8 – 36
CreateForkProcess . . . . .	8 – 38
StartForkProcess . . . . .	8 – 40

Background	8 – 41
AddItemAttribute	8 – 43
AddItemAttributeArray	8 – 46
SetItemAttribute	8 – 48
SetItemAttrDocument	8 – 51
SetItemAttributeArray	8 – 53
getItemTypes	8 – 56
GetItemAttribute	8 – 57
GetItemAttrDocument	8 – 59
GetItemAttrClob	8 – 60
getItemAttributes	8 – 61
GetItemAttrInfo	8 – 62
GetActivityAttrInfo	8 – 63
GetActivityAttribute	8 – 64
GetActivityAttrClob	8 – 66
BeginActivity	8 – 67
CompleteActivity	8 – 69
CompleteActivityInternalName	8 – 72
AssignActivity	8 – 74
Event	8 – 75
HandleError	8 – 77
SetItemParent	8 – 79
ItemStatus	8 – 80
getProcessStatus	8 – 81
Workflow Function APIs	8 – 82
loadItemAttributes	8 – 83
loadActivityAttributes	8 – 84
getActivityAttr	8 – 85
getItemAttr	8 – 87
setItemAttrValue	8 – 88
execute	8 – 89
Workflow Attribute APIs	8 – 90
WFAttribute	8 – 92
value	8 – 93
getName	8 – 94
getValue	8 – 95
getType	8 – 96
getFormat	8 – 97
getValueType	8 – 98
toString	8 – 99
compareTo	8 – 100
Workflow Core APIs	8 – 101

CLEAR	8 – 102
GET_ERROR	8 – 103
TOKEN	8 – 104
RAISE	8 – 105
CONTEXT	8 – 108
TRANSLATE	8 – 110
Workflow Purge APIs	8 – 111
Items	8 – 113
Activities	8 – 114
Notifications	8 – 115
Total	8 – 116
TotalPERM	8 – 117
AdHocDirectory	8 – 118
Purge Obsolete Workflow Runtime Data Concurrent Program	8 – 119
Workflow Directory Service APIs	8 – 121
GetRoleUsers	8 – 123
GetUserRoles	8 – 124
GetRoleInfo	8 – 125
GetRoleInfo2	8 – 126
IsPerformer	8 – 127
UserActive	8 – 128
GetUserName	8 – 129
GetRoleName	8 – 130
GetRoleDisplayName	8 – 131
SetAdHocUserStatus	8 – 132
SetAdHocRoleStatus	8 – 133
CreateAdHocUser	8 – 134
CreateAdHocRole	8 – 136
AddUsersToAdHocRole	8 – 138
SetAdHocUserExpiration	8 – 139
SetAdHocRoleExpiration	8 – 140
SetAdHocUserAttr	8 – 141
SetAdHocRoleAttr	8 – 142
RemoveUsersFromAdHocRole	8 – 143
Workflow LDAP APIs	8 – 144
Synch_changes	8 – 145
Synch_all	8 – 146
Schedule_changes	8 – 147
Workflow Preferences API	8 – 148
get_pref	8 – 148
Workflow Monitor APIs	8 – 149

GetAccessKey .....	8 – 150
GetDiagramURL .....	8 – 151
GetEnvelopeURL .....	8 – 153
GetAdvancedEnvelopeURL .....	8 – 155
Oracle Workflow Views .....	8 – 157
WF_ITEM_ACTIVITY_STATUSES_V .....	8 – 157
WF_NOTIFICATION_ATTR_RESP_V .....	8 – 159
WF_RUNNABLE_PROCESSES_V .....	8 – 160
WF_ITEMS_V .....	8 – 161
Workflow Queue APIs .....	8 – 162
EnqueueInbound .....	8 – 165
DequeueOutbound .....	8 – 167
DequeueEventDetail .....	8 – 170
PurgeEvent .....	8 – 172
PurgeItemType .....	8 – 173
ProcessInboundQueue .....	8 – 174
GetMessageHandle .....	8 – 175
DequeueException .....	8 – 176
DeferredQueue .....	8 – 177
InboundQueue .....	8 – 178
OutboundQueue .....	8 – 179
ClearMsgStack .....	8 – 180
CreateMsg .....	8 – 181
WriteMsg .....	8 – 182
SetMsgAttr .....	8 – 183
SetMsgResult .....	8 – 184
Document Management APIs .....	8 – 185
get_launch_document_url .....	8 – 186
get_launch_attach_url .....	8 – 187
get_open_dm_display_window .....	8 – 188
get_open_dm_attach_window .....	8 – 189
set_document_id_html .....	8 – 190
Overview of the Oracle Workflow Notification System .....	8 – 192
Notification Model .....	8 – 192
Notification APIs .....	8 – 197
Send .....	8 – 199
Custom Callback Function .....	8 – 200
SendGroup .....	8 – 203
Forward .....	8 – 205
Transfer .....	8 – 207
Cancel .....	8 – 209

CancelGroup .....	8 – 210
Respond .....	8 – 211
Responder .....	8 – 212
VoteCount .....	8 – 213
OpenNotificationsExist .....	8 – 214
Close .....	8 – 215
AddAttr .....	8 – 216
SetAttribute .....	8 – 217
GetAttrInfo .....	8 – 219
GetInfo .....	8 – 220
GetText .....	8 – 221
GetShortText .....	8 – 222
GetAttribute .....	8 – 223
GetAttrDoc .....	8 – 225
GetSubject .....	8 – 226
GetBody .....	8 – 227
GetShortBody .....	8 – 228
TestContext .....	8 – 229
AccessCheck .....	8 – 230
WorkCount .....	8 – 231
getNotifications .....	8 – 232
getNotificationAttributes .....	8 – 233
WriteToClob .....	8 – 234
Overview of the Oracle Workflow Business Event System .....	8 – 235
Business Event System Datatypes .....	8 – 236
Agent Structure .....	8 – 237
getName .....	8 – 237
getSystem .....	8 – 237
setName .....	8 – 238
setSystem .....	8 – 238
Parameter Structure .....	8 – 239
getName .....	8 – 239
getValue .....	8 – 239
setName .....	8 – 240
setValue .....	8 – 240
Parameter List Structure .....	8 – 241
Event Message Structure .....	8 – 242
Initialize .....	8 – 245
getPriority .....	8 – 245
getSendDate .....	8 – 245
getReceiveDate .....	8 – 246
getCorrelationID .....	8 – 246

getParameterList	8 – 246
getEventName	8 – 246
getEventKey	8 – 247
getEventData	8 – 247
getFromAgent	8 – 247
getToAgent	8 – 247
getErrorSubscription	8 – 247
getErrorMessage	8 – 248
getErrorStack	8 – 248
setPriority	8 – 248
setSendDate	8 – 248
setReceiveDate	8 – 249
setCorrelationID	8 – 249
setParameterList	8 – 249
setEventName	8 – 250
setEventKey	8 – 250
setEventData	8 – 250
setFromAgent	8 – 251
setToAgent	8 – 251
setErrorSubscription	8 – 251
setErrorMessage	8 – 251
setErrorStack	8 – 252
Content	8 – 252
Address	8 – 253
AddParameterToList	8 – 253
GetValueForParameter	8 – 253
Example for Using Abstract Datatypes	8 – 255
Mapping Between WF_EVENT_T and OMBAQ_TEXT_MSG	8 – 257
Event APIs	8 – 260
Raise	8 – 261
Send	8 – 265
NewAgent	8 – 267
Test	8 – 268
Enqueue	8 – 269
Listen	8 – 270
Workflow Agent Listener Concurrent Program	8 – 272
SetErrorInfo	8 – 273
SetDispatchMode	8 – 274
AddParameterToList	8 – 275
AddParameterToListPos	8 – 276
GetValueForParameter	8 – 277

GetValueForParameterPos .....	8 – 278
Event Subscription Rule Function APIs .....	8 – 279
Default_Rule .....	8 – 281
Log .....	8 – 283
Error .....	8 – 284
Warning .....	8 – 285
Success .....	8 – 286
Workflow_Protocol .....	8 – 287
Error_Rule .....	8 – 288
SetParametersIntoParameterList .....	8 – 289
Event Function APIs .....	8 – 290
Parameters .....	8 – 291
SubscriptionParameters .....	8 – 293
AddCorrelation .....	8 – 294
Generate .....	8 – 296
Receive .....	8 – 298
Business Event System Replication APIs .....	8 – 300
WF_EVENTS Document Type Definition .....	8 – 302
WF_EVENTS_PKG.Generate .....	8 – 303
WF_EVENTS_PKG.Receive .....	8 – 304
WF_EVENT_GROUPS Document Type Definition .....	8 – 305
WF_EVENT_GROUPS_PKG.Generate .....	8 – 306
WF_EVENT_GROUPS_PKG.Receive .....	8 – 307
WF_SYSTEMS Document Type Definition .....	8 – 308
WF_SYSTEMS_PKG.Generate .....	8 – 309
WF_SYSTEMS_PKG.Receive .....	8 – 310
WF_AGENTS Document Type Definition .....	8 – 311
WF_AGENTS_PKG.Generate .....	8 – 312
WF_AGENTS_PKG.Receive .....	8 – 313
WF_EVENT_SUBSCRIPTIONS Document Type Definition .....	8 – 314
WF_EVENT_SUBSCRIPTIONS_PKG.Generate .....	8 – 315
WF_EVENT_SUBSCRIPTIONS_PKG.Receive .....	8 – 316

## Index



## VOLUME 2

<b>Chapter 9</b>	<b>Oracle Workflow Home Page</b> . . . . .	<b>9 – 1</b>
	Accessing the Oracle Workflow Home Page . . . . .	9 – 2
	Setting User Preferences . . . . .	9 – 6
<b>Chapter 10</b>	<b>Viewing Notifications and Processing Responses</b> . . . . .	<b>10 – 1</b>
	Overview of Notification Handling . . . . .	10 – 2
	Reviewing Notifications via Electronic Mail . . . . .	10 – 2
	Viewing Notifications from a Web Browser . . . . .	10 – 12
	Reviewing a Summary of Your Notifications via Electronic Mail . . . . .	10 – 24
	Defining Rules for Automatic Notification Processing . . . . .	10 – 25
<b>Chapter 11</b>	<b>Monitoring Workflow Processes</b> . . . . .	<b>11 – 1</b>
	Overview of Workflow Monitoring . . . . .	11 – 2
	Workflow Monitor . . . . .	11 – 2
	Workflow Monitor Access . . . . .	11 – 7
<b>Chapter 12</b>	<b>Testing a Workflow Definition</b> . . . . .	<b>12 – 1</b>
	Testing Workflow Definitions . . . . .	12 – 2
<b>Chapter 13</b>	<b>Managing Business Events</b> . . . . .	<b>13 – 1</b>
	Managing Business Events . . . . .	13 – 2
	Events . . . . .	13 – 4
	Systems . . . . .	13 – 17
	Agents . . . . .	13 – 22
	Event Subscriptions . . . . .	13 – 34
	Setting Up Message Propagation . . . . .	13 – 53
	Raising Events . . . . .	13 – 65
	Signing Up Systems . . . . .	13 – 67
	Synchronizing Systems . . . . .	13 – 70
	Reviewing Local Queues . . . . .	13 – 72
	Workflow Agent Ping/Acknowledge . . . . .	13 – 77
	The Workflow Agent Ping/Acknowledge Item Type . . . . .	13 – 78
	Summary of the Master Ping Process . . . . .	13 – 79
	Master Ping Process Activities . . . . .	13 – 80
	Summary of the Detail Ping Process . . . . .	13 – 81

Detail Ping Process Activities .....	13 – 82
--------------------------------------	---------

## Chapter 14

<b>Predefined Workflow Events .....</b>	<b>14 – 1</b>
Predefined Workflow Events .....	14 – 2
Event Definition Events .....	14 – 2
Event Group Definition Events .....	14 – 3
System Definition Events .....	14 – 4
Agent Definition Events .....	14 – 4
Event Subscription Definition Events .....	14 – 5
Synchronize Event Systems Event .....	14 – 5
Seed Event Group .....	14 – 6
Ping Agent Events .....	14 – 8
System Signup Event .....	14 – 9
Any Event .....	14 – 10
Unexpected Event .....	14 – 12
User Entry Has Changed Event .....	14 – 15
Workflow Send Protocol .....	14 – 17
The Workflow Send Protocol Item Type .....	14 – 18
Summary of the Workflow Event Protocol Process .....	14 – 20
Workflow Event Protocol Process Activities .....	14 – 21
Workflow Send Protocol Events .....	14 – 24

## Chapter 15

<b>Demonstration Workflow Processes .....</b>	<b>15 – 1</b>
Sample Workflow Processes .....	15 – 2
Displaying the Process Diagram of a Sample Workflow .....	15 – 3
Requisition Process .....	15 – 5
Installing the Requisition Data Model .....	15 – 6
Initiating the Requisition Workflow .....	15 – 8
The Requisition Item Type .....	15 – 12
Summary of the Requisition Approval Process .....	15 – 13
Requisition Process Activities .....	15 – 15
Summary of the Notify Approver Subprocess .....	15 – 19
Notify Approver Subprocess Activities .....	15 – 21
Sample StartProcess Function .....	15 – 23
Example Function Activities .....	15 – 26
Example: Select Approver .....	15 – 26
Example: Verify Authority .....	15 – 29
Example Notification Activity .....	15 – 31
Example: Notify Requisition Approval Required .....	15 – 31
Product Survey Process .....	15 – 34

Installing the Product Survey Data Model .....	15 – 35
Initiating the Product Survey Workflow .....	15 – 36
The Product Survey Item Type .....	15 – 38
Summary of the Survey – Single Process .....	15 – 39
Survey – Single Process Activities .....	15 – 41
Summary of the Survey – Master/Detail Process .....	15 – 42
Survey – Master/Detail Process Activities .....	15 – 44
Summary of the Detail Survey Process .....	15 – 46
Detail Survey Process Activities .....	15 – 47
Document Review Process .....	15 – 49
The Document Management Item Type .....	15 – 49
Summary of the Document Review Process .....	15 – 50
Document Review Process Activities .....	15 – 52
Error Check Process .....	15 – 54
The Periodic Alert Item Type .....	15 – 54
Summary of the Error Check Process .....	15 – 56
Error Check Process Activities .....	15 – 57
Summary of the User Defined Alert Action Process .....	15 – 60
User Defined Alert Action Process Activities .....	15 – 61
Event System Demonstration .....	15 – 63
Installing the Event System Demonstration Data Model ...	15 – 64
Initiating the Event System Demonstration Workflow .....	15 – 66
The Event System Demonstration Item Type .....	15 – 71
Summary of the Buyer: Top Level PO Process .....	15 – 73
Buyer: Top Level PO Process Activities .....	15 – 75
Summary of the Buyer: Send PO to Supplier Subprocess ...	15 – 78
Buyer: Send PO to Supplier Subprocess Activities .....	15 – 78
Summary of the Buyer: Receive Supplier PO Acknowledgement Subprocess .....	15 – 80
Buyer: Receive Supplier PO Acknowledgement Subprocess Activities .....	15 – 81
Summary of the Buyer: Advanced Shipment Notice Subprocess .....	15 – 83
Buyer: Advanced Shipment Notice Subprocess Activities ..	15 – 84
Summary of the Buyer: Receive Supplier Invoicing Subprocess .....	15 – 85
Buyer: Receive Supplier Invoicing Subprocess Activities ...	15 – 86
Summary of the Supplier: Top Level Order Process .....	15 – 87
Supplier: Top Level Order Process Activities .....	15 – 88
Summary of the Supplier: Get Order Details Subprocess ...	15 – 91
Supplier: Get Order Details Subprocess Activities .....	15 – 92
Summary of the Supplier: Credit Check Subprocess .....	15 – 94

Supplier: Credit Check Subprocess Activities . . . . .	15 – 95
Summary of the Supplier: Stock Check Subprocess . . . . .	15 – 96
Supplier: Stock Check Subprocess Activities . . . . .	15 – 97
Summary of the Supplier: Advanced Shipment Notice Subprocess . . . . .	15 – 98
Supplier: Advanced Shipment Notice Subprocess Activities . . . . .	15 – 99
Summary of the Supplier: Send Supplier Invoice Subprocess . . . . .	15 – 100
Supplier: Send Supplier Invoice Subprocess Activities . . . . .	15 – 101
B2B Purchase Order Event . . . . .	15 – 102
B2B Purchase Order Acknowledgement Event . . . . .	15 – 105
B2B Advanced Shipment Notice Event . . . . .	15 – 106
B2B Invoice Event . . . . .	15 – 107

## Chapter 16

<b>Workflow Administration Scripts . . . . .</b>	<b>16 – 1</b>
Miscellaneous SQL Scripts . . . . .	16 – 2
FNDWFLST . . . . .	16 – 4
FNDWFPR . . . . .	16 – 5
WFNLADD.sql . . . . .	16 – 5
Wfagtlst.sql . . . . .	16 – 6
Wfbkg.sql . . . . .	16 – 6
Wfbkgchk.sql . . . . .	16 – 7
Wfchact.sql . . . . .	16 – 7
Wfchacta.sql . . . . .	16 – 7
Wfchita.sql . . . . .	16 – 8
Wfchitt.sql . . . . .	16 – 8
Wfchluc.sql . . . . .	16 – 8
Wfchlut.sql . . . . .	16 – 9
Wfchmsg.sql . . . . .	16 – 9
Wfchmsga.sql . . . . .	16 – 9
Wfdirchk.sql . . . . .	16 – 10
Wfevtenq.sql . . . . .	16 – 10
Wfjvstop.sql . . . . .	16 – 11
Wfmqupd.sql . . . . .	16 – 12
Wfnlena.sql . . . . .	16 – 12
Wfntfsh.sql . . . . .	16 – 12
Wfprot.sql . . . . .	16 – 12
Wfqclean.sql . . . . .	16 – 13
Wfrefchk.sql . . . . .	16 – 13
Wfretry.sql . . . . .	16 – 13
Wfrmall.sql . . . . .	16 – 14

Wfrmita.sql	16 – 14
Wfrmitms.sql	16 – 15
Wfrmitt.sql	16 – 15
Wfrmtime.sql	16 – 15
Wfrun.sql	16 – 15
Wfstat.sql	16 – 16
Wfstatus.sql	16 – 16
Wfstdchk.sql	16 – 16
Wfver.sql	16 – 16
Wfverchk.sql	16 – 17
Wfverupd.sql	16 – 17

<b>Appendix A</b>	<b>Oracle Workflow Builder Menus and Toolbars</b>	<b>A – 1</b>
	Oracle Workflow Builder Menus	A – 2
	Oracle Workflow Builder Toolbars	A – 7

<b>Appendix B</b>	<b>Oracle Workflow Implementation in Other Oracle Products</b>	<b>B – 1</b>
	Predefined Workflows Embedded in Oracle E–Business Suite	B – 2
	Oracle Workflow Business Event System Implementation in Oracle E–Business Suite	B – 16
	Oracle Workflow Implementation in the Oracle9i Platform	B – 18
	Oracle Support Policy for Predefined Workflows, Events, and Subscriptions	B – 20
	Customization Guidelines	B – 20
	Resolving Customization Issues	B – 21
	What Is NOT Supported	B – 21
	What Is Supported	B – 21

<b>Appendix C</b>	<b>Oracle Workflow Performance Concepts</b>	<b>C – 1</b>
	Oracle Workflow Performance Concepts	C – 2
	Designing Workflow Processes for Performance	C – 2
	Managing Runtime Data for Performance	C – 8

## Glossary

## Index



CHAPTER

# 9

## Oracle Workflow Home Page

**T**his 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.

**Note:** If Oracle Internet Directory/Single Sign-On integration has been implemented for your installation of Oracle Workflow, you can use single sign-on when accessing Oracle Workflow's web-based features. With single sign-on, a user who is logged into any participating Oracle*iAS* component is automatically authenticated when accessing any other participating component and does not need to log in again. See: Synchronizing Workflow Directory Services with Oracle Internet Directory: page 2 – 30.

### ► To Access the Oracle Workflow Home Page

1. Use a web browser to connect to the URL for the home page:

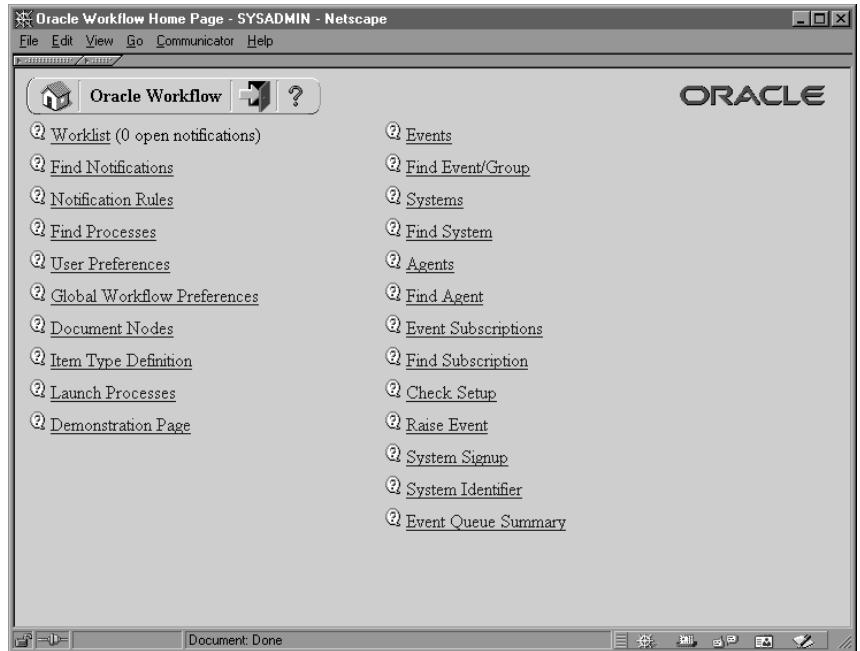
`<webagent>/wfa_html.home`

`<webagent>` represents the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



**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.
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. Some of the Event Manager web pages also include a Query icon that lets you enter query details to search for Event Manager objects.
4. Choose the Worklist link to display your list of workflow notifications. 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 – 15.
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 – 26.

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 – 9.
8. Choose the User Preferences link to set the preferences that control how you interact with Oracle Workflow. See: Setting User Preferences; page 9 – 6.
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 – 14.
10. If you are logged in as a role with workflow administrator privileges, the Document Nodes link appears. This functionality is reserved for future use. You do not need to perform any actions with this link.
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 – 24.
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 15 – 2.
14. If you are logged in as a role with workflow administrator privileges, you can choose the Events link to view and define Business Event System events. See: To Define an Event; page 13 – 5.
15. If you are logged in as a role with workflow administrator privileges, you can choose the Find Event/Group link to query for

events and event groups that match certain search criteria. See: To Find Events: page 13 – 14.

16. If you are logged in as a role with workflow administrator privileges, you can choose the Systems link to view and define Business Event System systems. See: To Define a System: page 13 – 18.
17. If you are logged in as a role with workflow administrator privileges, you can choose the Find System link to query for systems that match certain search criteria. See: To Find Systems: page 13 – 19.
18. If you are logged in as a role with workflow administrator privileges, you can choose the Agents link to view and define Business Event System agents. See: To Define an Agent: page 13 – 29.
19. If you are logged in as a role with workflow administrator privileges, you can choose the Find Agent link to query for agents that match certain search criteria. See: To Find Agents: page 13 – 32.
20. If you are logged in as a role with workflow administrator privileges, you can choose the Event Subscriptions link to view and define Business Event System subscriptions. See: To Define an Event Subscription: page 13 – 45.
21. If you are logged in as a role with workflow administrator privileges, you can choose the Find Subscription link to query for subscriptions that match certain search criteria. See: To Find Event Subscriptions: page 13 – 50.
22. If you are logged in as a role with workflow administrator privileges, you can choose the Check Setup link to check your Business Event System setup and schedule listeners and propagations for local agents. See: Setting Up Message Propagation: page 13 – 53.
23. If you are logged in as a role with workflow administrator privileges, you can choose the Raise Event link to raise a business event to the Event Manager. See: Raising Events: page 13 – 65.
24. If you are logged in as a role with workflow administrator privileges, you can choose the System Signup link to sign up one system with another to receive business events. See: Signing Up Systems: page 13 – 67.
25. If you are logged in as a role with workflow administrator privileges, you can choose the System Identifier link to retrieve the

system identifier information required for signing up systems. See: To Retrieve System Identifier Information: page 13 – 68.

26. If you are logged in as a role with workflow administrator privileges, you can choose the Event Queue Summary link to review the local queues used by the Business Event System. See: Reviewing Local Queues: page 13 – 72.

---

## 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 – 14.



**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 Oracle8i-compliant date format to use for your database session. An example of an Oracle8i-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. Leave the Document Home Node field blank. This functionality is reserved for future use.
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 – 49

CHAPTER

# 10

## Viewing Notifications and Processing Responses

**T**his 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 – 21 and *Setting User Preferences*: page 9 – 6.

Each notification message can include context-sensitive information about the process and directions on how to respond to the notification, if a response is required. The message can also include pointers to Web URLs and references to Oracle Applications forms that allow the user to get additional information related to the notification.

As a notification recipient, there may be occasions when you will not be able to view or respond to your notifications in a timely manner. Rather than create a bottleneck in a workflow process, you can take advantage of the Automatic Notification Handler to define rules that direct Oracle Workflow to automatically 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. For e-mail notifications that do not require a response, such as FYI (For Your Information) notifications, your workflow administrator determines how the notification status is updated when setting up the Notification Mailer. Depending on the setup configuration, either Oracle Workflow automatically updates the status of FYI notifications to 'Closed' after sending you the notifications by e-mail, or those notifications maintain an 'Open' status until you manually close them in the Notifications Worklist web page.

Once you read an FYI message, you can delete it from your inbox. However, if the Notification Mailer for your organization is set up to keep FYI notifications open after sending them by e-mail, you must also use the Notifications Worklist to manually close the notification, even if you have already deleted the notification message from your e-mail inbox. See: To View Notifications from the Worklist: page 10 – 17.

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 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 – 69.

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

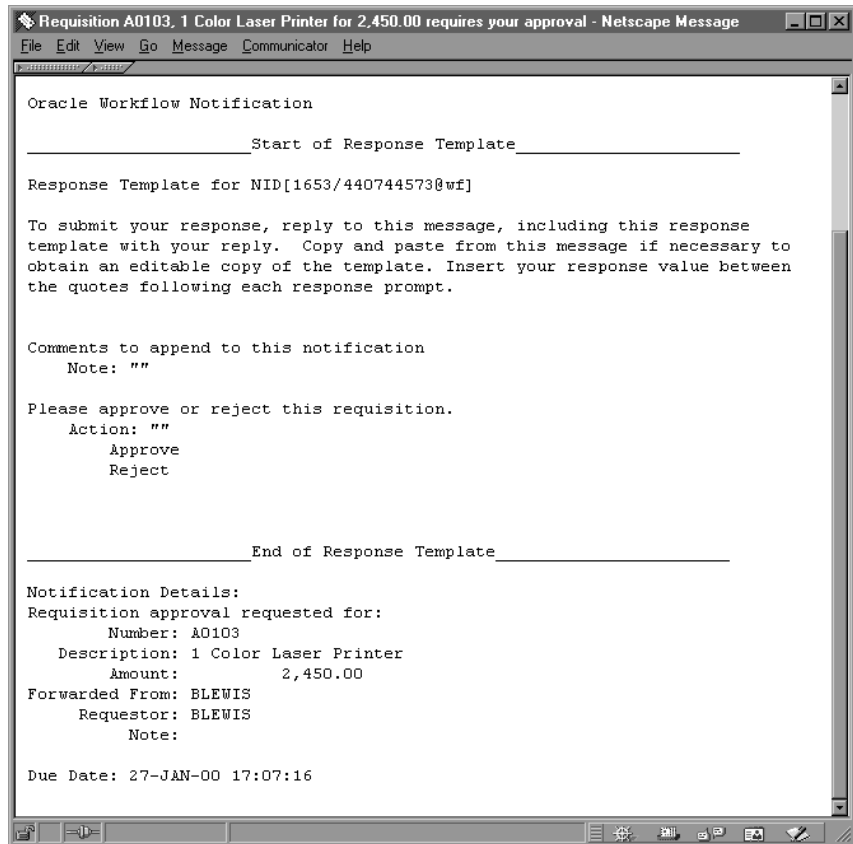
in your reply or by using a response template that includes the NID line.

**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 – 55

### ► To Respond to a Plain Text E-mail Notification Using Templated Response



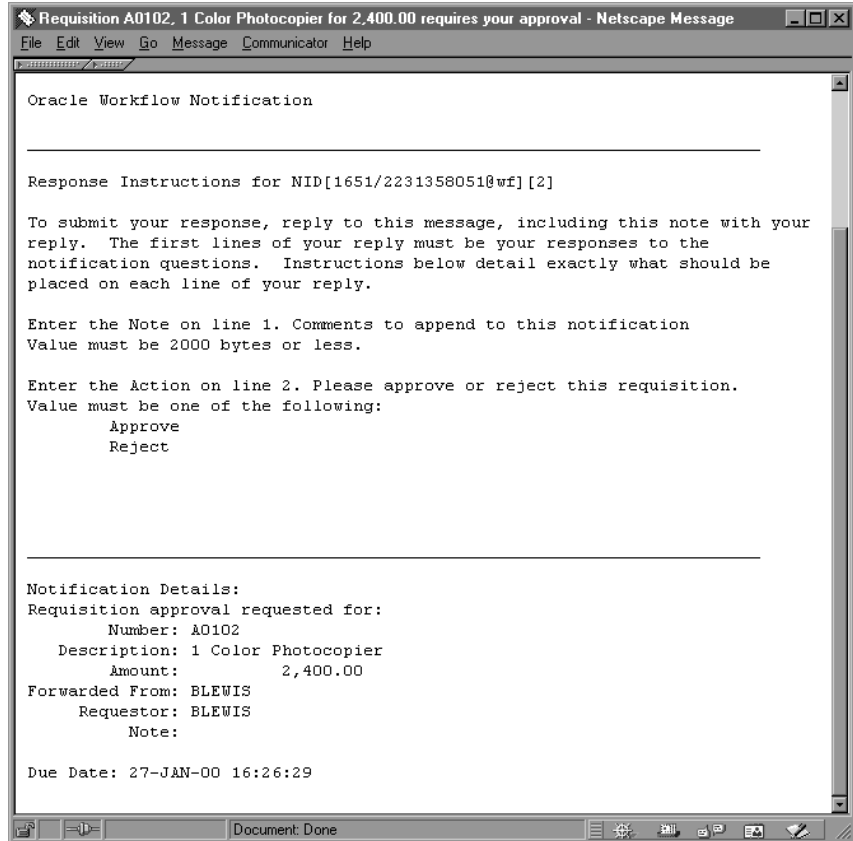
1. Your plain text e-mail notification includes information that is helpful for you to respond to the notification. Depending on the notification, the information may appear as references to other sources or as attachments. Some attachments, depending on their content may only be viewable if you display your notification using the Notification web pages. See: Viewing Notifications from a Web Browser: page 10 – 12.
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 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 – 50

► **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 – 12.
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.

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.

### Valid Response A – Approve

Approve
Let me know if this item meets expectations.
01-JAN-1998
1000.00

### Valid Response B – Reject

Reject
Too expensive.

**Note:** The blank lines in this response indicate acceptance of the default values for those response values.

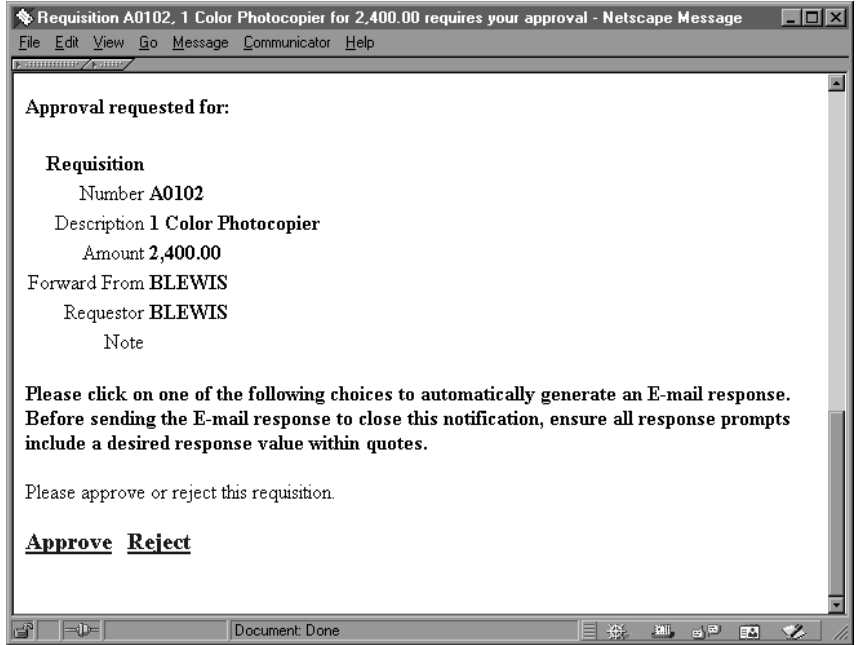
### Valid Response C – Reject

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

### See Also

Plain Text E-mail: page 2 – 50

► **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.

If your workflow administrator has configured the Notification Mailer to require a login when you select the Notification Detail Link, and you are not already logged in, you will be prompted to log in before you can access the Notification Details web page.

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 – 19.

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.

**Note:** If your workflow administrator has configured the Notification Mailer to use the Workflow Open Mail for Outlook Express message template, a link called "Click here to respond" will appear in place of the links for individual responses. The "Click here to respond" link provides access to your notification in the Notification Details web page.

Note that your Web browser must support JavaScript and Frames to open this link. When you choose the "Click here to respond" link, 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. If you are disconnected from your network, however, you will not be able to use the link.

Once you have accessed the Notification Details page, you can respond directly to your notification from this page. In this case, you should skip the remaining steps.

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 quotes ( ' ') following each response prompt.

**Note:** Response templates generated from a response link in an HTML-formatted e-mail notification provide single quotes to enclose your response values, rather than the double quotes



provided in plain text response templates. This use of single quotes accommodates e-mail applications that cannot process double quotes in the <A HREF="mailto:"> tag for the link but can accept single quotes.

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 – 51

### ► **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 – 9.

- Choose the Notification Detail Link attachment to display the Notification Details web page. See: To View the Details of a Notification: page 10 – 19.

**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.

If your workflow administrator has configured the Notification Mailer to require a login when you select the Notification Detail Link, and you are not already logged in, you will be prompted to log in before you can access the Notification Details web page.

## See Also

Plain Text E-mail with an HTML Attachment: page 2 – 53

### ► 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 – 12.

---

## 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 – 14.
  - *<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, and then by BEGIN\_DATE in descending order (the notifications with the highest priority and the most recent date will be displayed first).
  - *<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 – 14.
- 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.
  - For Oracle Applications users only, your system administrator can add the Notifications Worklist web page to your application by using the Oracle Applications function FND\_FNDWFNOT. This function calls the web page `wfa_html.worklist`. An Oracle Applications System Administrator or Oracle Applications developer must add this function to the Navigate menu of a user's responsibility or call this function from an Oracle Applications form. You can then use the modified menu or form to navigate to the Notifications Worklist web page. See: *Overview of Function Security, Oracle Applications System Administrator's Guide, Overview of Menus and Function Security, Oracle Applications Developer's Guide, Menus Window, Oracle Applications Developer's Guide, and Overview of Form Development Steps, Oracle Applications Developer's Guide.*
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, you will be prompted 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

The screenshot shows a Netscape browser window titled "Find Notifications - CDOUGLAS - Netscape". The browser's menu bar includes "File", "Edit", "View", "Go", "Communicator", and "Help". The page header features a home icon, the text "Find Notifications", a search icon, and a help icon. The Oracle logo is positioned in the top right corner. The main content area contains a search form with the following fields and options:

- From:** A text input field.
- Status:** A dropdown menu with "Open" selected.
- Type:** A dropdown menu with "All" selected.
- Subject:** A wide text input field.
- Sent:** Two text input fields separated by a hyphen.
- Due:** Two text input fields separated by a hyphen.
- Priority:** A dropdown menu with "All" selected.
- Notifications Delegated to:** A checkbox followed by a text input field.

A "Find" button is located at the bottom right of the form area.

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:
  - **From** – enter a role to search for all notifications from that role. The From role for a notification is determined by the #FROM\_ROLE message attribute. See: #FROM\_ROLE Attribute: page 4 – 25.
  - **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 criterion 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 – 24.
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 – 14

Find Notifications - CDOUGLAS - Netscape

File Edit View Go Communicator Help

Find Notifications ?

ORACLE

Notification ID

OR

Owner

To

From

Status

Type

Subject

Sent  -

Due  -

Priority

Find

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:

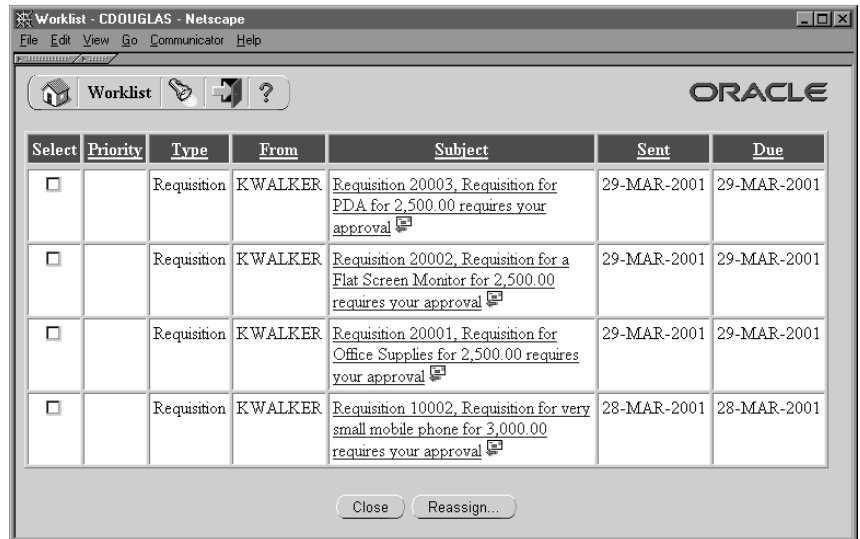
- Notification ID—enter a specific notification ID. Note that if you specify a notification ID, all other search criteria are ignored.

- 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 – 24.
- 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 – 24.

**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.

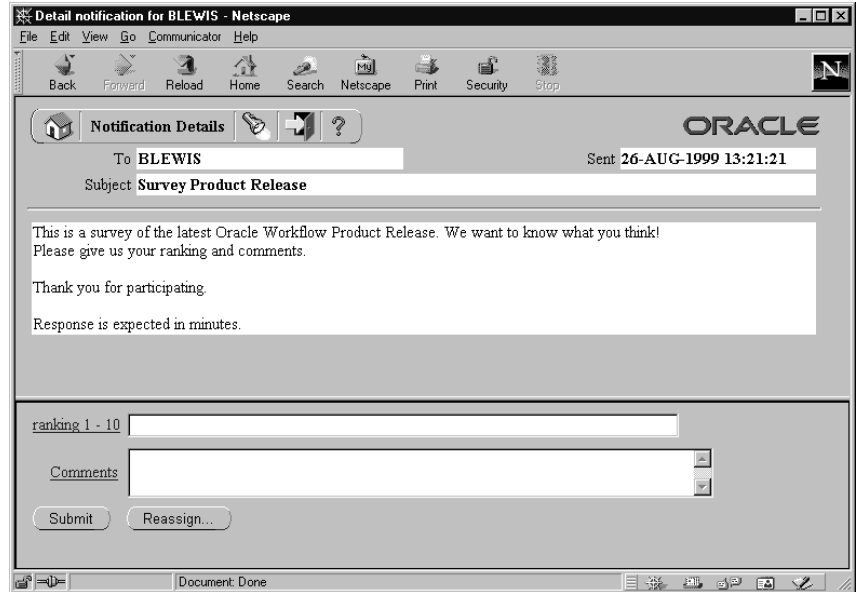
The Worklist displays the following information for each notification:

- 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.
  - From—the role from which the notification was sent. The From role for a notification is determined by the #FROM\_ROLE message attribute. See: #FROM\_ROLE Attribute: page 4 – 25.
  - 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: page 10 – 22.
  6. You can navigate to the full details of any notification and act on the notification by clicking 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. A reference URL link opens another Web browser window and connects to a specified URL.
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 three types of attachment links:
  - A reference URL link that opens another Web browser window and connects to a specified URL.
  - A PL/SQL or PL/SQL CLOB document link that displays the contents of a document generated from a PL/SQL function.
  - 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. The Oracle Applications socket listener must be activated, and the socket listener port must be set to the port where the form should be launched. See: *Setting the Socket Listener Profile Options: page 2 – 40.*



**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 to that effect is displayed. Also, you cannot update information in a form that is attached in the upper frame for reference only.

When you choose the attached form icon, a launch window appears to make the socket connection. You can choose the Keep Window button to continue displaying the launch window. Otherwise, this window is automatically closed after thirty seconds.



The form that is launched appears in the Oracle Applications Navigator window on top of the browser window displaying the notification.

**Note:** If the form does not appear, check for error messages in the JInitiator console.

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:** 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.



The Oracle Applications socket listener must be activated, and the socket listener port must be set to the port where the form should be launched. See: Setting the Socket Listener Profile Options: page 2 – 40.

When you choose the attached form icon, a launch window appears to make the socket connection. You can choose the Keep Window button to continue displaying the launch window. Otherwise, this window is automatically closed after thirty seconds.



The form that is launched appears in the Oracle Applications Navigator window on top of the browser window displaying the notification.

**Note:** If the form does not appear, check for error messages in the JInitiator console.

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.

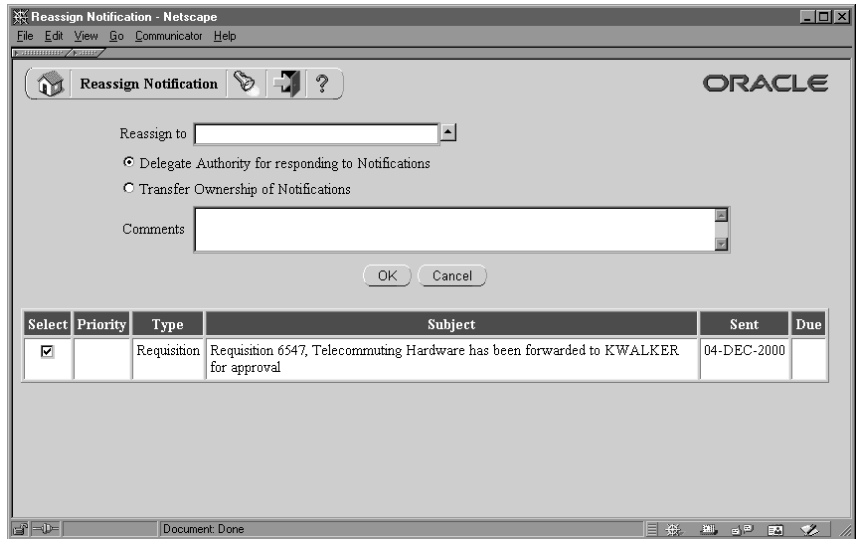
6. A Find icon in the toolbar of the Detail Notification page lets you navigate back to the Find Notifications screen at any time to search for and display other notifications.

### ► 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.



**Attention:** Your workflow may include a special #HIDE\_REASSIGN attribute to restrict reassignment of notifications. If so, the Reassign button may not be displayed in the Response frame, and you should not reassign the notification. See: #HIDE\_REASSIGN Attribute: page 4 – 25.



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 – 24.
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 may include special logic called a post-notification function 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. See: Post-Notification Functions: page 8 – 13.

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. For a field that supports a list of values, click on the field's up-arrow icon to display a list of values window.
2. In the Find field, enter search criteria and choose the Find button to retrieve a subset of values that match your criteria. You can also choose the Clear button to clear the Find field. If you do not specify any search criteria and simply choose Find, you retrieve the complete list of values.
3. Click on a value from the list to select that value and close the list of values window. The value you select populates the original field.

---

## **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 – 55.

You can receive your e-mail notification summary using any e-mail reader. The following example shows a notification summary received through Netscape Messenger as the mail client.



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 – 14.
- *<rolename>* represents an internal role name that you want to query routing rules for. Note, however, that you can query for



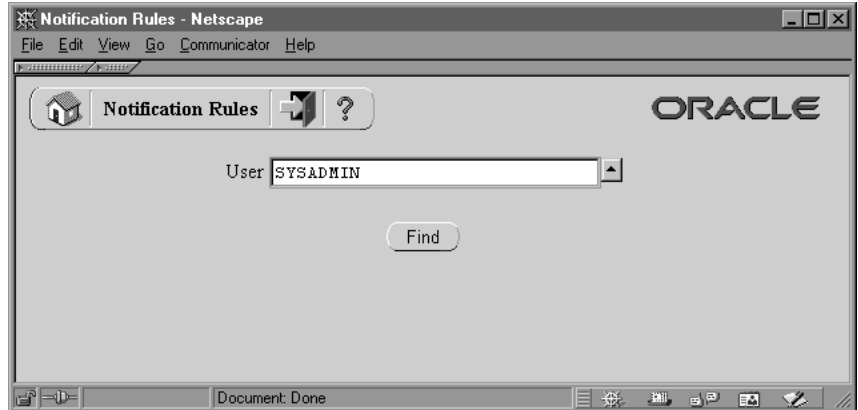
roles other than your current role only if your current role has workflow administrator privileges. See: Setting Global User Preferences: page 2 – 14.

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.



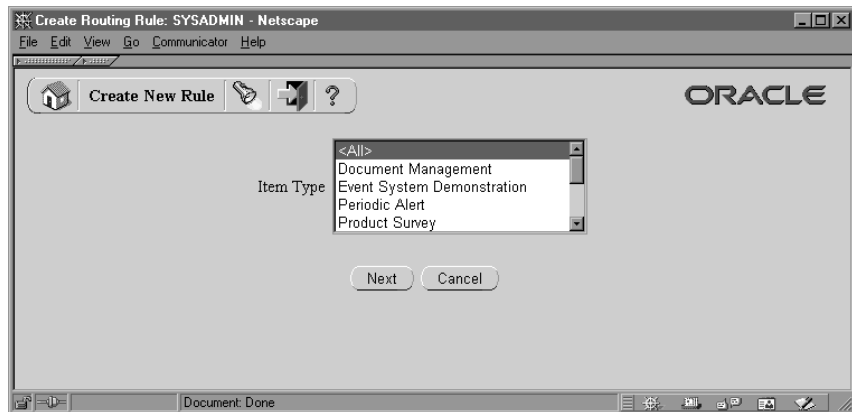
**Attention:** Both of these URLs access secured pages, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears.

**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.

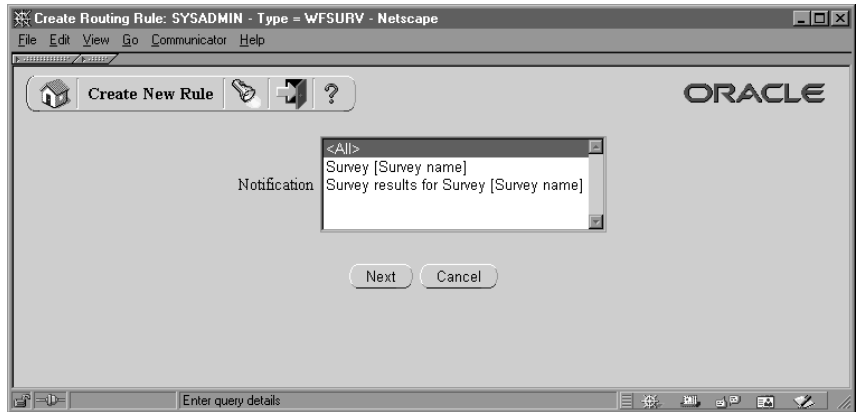
2. The Notification Rules page for the role appears, listing all existing rules for the current role. Choose Create Rule.



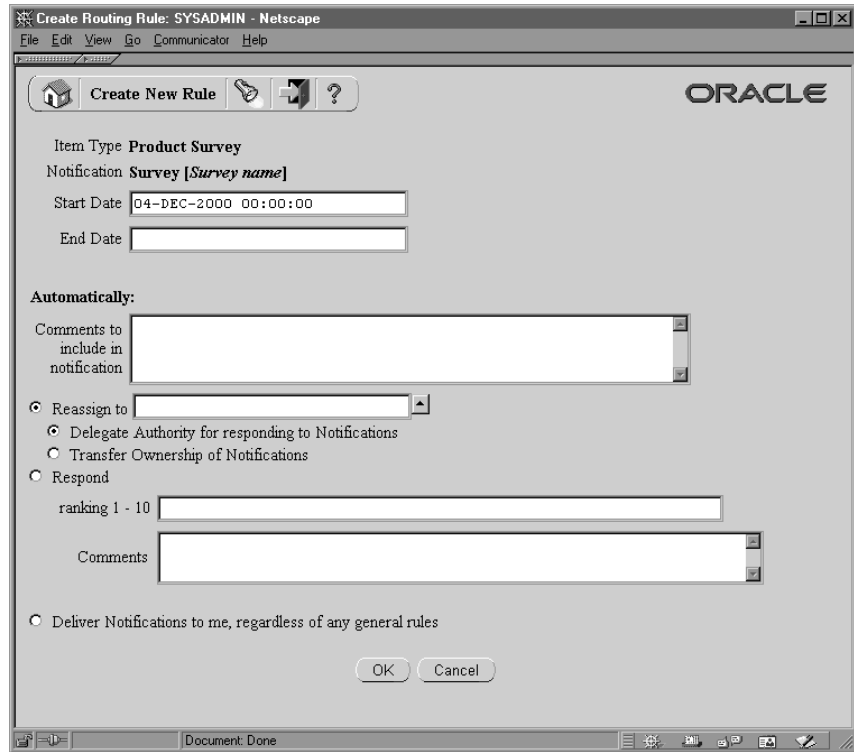
3. In the Item Type poplist field, select the item type to which this rule applies or select <All> if you want this rule to apply to notifications associated with any item type.



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 – 24.
  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 – 14.

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.

CHAPTER

# 11



## Monitoring Workflow Processes

**T**his 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.

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager as an additional administration tool to review and manage work items. For more information, please refer to the Oracle Applications Manager online help.

Also, if you are using the standalone version of Oracle Workflow available with Oracle9i Release 2, you can use the standalone Oracle Workflow Manager component available through Oracle Enterprise Manager as an additional administration tool to review and manage work items. For more information, please refer to the Oracle Workflow Manager online help.

### See Also

Oracle Workflow Views: page 8 – 157

---

## Workflow Monitor

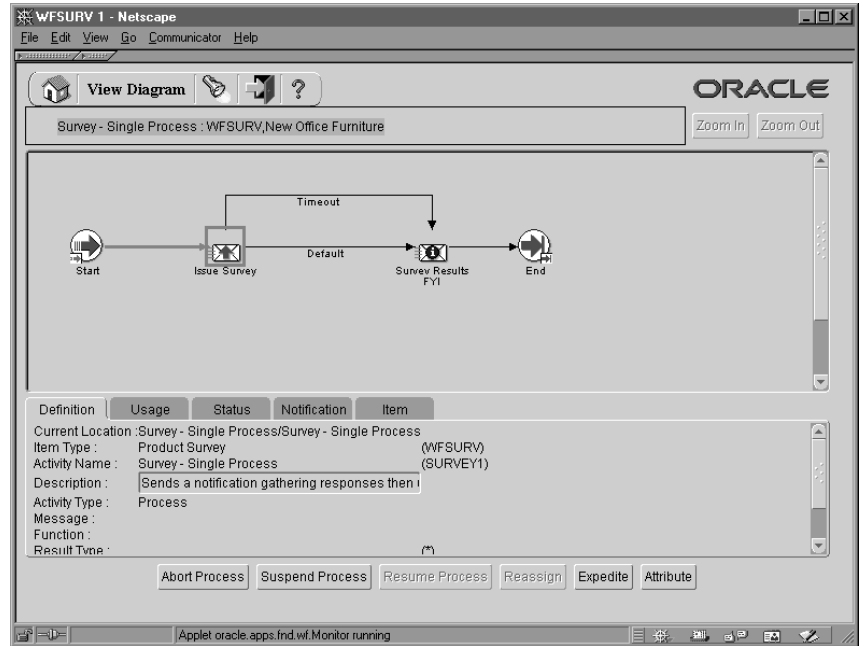
The Workflow Monitor is a tool that allows you to view and administer the status of a specific instance of a workflow process. You can use the point-and-click interface to display detailed status information about activities in the process as well as about the process as a whole. The Workflow Monitor 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. The following table shows what state each color indicates.

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, Event, 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 – 36.
- **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 – 32.

- **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 – 34.
- **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 – 74.
- **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 – 77.
- **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 1.1.8 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 – 151.

## **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 1.1.8 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 HTTP 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 – 110.

**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.

## **Access to the Workflow Monitor from Oracle Applications**

---

For Oracle Applications only, you can add access to the Workflow Monitor to your application by using the Oracle Applications function FND\_FNDWFAS. This function calls the web page wf\_monitor.show.

You can add this function to the Navigate menu of a user's responsibility or call the function from an Oracle Applications form. See: *Menus Window, Oracle Applications Developer's Guide* and *Overview of Form Development Steps, Oracle Applications Developer's Guide*.

If you call FND\_FNDWFAS without passing any parameters in the call, then wf\_monitor.show displays the Find Processes web page.

If you want to specify a process instance to display, you must pass the function the following parameters:

- ITEM\_TYPE—A valid item type.

- `ITEM_KEY`—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.
- `ADMIN_MODE`—`YES` to run the monitor in 'ADMIN' mode, or `NO` to run the monitor in 'USER' mode.
- `ACCESS_KEY`—The access key password to run the monitor in the selected mode. Use the Workflow Monitor API `GetAccessKey()` to retrieve the appropriate access key. See: `GetAccessKey`: page 8 – 150.

When you pass these parameters, `wf_monitor.show` displays the Workflow Monitor Notifications List web page for the specified process instance.

You can call the function `FND_FUNCTION.EXECUTE` to execute `FND_FNDWFIAS` specifying your parameters. See: `FND_FUNCTION.EXECUTE`, *Oracle Applications Developer's Guide*. Use the following syntax:

```
fnd_function.execute(
    FUNCTION_NAME=>'FND_FNDWFIAS',
    OTHER_PARAMS=>'ITEM_TYPE=' || <item_type> ||
    ' ITEM_KEY=' || <item_key> || ' ADMIN_MODE=' || <admin_mode> ||
    ' ACCESS_KEY=' || (wf_monitor.GetAccessKey('<item_type>',
    '<item_key>', '<admin_mode>' ));
```

The function `FND_FNDWFIAS` uses Oracle Applications function security to control user access. See: `Overview of Function Security`, *Oracle Applications System Administrator's Guide* and `Overview of Menus and Function Security`, *Oracle Applications Developer's Guide*.

**Note:** In Oracle Applications, you can also call the function `FND_UTILITIES.OPEN_URL` to open a web browser and have it connect to a specified URL, such as the Find Processes URL or a Workflow Monitor diagram URL. However, this function does not use Oracle Applications function security. See: `FND_UTILITIES:Utility Routine`, *Oracle Applications Developer's Guide*.

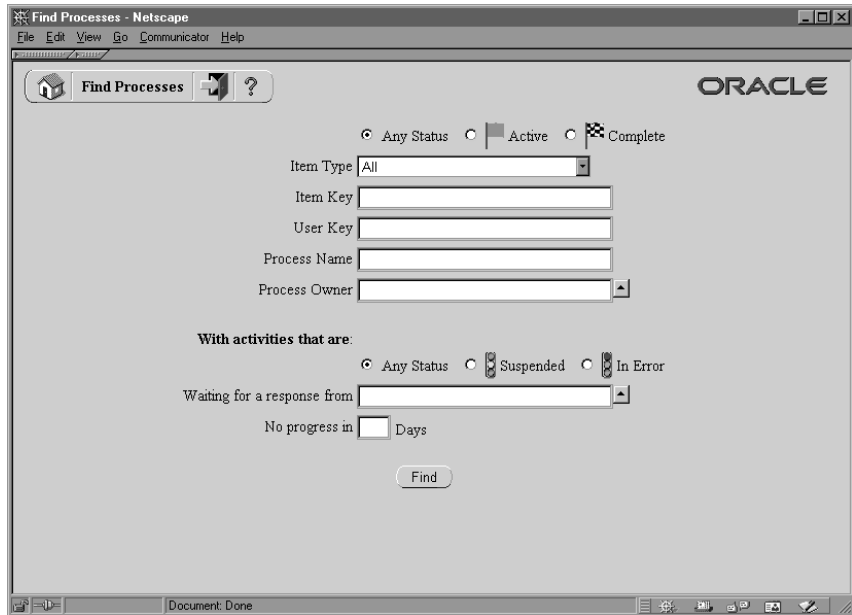
## Using the Find Processes Web Page

---

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 Workflow Monitor 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 – 23.
  - Process Name—Specify the display name of a process activity.



- 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.

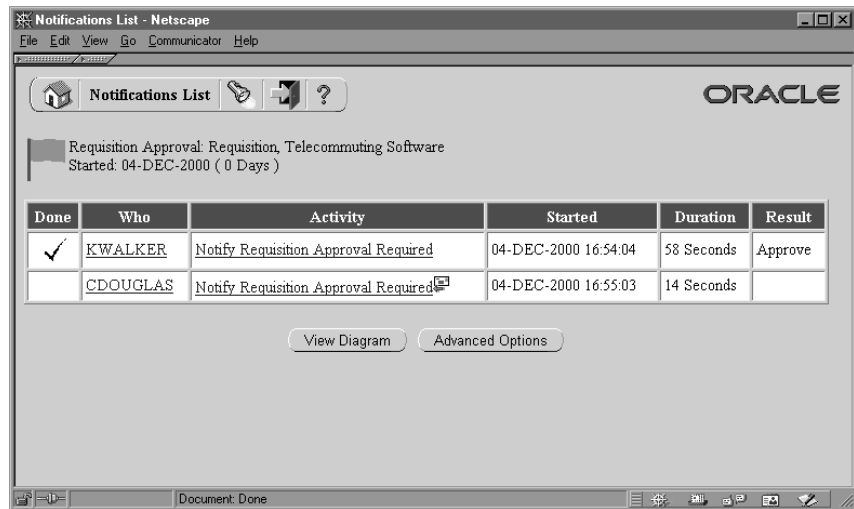
- Optionally, you can also find workflow process instances with activities that are Suspended, In Error, or that have Any Status.
- You can find workflow process instances that have activities waiting for a response from a particular user or role.
- You can also identify workflow process instances that have not progressed for a specified number of days.
- 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
Requisition	1950	Telecommuting Software	<a href="#">Requisition Approval</a>				04-DEC-2000 16:54:03
Requisition	6547	Telecommuting Hardware	<a href="#">Requisition Approval</a>	🚩			04-DEC-2000 16:28:47

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.

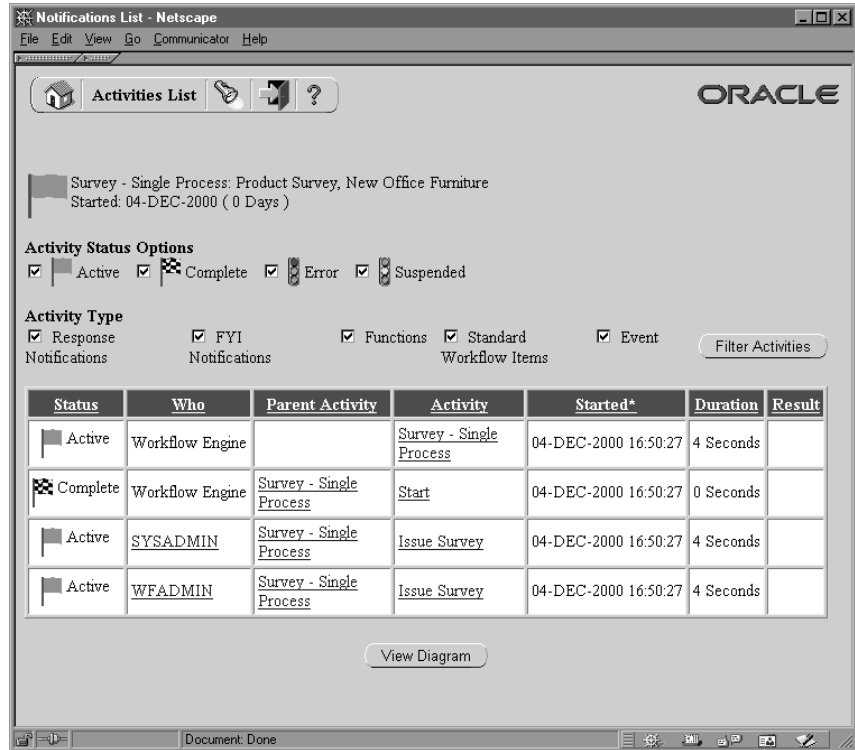
3. Choose a user link in the Who column if you want to send e-mail to the user that a notification has been assigned to.  

**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, function activities, activities that belong to the Standard item type, and/or event activities.
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 – 21

Setting Global User Preferences: page 2 – 14

CHAPTER

# 12

## Testing a Workflow Definition

**T**his 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 15 – 6.
- 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 web server documentation for more information.
- If you plan to use e-mail to view notifications, you can send all notifications to a single test e-mail address by setting the TEST\_ADDRESS parameter in the Notification Mailer configuration file. See: *To create a configuration file for the Notification Mailer*: page 2 – 58.

► **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.



Item Type	Internal Name	Description
<a href="#">Periodic Alert</a>	WFALEERT	\$Header: wfalert.wft 26.1 2000/10/19 01:09:24 kma ship \$ Periodic Alert process sample. Checks for errors and sends summary FYI notification when errors are detected.
<a href="#">Requisition</a>	WFDEMO	\$Header: wfdemo.wft 26.1 2000/10/19 01:09:39 kma ship \$ Requisition Approval demo for Oracle Workflow Version 2.5 Standalone
<a href="#">Document Management</a>	WFDM	\$Header: wfdm.wft 26.1 2000/10/19 01:09:46 kma ship \$ Document Management Integration demo for Oracle Workflow
<a href="#">Event System Demonstration</a>	WFEVDEME	Event System Demonstration: One Business Event System acts as a Buyer system, and another Business Event System acts as a Supplier system.
<a href="#">Workflow Agent Ping/Acknowledge</a>	WFPING	Master/Detail processes which ping all inbound agents and wait for acknowledgement message before completing
<a href="#">Workflow Send Protocol</a>	WFSNDPRT	Receives Messages from the Event System, and routes the Agent to the agent defined in the Event Subscription. If required, the workflow process will wait for an acknowledgment message and/or send an acknowledgment message.
<a href="#">Product Survey</a>	WFSURV	\$Header: wfsvr.wft 26.1 2000/10/19 01:09:55 kma ship \$ Product Survey demo for Oracle Workflow



**Attention:** Note that you can also connect to this page directly using the secured URL:

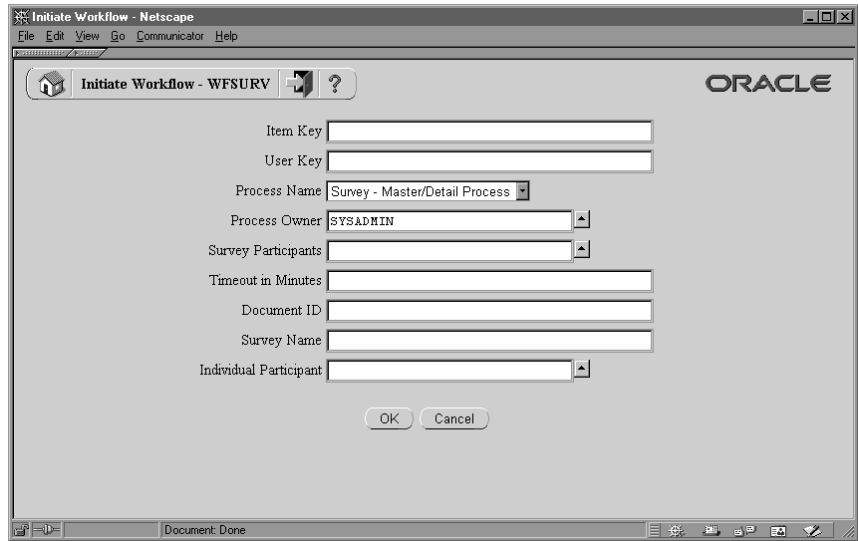
`<webagent>/wf_initiate.ItemType`

`<webagent>` represents the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



**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.

- The Launch Processes page displays all the item type definitions stored in the database except the Oracle Workflow seeded item types: Wferror, Wfmail, and Wfst. 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



## Managing Business Events

**T**his chapter tells you how to manage business events using the Oracle Workflow Event Manager web pages.

---

## Managing Business Events

The Oracle Workflow Business Event System is an application service that leverages the Oracle Advanced Queuing (AQ) infrastructure to communicate business events between systems. The Business Event System consists of the Event Manager and workflow process event activities.

The Event Manager contains a registry of business events, systems, named communication agents within those systems, and subscriptions indicating that an event is significant to a particular system. Events can be raised locally or received from an external system or the local system through AQ. When a local event occurs, the subscribing code is executed in the same transaction as the code that raised the event, unless the subscriptions are deferred.

Subscriptions can include the following types of processing:

- Executing custom code on the event information
- Sending event information to a workflow process
- Sending event information to other queues or systems

Business events are represented within workflow processes by event activities. By including event activities in a workflow process, you can model complex processing or routing logic for business events beyond the options of directly running a predefined function or sending the event to a predefined recipient. See: Event Activity: page 4 – 45.

The uses of the Business Event System include:

- **System integration messaging hubs**—Oracle Workflow with the Business Event System can serve as a messaging hub for complex system integration scenarios. The Event Manager can be used to “hard-wire” routing between systems based on event and originator. Workflow process event activities can be used to model more advanced routing, content-based routing, transformations, error handling, and so on.
- **Distributed applications messaging**—Applications can supply Generate and Receive event message handlers for their business entities. For example, message handlers can be used to implement Master/Copy replication for distributed applications.
- **Message-based system integration**—You can set up subscriptions which cause messages to be sent from one system to another when business events occur. In this way, you can use the Event Manager to implement point-to-point messaging integration.

- **Business–event based workflow processes**—You can develop sophisticated workflow processes that include advanced routing or processing based on the content of business events.
- **Non–invasive customization of packaged applications**—Analysts can register interesting business events for their internet or intranet applications. Users of those applications can register subscriptions to those events to trigger custom code or workflow processes.

## Event Manager

---

The Oracle Workflow Event Manager lets you register interesting business events that may occur in your applications, the systems among which events will be communicated, named communication agents within those systems, and subscriptions indicating that an event is significant to a particular system. You can use the Event Manager web pages to define and maintain these events, systems, agents, and subscriptions.

**Note:** You must have workflow administrator privileges to access the Event Manager web pages.

You can use the Workflow XML Loader to upload and download XML definitions for Business Event System objects between a database and a flat file. See: Using the Workflow XML Loader: page 2 – 112.

When an event is raised by a local application or received from a local or external system, the Event Manager executes any subscriptions to that event. Depending on the action defined in the subscription, the Event Manager may call custom code, send the event information to a workflow process, or send the event information to an agent.

The Event Manager also lets you complete your setup for message propagation, including scheduling listeners for inbound event messages and propagations for outbound event messages.

After you finish setting up the Business Event System, you can use the Event Manager to raise events manually, sign up systems to receive business events from each other, synchronize systems with each other, and review your local queues. You can test your setup using Workflow Agent Ping/Acknowledge.

---

## Events

A business event is an occurrence in an internet or intranet application or program that might be significant to other objects in a system or to external agents. For instance, the creation of a purchase order is an example of a business event in a purchasing application. You can define your significant events in the Event Manager.

Oracle Workflow provides several predefined events for significant occurrences within the Business Event System. See: Predefined Workflow Events: page 14 – 2

When an event occurs in an application on your local system, an event key must be assigned to uniquely identify that particular instance of the event. Then the event must be raised to the Event Manager.

You can raise an event by any of the following methods:

- Raise the event from the application where the event occurs using the *WF\_EVENT.Raise()* API. See: Raise: page 8 – 261.
- Raise the event from a workflow process using a Raise event activity. See: Event Activity: page 4 – 54.
- Raise the event manually using the Raise Events page. See: Raising Events: page 13 – 65.

Additionally, the Event Manager can receive events sent from the local system or remote systems.

When you define an event in the Event Manager, you must assign it a unique internal name, which is case-sensitive. The suggested format for these internal names is a compound structure of identifiers separated by periods (.) as follows:

```
<company>.<family>.<product>.<component>.<object>.<event>
```

This format allows you to organize the events you define into a classification hierarchy.

You can also define event groups that let you associate any events you want with each other and reference them as a group in event subscriptions. An event group is a type of event composed of a set of individual member events. The internal names of event groups should follow the same format as the names of individual events. Once you have defined an event group, you can register a subscription on the group rather than having to create separate subscriptions for each individual event within it. The subscription will be executed whenever any one of the group's member events occurs.



**Note:** Event groups cannot be used to raise events. You must raise each event individually.

Any detail information needed to describe what occurred in an event, in addition to the event name and event key, is called the event data. For example, the event data for a purchase order event includes the item numbers, descriptions, and cost. The event data can be structured as an XML document.

The application where the event occurs can include the event data when raising the event to the Event Manager. If the application will not provide the event data, you should specify a Generate function for the event that can produce the complete event data from the event name, event key, and an optional parameter list. The Generate function must follow a standard API. See: Raise: page 8 – 261 and Standard API for an Event Data Generate Function: page 7 – 21.

The Event Manager checks each subscription before executing it to determine whether the subscription requires the event data. If the event data is required but is not already provided, the Event Manager calls the Generate function for the event to produce the event data. If the event data is required but no Generate function is defined for the event, Oracle Workflow creates a default set of event data using the event name and event key.

**Note:** If the Generate function is costly, and you want to return control to the calling application more quickly after raising the event, you can defer all the subscriptions that require the complete event data. Then the Event Manager will not run the Generate function until those subscriptions are executed at a later time. See: Deferred Subscription Processing: page 13 – 41.

If you use a program to create event definitions automatically, the program can set its own name and brief identifier as the owner name and owner tag for the events. The program can then use this identifying information to locate the events that it owns. You can use the Edit Event and Edit Group pages to update the owner name and owner tag manually if necessary.

## ► To Define an Event

1. Use a web browser to connect to the following URL:

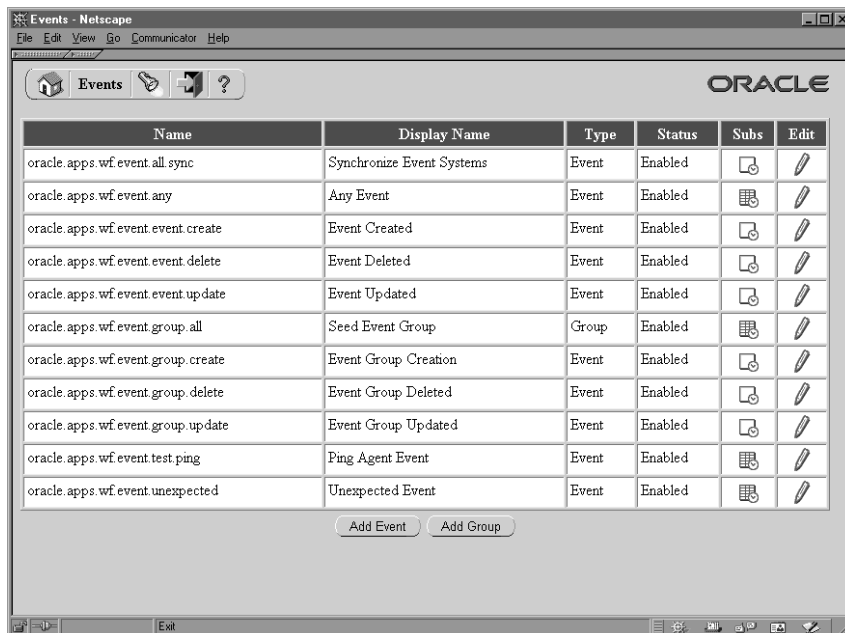
```
<webagent>/wf_event_html.listevents
```

Replace *<webagent>* with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



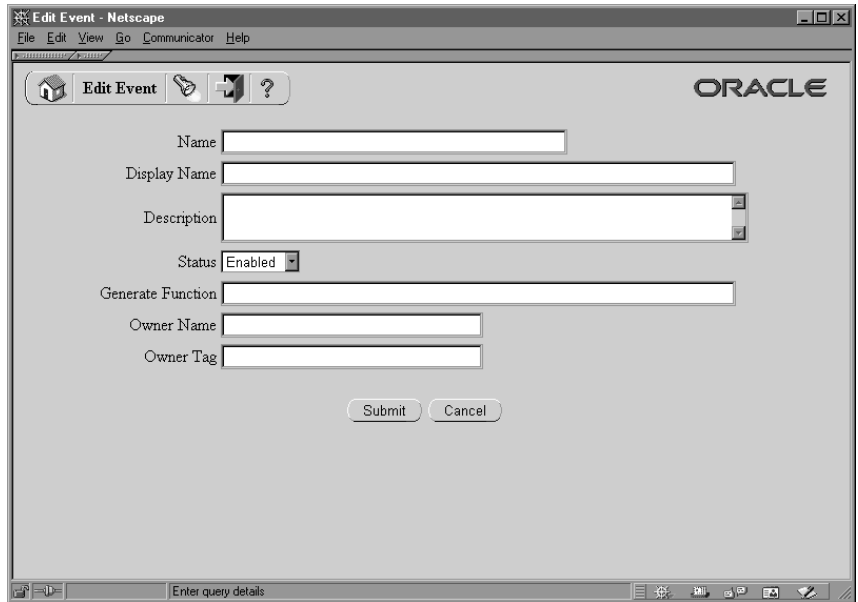
**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

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



2. The Events page appears, displaying a list of existing events. The Events page summarizes the internal name, display name, type, and status of each event.

Choose the Add Event button to open the Edit Event page.



3. Enter the internal name of the event in the Name field. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying an event. The internal name is case-sensitive. The suggested format is a compound structure of identifiers separated by periods (.) as follows:
 

*<company>. <family>. <product>. <component>. <object>. <event>*
4. Enter a Display Name for the event. This name appears in the Events list.
5. Enter an optional description for the event.
6. In the Status field, select Enabled or Disabled as the event status. If you disable an event, it still remains in the Events list for reference, but you cannot use the event in active subscriptions.
7. If you are defining an event that occurs on your local system, enter the Generate Function for the event. The Generate function is a PL/SQL procedure that can produce the complete event data from the event name, event key, and an optional parameter list. See: Standard API for an Event Data Generate Function: page 7 – 21.
8. You can optionally identify the program or application that owns the event by entering the program name in the Owner Name field and the program ID in the Owner Tag field. The Owner Name and Owner Tag are not required if you are defining an event manually in the Edit Event page. However, if you use a program to create

event definitions automatically, the Event Manager displays the owner information set by that program in these fields. You can use the Edit Event page to update this information manually if necessary.

9. Choose the Submit button to save the event and return to the Events page. The Events page displays an updated list of events.

You can also choose the Cancel button to return to the Events page without saving the event.

### ► To Define an Event Group

1. Use a web browser to connect to the following URL:

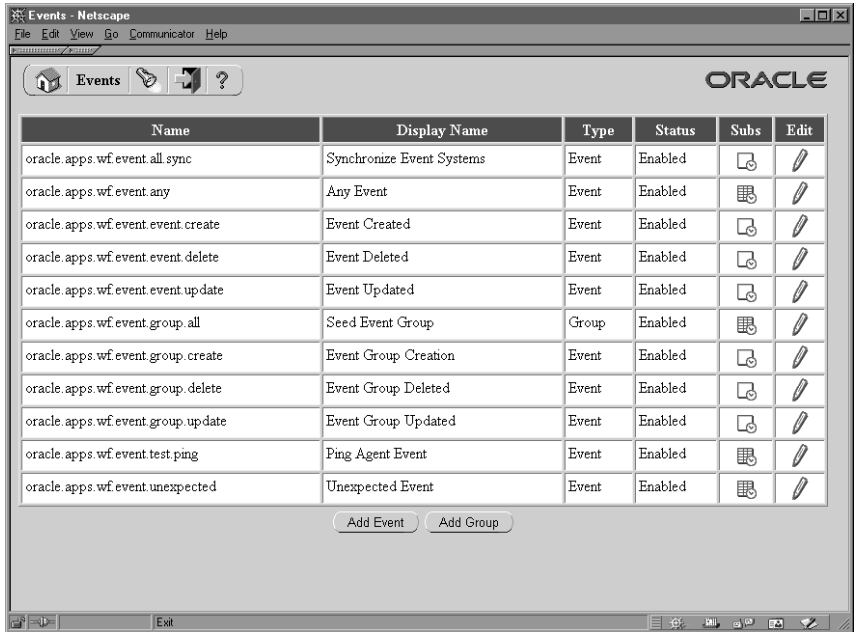
```
<webagent>/wf_event_html.listevents
```

Replace *<webagent>* with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.

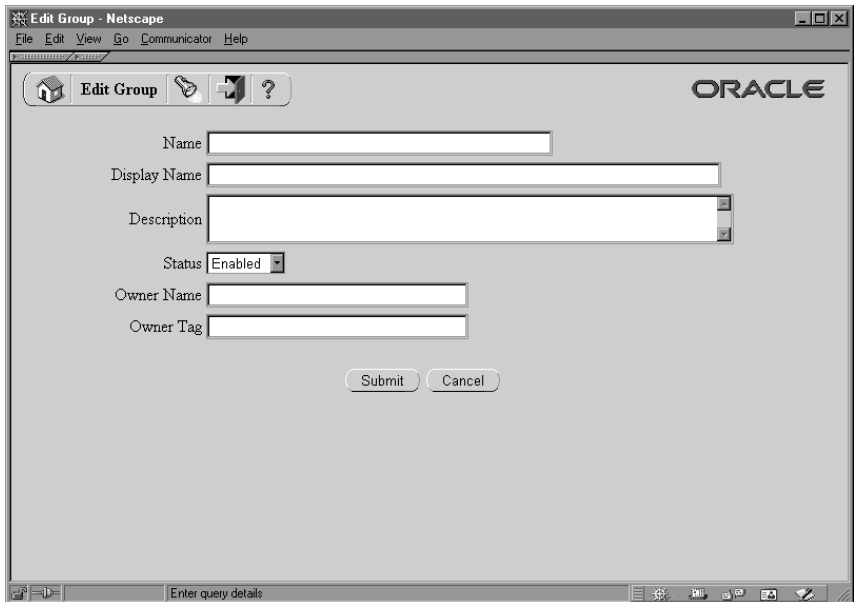


**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

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



2. The Events page appears, displaying a list of existing events. Choose the Add Group button to open the Edit Group page.



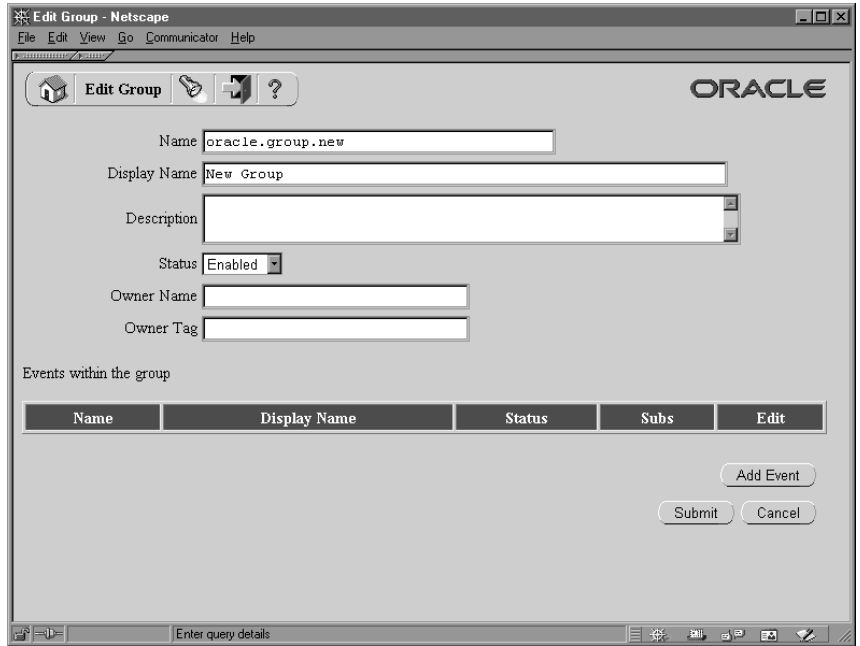
3. Enter the internal name of the event group in the Name field. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying an event group. The internal name is case-sensitive and should have a compound structure of identifiers separated by periods (.) in the following format:

*<company>.<family>.<product>.<component>.<object>.<event>*

4. Enter a Display Name for the event group. This name appears in the Events list.
5. Enter an optional description for the event group.
6. In the Status field, select Enabled or Disabled as the event group status. If you disable an event group, it still remains in the Events list for reference, but you cannot use the event group in active subscriptions.
7. You can optionally identify the program or application that owns the event group by entering the program name in the Owner Name field and the program ID in the Owner Tag field. The Owner Name and Owner Tag are not required if you are defining an event group manually in the Edit Group page. However, if you use a program to create event group definitions automatically, the Event Manager displays the owner information set by that program in these fields. You can use the Edit Group page to update this information manually if necessary.
8. Choose the Submit button to save the event group.

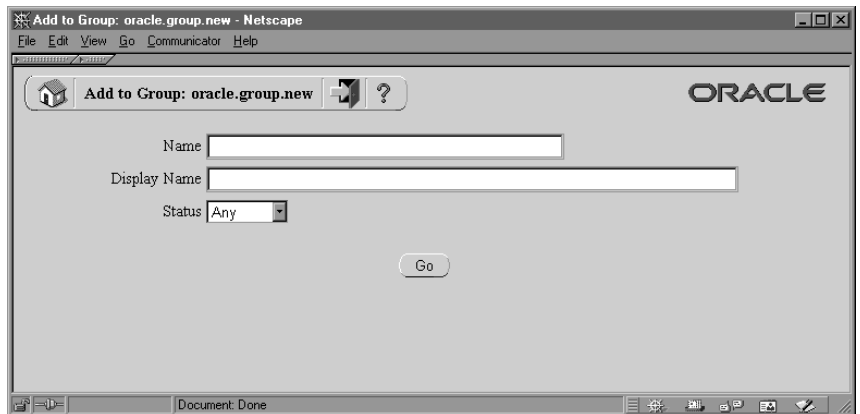
**Note:** You can also choose the Cancel button to return to the Events page without saving the event group.

After you save the event group definition, the Edit Group page displays the list of member events for that group, including the name, display name, and status of each event.



9. To add a member event to the group, choose the Add Event button.

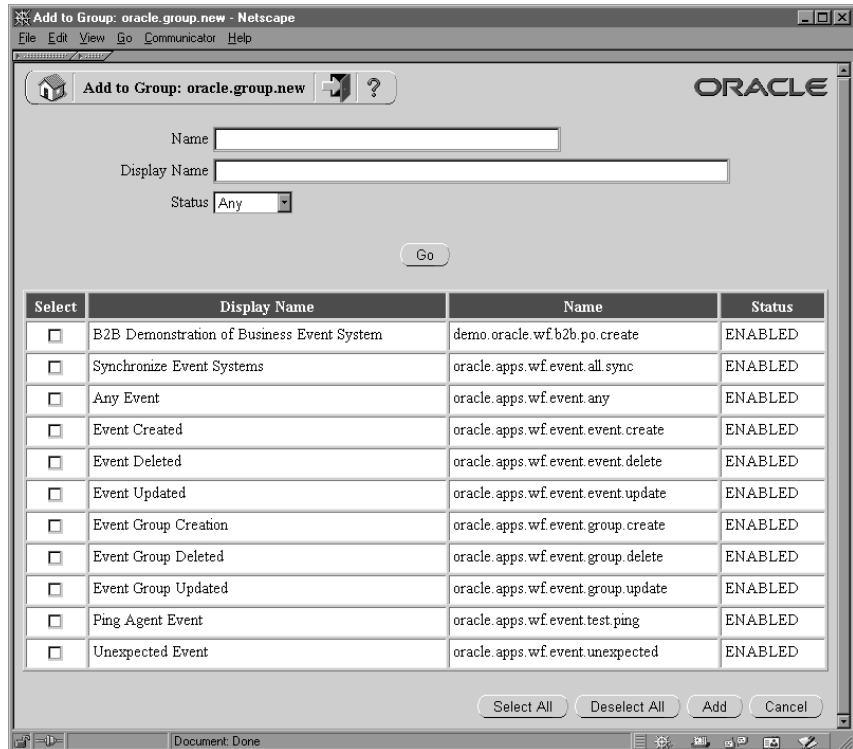
**Note:** An event group can contain only individual events as its members. It cannot contain another group.



10. In the Add to Group page that appears, enter search criteria to locate the event you want to add. The search criteria are:

- Name—enter the internal name of the event you want to add.

- Display Name—enter the display name of the event you want to add.
  - Status—choose Enabled or Disabled as the status of the event you want to add. Choose Any to search for events of any status.
11. Choose the Go button. The Add to Group page displays a list of events that match your search criteria.



12. Select the event or events that you want to add to your event group. You can choose the Select All button to select all the events in the list, or choose the Deselect All button to deselect all the events in the list.

If you want to search for different events, enter new search criteria and choose the Go button. The Add to Group page displays the list of events that match your new search criteria.

You can also choose the Cancel button to cancel your current search and return to your previous search results.

13. When you have finished selecting the events you want to add, choose the Add button to add the selected events to your event



group. The Edit Group page appears, displaying the updated list of event group members.

**Edit Group - Netscape**

File Edit View Go Communicator Help

**Edit Group** ORACLE

Name: oracle.group.new

Display Name: New Group

Description: [Text Area]

Status: Enabled

Owner Name: [Text Field]

Owner Tag: [Text Field]

Events within the group

Select	Name	Display Name	Status	Subs	Edit
<input type="checkbox"/>	oracle.apps.wf.event.event.delete	Event Deleted	ENABLED		
<input type="checkbox"/>	oracle.apps.wf.event.event.update	Event Updated	ENABLED		
<input type="checkbox"/>	oracle.apps.wf.event.event.create	Event Created	ENABLED		

Select All Deselect All Delete Add Event

Submit Cancel

Document Done

14. Choose the Submit button to save the event group definition.

**Note:** You can also choose the Cancel button to return to the Add to Group page with your latest search results.

15. If you want to remove a member event from the group, select the event or events you want to delete in the Edit Group page. Choose Select All to select all the events in the list, or choose Deselect All to deselect all the events in the list.

16. Choose the Delete button to remove the selected events from your event group. The Edit Group page displays the updated list of event group members.

**Note:** Removing an individual member event from an event group does not delete the event definition for the individual event. The individual event remains in the Events list.

17. To view the subscriptions that reference an event, choose the schedule icon in the Subs column for that event. The Event

Subscriptions page appears, displaying the list of subscriptions to the event.

**Note:** For events that do not have any subscriptions yet, a blank schedule icon appears. For events that do have subscriptions referencing them, a full schedule icon appears.

You can begin defining a new subscription on the event by choosing the Add Subscription button in the Event Subscriptions page. The Edit Subscription page appears with the event name automatically entered in the Event Filter field. See Defining Event Subscriptions: page 13 – 34.

18. To update an event, choose the pencil icon in the Edit column for that event. The Edit Event page appears. Make your changes to the event definition and save your work. See: To Define an Event: page 13 – 5.

### ► To Find Events

1. Use a web browser to connect to the following URL:

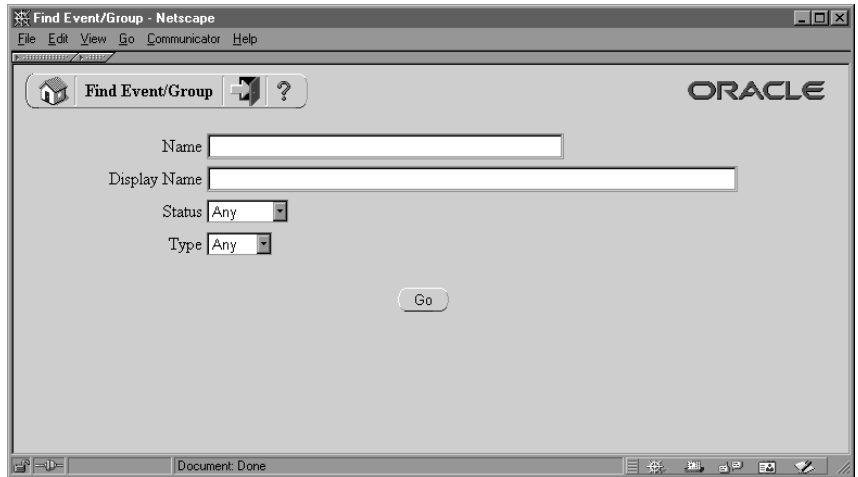
```
<webagent>/wf_event_html.findevent
```

Replace *<webagent>* with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

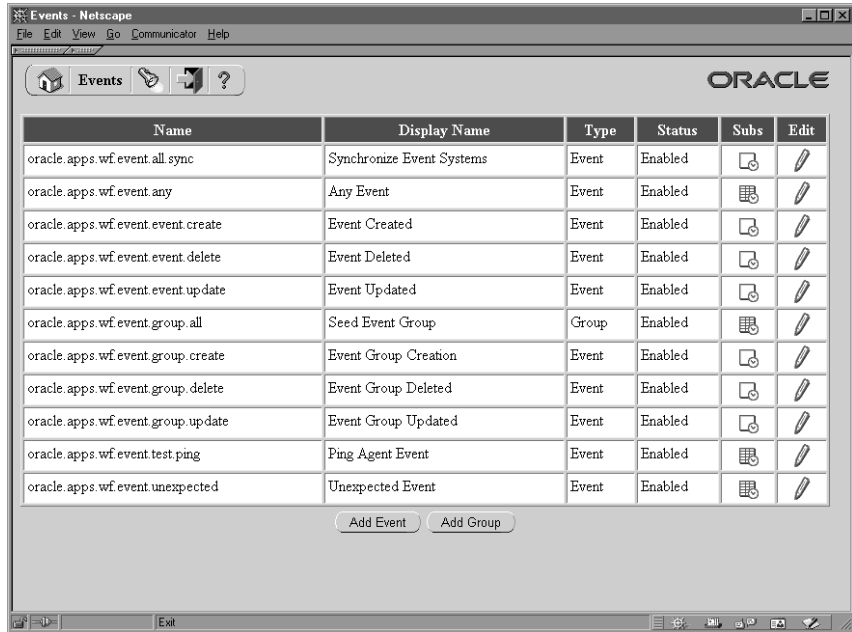
**Note:** You can also access the Find Event/Group web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 9 – 2.



2. The Find Event/Group page appears. The Find Event/Group page lets you enter search criteria to locate specific events. The search criteria are:
  - Name—enter the internal name of the event you want to display.
  - Display Name—enter the display name of the event you want to display.
  - Status—choose Enabled or Disabled as the status of the events you want to display, or choose Any to display events of any status.
  - Type—choose Event or Group as the type of the events you want to display, or choose Any to display events of any type.
3. Choose the Go button. The Events page appears, displaying a list of events that match your search criteria.

► **To Update or Delete an Event**

1. Locate the event you want in the Events page. You can use the Find Event/Group page to find the event that you want and display the event in the Events page. See: To Find Events: page 13 – 14.



- To view the subscriptions that reference an event, choose the schedule icon in the Subs column for that event. The Event Subscriptions page appears, displaying the list of subscriptions to the event.

**Note:** For events that do not have any subscriptions yet, a blank schedule icon appears. For events that do have subscriptions referencing them, a full schedule icon appears.

You can begin defining a new subscription on the event by choosing the Add Subscription button in the Event Subscriptions page. The Edit Subscription page appears with the event name automatically entered in the Event Filter field. See Defining Event Subscriptions: page 13 – 34.

- To update an event, choose the pencil icon in the Edit column for that event. The Edit Event page appears. Make your changes to the event definition and save your work. See: To Define an Event: page 13 – 5.
- To delete an event, choose the trash icon in the Delete column for that event, and choose OK in the confirmation window that appears. You can also choose Cancel in the confirmation window to return to the Events page without deleting the event.

**Note:** You can only delete events that do not have any subscriptions referencing them and that do not belong to any event groups.

---

## Systems

A system is a logically isolated software environment such as a host machine or database instance. You should define each system to or from which you will communicate events in the Event Manager.

When you define a system, you can specify whether it is associated with a master system from which you want it to receive Event Manager object definition updates.

Each system can expose one or more addressable points of communication, called agents. After you define your systems, you should define the agents within those systems that you will use to communicate business events. See: Agents: page 13 – 22.

### Local System

---

When you install Oracle Workflow in a database, that database is automatically defined as a system in the Event Manager and set as the local system in the Global Workflow Preferences page. The following table lists the default properties of the local system definition.

System Property	Value
Name	<database global name>
Display Name	<database global name>
Description	Local System Created by Oracle Workflow Configuration Assistant
Master	(blank)

Table 13 – 1 (Page 1 of 1)

You can update the local system definition if necessary.

Oracle Workflow sets the status of the local system to Enabled by default. After you finish setting up the Business Event System, you can use the Global Workflow Preferences page to set the system status that you want for event processing. See: Setting Global User Preferences: page 2 – 14.

## ► To Define a System

1. Use a web browser to connect to the following URL:

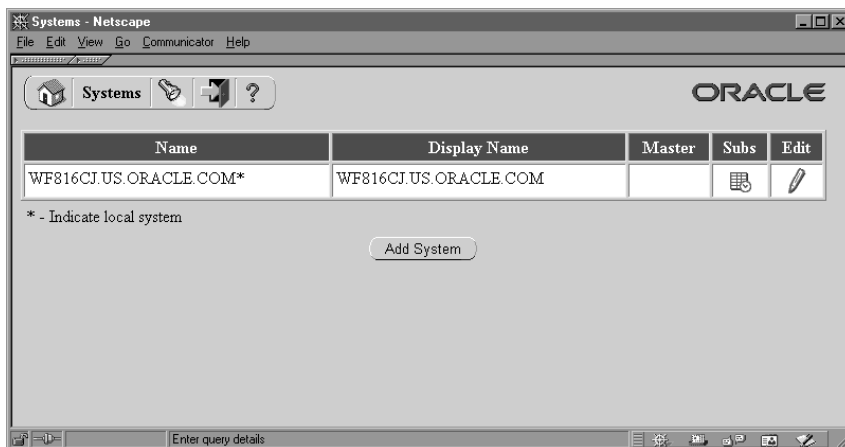
`<webagent>/wf_event_html.listsystems`

Replace `<webagent>` with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



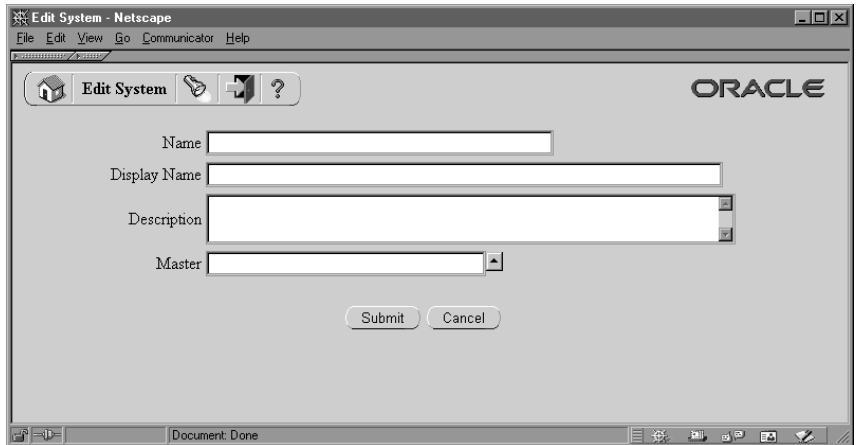
**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

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



2. The Systems page appears, displaying a list of existing systems. The Systems page summarizes the internal name, display name, and master system of each system. An asterisk marks the local system.

Choose the Add System button to open the Edit System page.



3. Enter the internal name of the system in the Name field. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying a system.



**Attention:** The internal name must be all-uppercase and should not include any single or double quotation marks ( ' or " ) or spaces.

4. Enter a Display Name for the system. This name appears in the Systems list.
5. Enter an optional description for the system.
6. Optionally enter a Master system from which you want this system to receive Event Manager object definition updates. Click on the Master field's up arrow icon to display a list of systems from which to choose. See: Using a List of Values: page 13 – 22.
7. Choose the Submit button to save the system and return to the Systems page. The Systems page displays an updated list of systems.

You can also choose the Cancel button to return to the Systems page without saving the system.

### ► To Find Systems

1. Use a web browser to connect to the following URL:

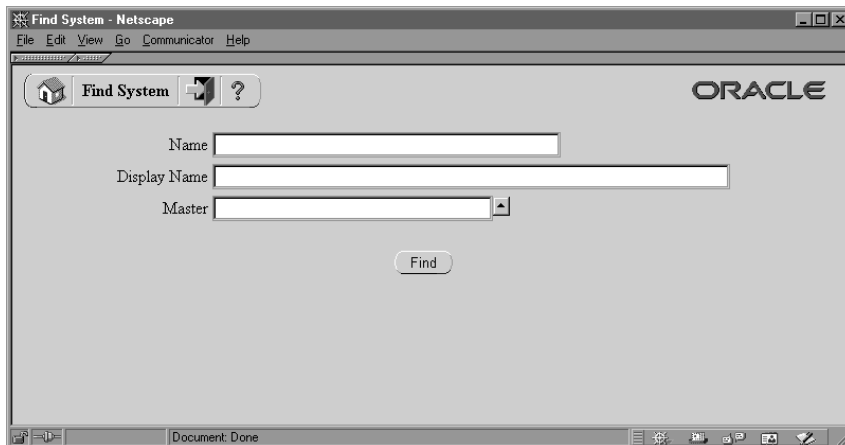
```
<webagent>/wf_event_html.findsystem
```

Replace *<webagent>* with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

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



2. The Find System page appears. The Find System page lets you enter search criteria to locate specific systems. The search criteria are:
  - Name—enter the internal name of the system you want to display.
  - Display Name—enter the display name of the system you want to display.
  - Master—enter the master system for the system you want to display. Click on the field's up arrow icon to display a list of systems from which to choose. See: Using a List of Values: page 13 – 22.
3. Choose the Find button. The Systems page appears, displaying a list of systems that match your search criteria. An asterisk marks the local system.



► **To Update or Delete a System**

1. Locate the system you want in the Systems page. You can use the Find System page to find the system that you want and display the system in the Systems page. See: To Find Systems: page 13 – 19.



2. To view the subscriptions for a system, choose the schedule icon in the Subs column for that system. The Event Subscriptions page appears, displaying the list of subscriptions by the system.

**Note:** For systems that do not have any subscriptions yet, a blank schedule icon appears. For systems that do have subscriptions, a full schedule icon appears.

You can begin defining a new subscription for the system by choosing the Add Subscription button in the Event Subscriptions page. The Edit Subscription page appears with the system name automatically entered in the System field. See Defining Event Subscriptions: page 13 – 34.

3. To update a system, choose the pencil icon in the Edit column for that system. The Edit System page appears. Make your changes to the system definition and save your work. See: To Define a System: page 13 – 18.
4. To delete a system, choose the trash icon in the Delete column for that system, and choose OK in the confirmation window that appears. You can also choose Cancel in the confirmation window to return to the Systems page without deleting the system.

**Note:** You can only delete systems that do not have any agents defined on them or any subscriptions referencing them.

## ► Using a List of Values

1. For a field that supports a list of values, click on the field's up-arrow icon to display a list of values window.
2. In the Find field, enter search criteria and choose the Find button to retrieve a subset of values that match your criteria. You can also choose the Clear button to clear the Find field. If you do not specify any search criteria and simply choose Find, you retrieve the complete list of values.
3. Click on a value from the list to select that value and close the list of values window. The value you select populates the original field.

---

## Agents

An agent is a named point of communication within a system. Communication within and between systems is accomplished by sending a message from one agent to another. A single system can have several different agents representing different communication alternatives. For example, a system may have different agents to support inbound and outbound communication, communication by different protocols, different propagation frequencies, or other alternatives.

You should define each agent that you will use to communicate events in the Event Manager. Each agent's name must be unique within its system. The agent can be referenced in code within Oracle Workflow by a compound name in the following format:

```
<agent_name>@<system_name>
```

For example, the agent WF\_IN within the system HUB could be referenced as WF\_IN@HUB.

After defining the agents on your local system, you should set them up for event message propagation by scheduling listeners for local inbound agents and propagations for local outbound agents. See: *Scheduling Listeners for Local Inbound Agents: page 13 – 56* and *Scheduling Propagations for Local Outbound Agents: page 13 – 56*.

### **Assigning a Direction to an Agent**

When you define an agent in the Event Manager, you must specify the direction of communication that the agent supports on its local system.

- In—Inbound communication to the system. The agent receives messages in a specific protocol and presents them to the system in a standard format.
- Out—Outbound communication from the system. The agent accepts messages from the system in a standard format and sends them using the specified protocol.

### **Assigning a Protocol to an Agent**

---

You must associate each agent with the protocol by which it communicates messages. The protocol specifies how the messages are encoded and transmitted. For a message to be successfully communicated, the sending and receiving agents must use the same protocol.

A protocol can represent a network standard, such as SQLNET. It can also represent a business-to-business standard that defines the higher-level message format and handshaking agreements between systems in addition to the network standard.

The Business Event System interacts with an agent through an AQ queue. You can use AQ to perform the propagation of messages by the SQLNET protocol which it supports. In Oracle9i, AQ also includes Internet access functionality that lets you perform AQ operations over the Internet by using AQ's Internet Data Access Presentation (IDAP) for messages and transmitting the messages over the Internet using transport protocols such as HTTP or HTTPS. Additionally, the Messaging Gateway feature of AQ in Oracle9i enables communication between applications based on non-Oracle messaging systems and AQ, letting you integrate with third party messaging solutions. You can also implement other services to propagate messages by different protocols. The following table shows which services you can use for various protocols, depending on your database version.

<b>Database Version</b>	<b>SQLNET Protocol</b>	<b>HTTP/HTTPS Protocols</b>	<b>Integration with Third Party Messaging Solutions</b>
Oracle8i	Oracle Advanced Queuing	Oracle Message Broker	Oracle Message Broker
Oracle9i	Oracle Advanced Queuing	Oracle Advanced Queuing	Oracle Advanced Queuing Messaging Gateway feature

**Table 13 – 2 (Page 1 of 1)**

To implement a custom protocol, you must perform the following steps:

1. Define AQ queues to hold pending inbound and outbound messages.
2. Provide code that propagates messages to and from the AQ queues.
3. Define a lookup code for the new protocol in the Event Protocol Type (WF\_AQ\_PROTOCOLS) lookup type, which is stored in the Standard item type. The Event Manager uses the Event Protocol Type lookup type to validate the protocol for an agent. See: To Create Lookup Codes for a Lookup Type: page 4 – 21.
4. Define agents with the new protocol. See: To Define an Agent: page 13 – 29.

If an agent supports inbound communication, you must specify the address by which systems can communicate with it. The format of the address depends on the agent's protocol. For agents that use the SQLNET protocol, the address must be in the following format to enable AQ propagation:

```
<schema>.<queue>@<database link>
```

In this format, *<schema>* represents the schema that owns the queue, *<queue>* represents the queue name, and *<database link>* represents the name of the database link to the instance where the queue is located.

**Note:** You must enter the database link name exactly as the name was specified when the database link was created. For example, if a database link is named ORA816.US.ORACLE.COM, you must enter that complete name in the address of an agent on that database. You cannot abbreviate the name to ORA816.

The names of the database links that you want to use for the Business Event System should be fully qualified with the domain names. To confirm the names of your database links, use the following syntax:

```
SELECT db_link FROM all_db_links
```

See: Creating Database Links: page 2 – 96.

### **Assigning a Queue to an Agent**

---

You must associate each agent with an AQ queue. The local system uses this queue to interact with the agent. To send messages, the system enqueues the messages on the queue and sets the recipient

addresses. To receive messages, the system runs a queue listener on the queue.

Event messages within the Oracle Workflow Business Event System are encoded in a standard format defined by the datatype `WF_EVENT_T`. You must assign each agent a PL/SQL package called a queue handler that translates between this standard Workflow format and the format required by the agent's queue. See: *Event Message Structure*: page 8 – 242.

**Note:** Even if the agent's queue uses `WF_EVENT_T` as its payload type, a queue handler is still required in order to set native AQ message properties.

Oracle Workflow provides two standard queue handlers, called `WF_EVENT_QH` and `WF_ERROR_QH`, for queues that use `SQLNET` propagation and use the `WF_EVENT_T` datatype as their payload type. You can use `WF_EVENT_QH` with queues that handle normal Business Event System processing, while `WF_ERROR_QH` should be used exclusively with error queues.

Oracle Workflow also provides a queue handler called `WF_EVENT_OMB_QH`, which you can use if you implement Oracle Message Broker with Oracle8i to propagate messages between systems by another protocol such as HTTP. See: *Setting Up the WF\_EVENT\_OMB\_QH Queue Handler*: page 2 – 100 and *Mapping Between WF\_EVENT\_T and OMBAQ\_TEXT\_MSG*: page 8 – 257.

If you want to use queues that require a different format, create a custom queue handler for that format. Your custom queue handler must include a set of standard APIs to enqueue and dequeue messages in the custom format. See: *Standard APIs for a Queue Handler*: page 7 – 23.

## Standard Agents

---

When you install Oracle Workflow, four standard agents are automatically defined for the Business Event System.

- `WF_IN`—Standard inbound agent
- `WF_OUT`—Standard outbound agent
- `WF_DEFERRED`—Standard agent for deferred subscription processing
- `WF_ERROR`—Standard agent for error handling

These agents use standard queues that are automatically defined when you install Oracle Workflow. See: *Setting Up Queues*: page 2 – 97.

You can enable or disable the WF\_IN and WF\_OUT agents, but you must not make any other changes to their definitions. You must not make any changes to the definitions of the WF\_DEFERRED and WF\_ERROR agents.

However, you must schedule listeners for the WF\_DEFERRED and WF\_ERROR agents to enable deferred subscription processing and error handling for the Business Event System, respectively. Also, if you want to use the WF\_IN and WF\_OUT agents for event message propagation, schedule a listener for WF\_IN and propagations for WF\_OUT as well. See: Scheduling Listeners for Local Inbound Agents: page 13 – 56 and Scheduling Propagations for Local Outbound Agents: page 13 – 56.

Additionally, a standard agent named WF\_SMTP\_O\_1\_QUEUE is defined for the Notification Mailer SMTP queue. This agent appears in the Check Setup page and the Event System Local Queues page, enabling you to use these pages to check the number of notification messages on the Notification Mailer queue. The WF\_SMTP\_O\_1\_QUEUE agent is not used by the Business Event System, however, so its status is Disabled and no queue handler is defined for it. You must not run an agent listener for this agent. See: Implementing the Notification Mailer: page 2 – 48, Checking the Business Event System Setup: page 13 – 53, and Reviewing Local Queues: page 13 – 72.

The following table lists the default properties for the standard WF\_IN agent.

Agent Property	Value
Name	WF_IN
Display Name	WF_IN
Description	WF_IN
Protocol	SQLNET
Address	<workflow schema>.WF_IN@<local database>
System	<local system>
Queue Handler	WF_EVENT_QH
Queue Name	<workflow schema>.WF_IN

**Table 13 – 3 (Page 1 of 2)**

Agent Property	Value
Direction	In
Status	Enabled

**Table 13 – 3 (Page 2 of 2)**

The following table lists the default properties for the standard WF\_OUT agent.

Agent Property	Value
Name	WF_OUT
Display Name	WF_OUT
Description	WF_OUT
Protocol	SQLNET
Address	<workflow schema>.WF_OUT@<local database>
System	<local system>
Queue Handler	WF_EVENT_QH
Queue Name	<workflow schema>.WF_OUT
Direction	Out
Status	Enabled

**Table 13 – 4 (Page 1 of 1)**

The following table lists the default properties for the standard WF\_DEFERRED agent.

Agent Property	Value
Name	WF_DEFERRED
Display Name	WF_DEFERRED
Description	WF_DEFERRED
Protocol	SQLNET
Address	<workflow schema>.WF_DEFERRED@<local database>

**Table 13 – 5 (Page 1 of 2)**

Agent Property	Value
System	<local system>
Queue Handler	WF_EVENT_QH
Queue Name	<workflow schema>.WF_DEFERRED
Direction	In
Status	Enabled

**Table 13 – 5 (Page 2 of 2)**

The following table lists the default properties for the standard WF\_ERROR agent.

Agent Property	Value
Name	WF_ERROR
Display Name	WF_ERROR
Description	WF_ERROR
Protocol	SQLNET
Address	<workflow schema>.WF_ERROR@<local database>
System	<local system>
Queue Handler	WF_ERROR_QH
Queue Name	<workflow schema>.WF_ERROR
Direction	In
Status	Enabled

**Table 13 – 6 (Page 1 of 1)**

The following table lists the default properties for the standard WF\_SMTP\_O\_1\_QUEUE agent.

Agent Property	Value
Name	WF_SMTP_O_1_QUEUE
Display Name	WF_SMTP_O_1_QUEUE

**Table 13 – 7 (Page 1 of 2)**



Agent Property	Value
Description	This is the Mailer Queue – do not submit the Agent Listener Concurrent Program on this Agent.
Protocol	SQLNET
Address	<workflow schema>.WF_SMTP_O_1_QUEUE @<local database>
System	<local system>
Queue Handler	
Queue Name	<workflow schema>.WF_SMTP_O_1_QUEUE
Direction	In
Status	Disabled

Table 13 – 7 (Page 2 of 2)

### ► To Define an Agent

1. Use a web browser to connect to the following URL:

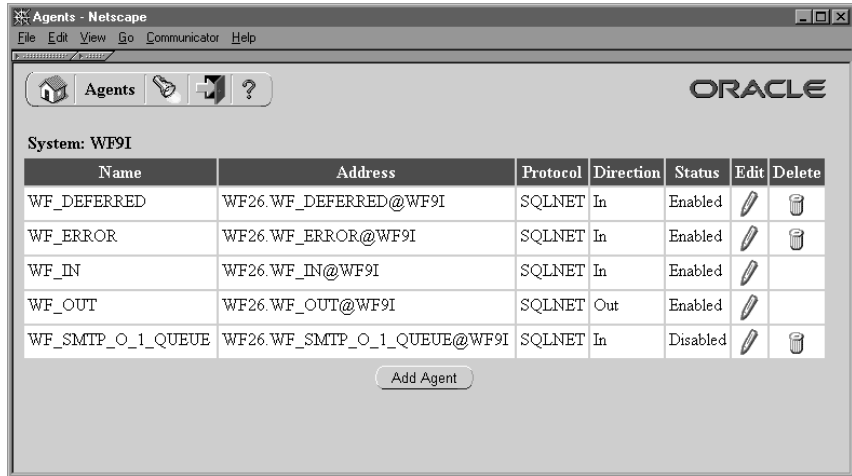
`<webagent>/wf_event_html.listagents`

Replace `<webagent>` with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



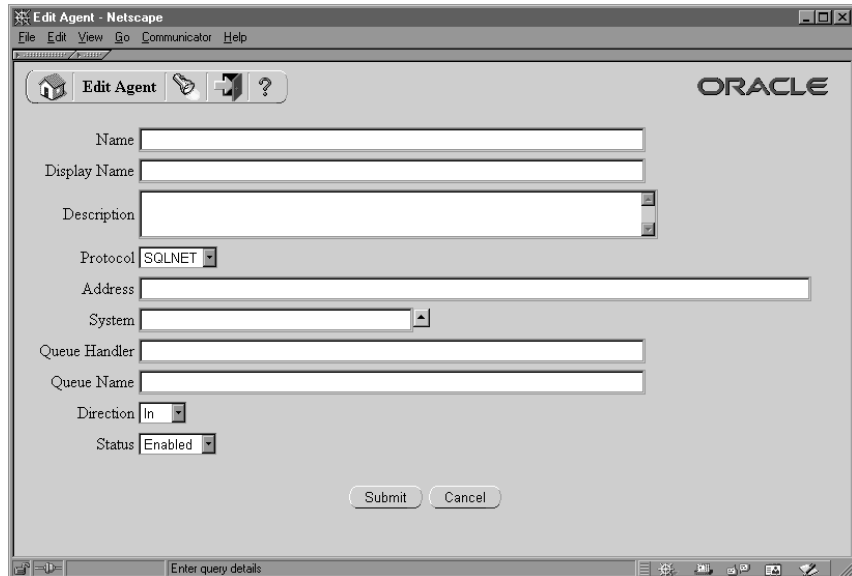
**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

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



- The Agents page appears, displaying a list of existing agents grouped by the system where they are located. The Agents page summarizes the internal name, address, protocol, direction, and status of each agent.

Choose the Add Agent button to open the Edit Agent page.



- Enter the internal name of the agent in the Name field. The agent's internal name must be unique within the agent's system. Oracle

Workflow APIs, SQL scripts, and PL/SQL procedures may refer to the internal name when identifying an agent.



**Attention:** The internal name must be all-uppercase and should not include any single or double quotation marks ( ' or " ) or spaces.

4. Enter a Display Name for the agent.
5. Enter an optional description for the agent.
6. Select the message communication protocol that the agent supports.
7. If the agent supports inbound communication to its system, enter the address for the agent. The format of the address depends on the protocol you select.

For agents that use the SQLNET protocol, the address must be in the following format to enable AQ propagation:

```
<schema>.<queue>@<database link>
```

*<schema>* represents the schema that owns the queue, *<queue>* represents the queue name, and *<database link>* represents the database link to the instance where the queue is located.

**Note:** You must enter the database link name exactly as the name was specified when the database link was created. See: [Creating Database Links: page 2 – 96](#).

8. Enter the system in which the agent is defined. Click on the System field's up arrow icon to display a list of systems from which to choose. See: [Using a List of Values: page 13 – 22](#).
9. Enter the Queue Handler for the agent. The queue handler is the PL/SQL package that translates between the Workflow event message format (WF\_EVENT\_T) and the message format required by the queue associated with the agent. See: [Standard APIs for a Queue Handler: page 7 – 23](#).



**Attention:** You must enter the queue handler name in all uppercase.

10. Enter the name of the queue that the local system uses to interact with the agent. Since only the local system refers to this queue name, the queue name should be within the scope of this system, without requiring a database link. Use the following format to specify the queue name:

```
<schema>.<queue>
```

`<schema>` represents the schema that owns the queue, and `<queue>` represents the queue name.



**Attention:** You must enter the queue name in all uppercase.

11. In the Direction field, select In for an agent that supports inbound communication to its system, or select Out for an agent that supports outbound communication from its system.
12. In the Status field, select Enabled or Disabled as the agent status. If you disable an agent, it still remains in the Agents list for reference, but you cannot use the agent in active subscriptions.
13. Choose the Submit button to save the agent and return to the Agents page. The Agents page displays an updated list of agents.  
  
You can also choose the Cancel button to return to the Agents page without saving the agent.

### ► To Find Agents

1. Use a web browser to connect to the following URL:

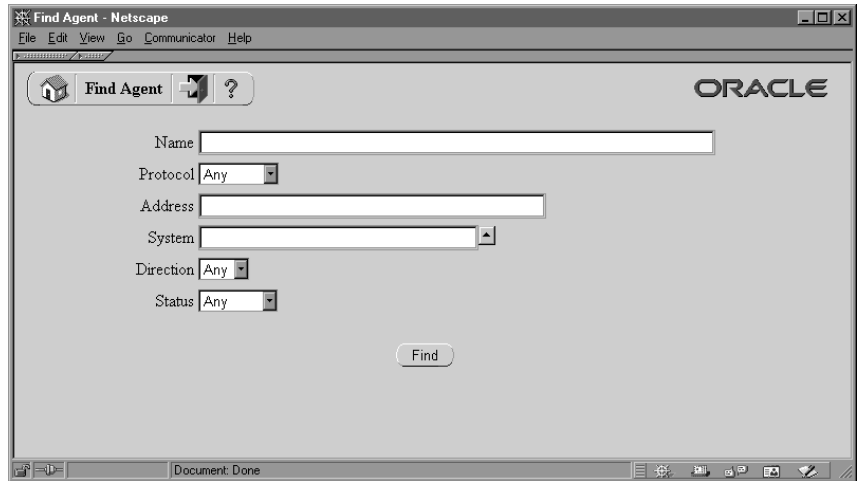
```
<webagent>/wf_event_html.findagent
```

Replace `<webagent>` with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

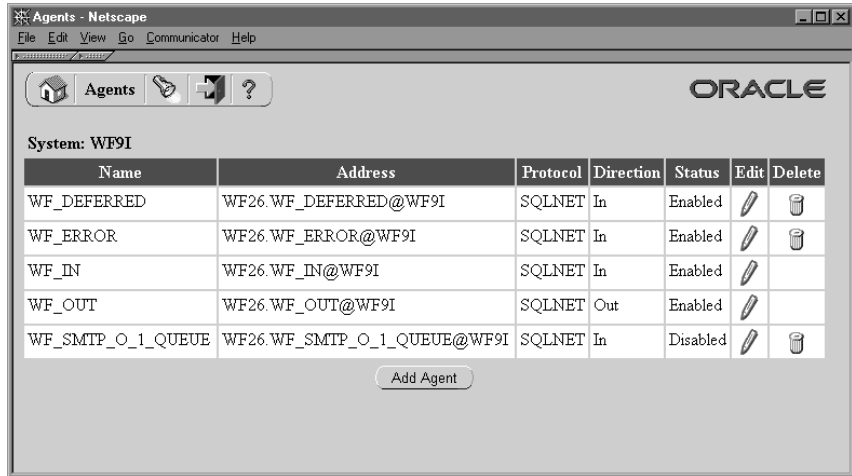
**Note:** You can also access the Find Agent web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 9 – 2.



2. The Find Agent page appears. The Find Agent page lets you enter search criteria to locate specific agents. The search criteria are:
  - Name—enter the internal name of the agent you want to display.
  - Protocol—choose the protocol of the agent you want to display, or choose Any to display agents of any protocol.
  - Address—enter the address of the agent you want to display.
  - System—enter the system of the agent you want to display. Click on the field's up arrow icon to display a list of systems from which to choose. See: Using a List of Values: page 13 – 22.
  - Direction—choose In or Out as the direction of the agents you want to display, or choose Any to display agents of any direction.
  - Status—choose Enabled or Disabled as the status of the agents you want to display, or choose Any to display agents of any status.
3. Choose the Find button. The Agents page appears, displaying a list of agents that match your search criteria.

► **To Update or Delete an Agent**

1. Locate the agent that you want in the Agents page. You can use the Find Agent page to find the agent that you want and display the agent in the Agents page. See: To Find Agents: page 13 – 32.



2. To update an agent, choose the pencil icon in the Edit column for that agent. The Edit Agent page appears. Make your changes to the agent definition and save your work. See: To Define an Agent: page 13 – 29.
3. To delete an agent, choose the trash icon in the Delete column for that agent, and choose OK in the confirmation window that appears. You can also choose Cancel in the confirmation window to return to the Agents page without deleting the agent.

**Note:** You can only delete agents that do not have any subscriptions referencing them.

**Note:** Whenever you make changes to your agents that affect the physical implementation required for message propagation, you should recheck your propagation setup. See: Setting Up Message Propagation: page 13 – 53.

---

## Event Subscriptions

An event subscription is a registration indicating that a particular event is significant to a particular system and specifying the processing to perform when the triggering event occurs. You can define your event subscriptions in the Event Manager.

When you install Oracle Workflow, several default subscriptions to predefined Workflow events are automatically created. You can update, enable, or disable these subscriptions to perform the event processing that you want. See: Predefined Workflow Events: page 14 – 2.

Whenever an event is raised locally or received from an external source, the Event Manager searches for and executes any active subscriptions by the local system to that event or to the Any event. If no active subscriptions exist for the event that occurred (apart from subscriptions to the Any event), then Oracle Workflow executes any active subscriptions to the Unexpected event. See: Any Event: page 14 – 10 and Unexpected Event: page 14 – 12.

## **Defining the Subscriber**

---

To begin defining a subscription, you specify which system is the subscriber. The subscriber is the system where you want the subscription to execute.

Each subscription defines an action on exactly one system, so you should define a separate subscription for each system involved in the processing you want to perform. For example, if you want to propagate data from one system to another, you should define one subscription for the sending system, and another subscription for the receiving system.

## **Defining How a Subscription is Triggered**

---

You must specify the source type of the events to which the subscription applies. Events can have the following source types:

- Local—The subscription applies only to events raised on the subscribing system.
- External—The subscription applies only to events received by an inbound agent on the subscribing system.

**Note:** All event messages received by an inbound agent on the subscribing system are considered to have an External source, whether the sending agent is located on a remote system or on the local system.

- Error—The subscription applies to only errored events dequeued from the WF\_ERROR queue.

Next, select the event that you want to trigger the subscription. You can choose either an individual event or an event group. If you choose an event group, the subscription will be triggered whenever any one of the group's member events occurs.

You can also optionally restrict the subscription to be triggered only by events received from a specific source agent. However, in most cases you do not need to specify a source agent.

## **Controlling How a Subscription is Executed**

---

If you define multiple subscriptions to the same event, you can control the order in which the Event Manager executes those subscriptions by specifying a phase number for each subscription. Subscriptions are executed in ascending phase order. For example, you can enter 10 for the subscription that you want to execute first when an event occurs, 20 for the subscription that you want to execute second, and so on. You can use phases to ensure that different types of actions are performed in the appropriate order, such as executing subscriptions that perform validation before subscriptions that perform other types of processing.

**Note:** If you enter the same phase number for more than one subscription, the Event Manager may execute those subscriptions in any order, relative to each other. However, the Event Manager will still execute that group of subscriptions in their specified place in the phase order, relative to subscriptions with other phase numbers.

You can also use the phase number for a subscription to control whether the subscription is executed immediately or is deferred. The Event Manager treats subscriptions with a phase number of 100 or higher as deferred subscriptions. See: *Deferred Subscription Processing*; page 13 – 41.

Depending on the processing to be performed, a subscription may require the complete set of event information contained in the event data, or it may require only the event key that identifies the instance of the event. You can improve performance by specifying *Key* as the rule data for subscriptions that do not require the complete event data. For locally raised events, the Event Manager checks each subscription before executing it to determine whether the subscription requires the complete event data. If the event data is required but is not already provided, the Event Manager runs the *Generate* function for the event to produce the event data. However, if no subscriptions to the event require the event data, then the Event Manager will not run the *Generate* function, minimizing the resources required to execute the subscriptions.

**Note:** Even if there are subscriptions that require the complete event data, you can return control to the calling application more quickly after raising the event by deferring all those subscriptions. Then the Event Manager will not run the *Generate* function until those subscriptions are executed at a later time.



## Defining the Action for a Subscription

---

Subscription processing can include the following types of processing:

- Run a function on the event message.
- Send the event message to a workflow process.
- Send the event message to an agent.

### Running a Rule Function

To run a function on the event message, you must specify the rule function that you want to execute. You can also specify any additional parameters that you want to pass to the function.

Oracle Workflow provides a standard default rule function to perform basic subscription processing. This function is executed by default if no other rule function is specified for the subscription. The default rule function includes the following actions:

- Sending the event message to a workflow process, if specified in the subscription definition
- Sending the event message to an agent, if specified in the subscription definition

See: `Default_Rule`: page 8 – 281.

Oracle Workflow also provides some standard rule functions that you can use for testing and debugging or other purposes. See: `Event Subscription Rule APIs`: page 8 – 279.

You can extend your subscription processing by creating custom rule functions. Custom rule functions must be defined according to a standard API. See: `Standard API for an Event Subscription Rule Function`: page 7 – 25.

You can use a rule function for many different purposes, including:

- Performing validation
- Processing inbound messages as a Receive message handler for an application
- Making modifications to an outbound message, such as adding a correlation ID that associates this message with other messages

**Note:** You can call `WF_EVENT_FUNCTIONS_PKG.AddCorrelation()` within a custom rule function to add a correlation ID during your custom processing. See: `AddCorrelation`: page 8 – 294.

A rule function may read or write to the event message or perform any other database action. However, you should never commit within a rule function. The Event Manager never issues a commit as it is the responsibility of the calling application to commit. Additionally, the function must not change the connection context in any way, including security and NLS settings. If a rule function returns an error, subscription processing is halted.

If the subscription processing that you want to perform for an event includes several successive steps, you may find it advantageous to define multiple subscriptions to the event with simple rule functions that you can reuse, rather than creating complex specialized rule functions that cannot be reused. You can enter phase values for the subscriptions to specify the order in which they should be executed.

**Note:** If you enter a rule function other than the default function, you can still enter workflow and agent information for your function to reference, but Oracle Workflow does not automatically send the event message to the specified workflow and agent. Instead, you must either explicitly include the send processing in your rule function, or define a separate subscription that does use the default rule function to perform the send processing.

### **Sending the Event to a Workflow Process**

To send the event to a workflow process, you must specify the item type and process name of the process. The item key for the process is determined either by the correlation ID specified in the event message, or by the event key if no correlation ID is specified.

**Note:** You can call `WF_EVENT_FUNCTIONS_PKG.AddCorrelation()` during subscription processing to add a correlation ID to the event message. To use `AddCorrelation()`, you must enter a subscription parameter named ITEMKEY that specifies a function to generate the correlation ID. The function must be specified in the following format:

```
ITEMKEY=<package_name.function_name>
```

See: AddCorrelation: page 8 – 294.

By sending an event to a workflow process, you can model complex processing or routing logic beyond the options of directly running a predefined function or sending the event to a predefined recipient. For example, you can branch to different functions, initiate subprocesses, send notifications, or select recipient agents, based on the contents of the event message, or modify the event message itself.

Events are represented within workflow processes by event activities. See: Event Activity: page 4 – 54.

When the process receives the event, the Workflow Engine stores the event name, event key, and event message in item type attributes, as specified in the Receive event activity node's event details. The Workflow Engine also sets any parameters in the event message parameter list as item type attributes for the process, creating new item type attributes if a corresponding attribute does not already exist for any parameter. Also, the subscription's globally unique identifier (GUID) is set as a dynamic item attribute so that the workflow process can reference other information in the subscription definition.

If the event was originally raised by a Raise event activity in another workflow process, the item type and item key for that process are included in the parameter list within the event message. In this case, the Workflow Engine automatically sets the specified process as the parent for the process that receives the event, overriding any existing parent setting. See: SetItemParent: page 8 – 79.

If you want to specify additional parameters to set as item attributes for the workflow process, you can enter these parameters in the Parameters field of a subscription and use *WF\_RULE.SetParametersIntoParameterList()* in the subscription rule function to set the subscription parameters into the event message parameter list. The event parameters will then be set as item attributes for the workflow process when the process receives the event. See: SetParametersIntoParameterList: page 8 – 289.

**Note:** To send an event to a workflow process, you must either use the default rule function provided by Oracle Workflow or include send processing in your custom rule function. See: Standard API for an Event Subscription Rule Function: page 7 – 25.

### **Sending the Event to an Agent**

To send an event to an agent, you must specify either the Out Agent that you want to send the outbound message, or the To Agent that you want to receive the inbound message, or both.

- If you specify both a To Agent and an Out Agent, Oracle Workflow places the event message on the Out Agent's queue for propagation, addressed to the To Agent.
- If you specify a To Agent without an Out Agent, Oracle Workflow selects an outbound agent on the subscribing system whose queue type matches the queue type of the To Agent. The

event message is then placed on this outbound agent's queue for propagation, addressed to the To Agent.

- If you specify an Out Agent without a To Agent, Oracle Workflow places the event message on the Out Agent's queue without a specified recipient.
  - You can omit the To Agent if the Out Agent uses a multi-consumer queue with a subscriber list. (The standard Workflow queue handlers work only with multi-consumer queues.) In this case the queue's subscriber list determines which consumers can dequeue the message. If no subscriber list is defined for that queue, however, the event message is placed on the WF\_ERROR queue for error handling.

**Note:** The subscriber list for a multi-consumer queue in Oracle Advanced Queuing is different from event subscriptions in the Oracle Workflow Business Event System. For more information, see: *Subscription and Recipient Lists, Oracle Application Developer's Guide – Advanced Queuing*.

- You can also omit the To Agent if the Out Agent uses a single-consumer queue for which you have defined a custom queue handler. For a single-consumer queue, no specified consumer is required.

You can optionally specify the priority with which the recipient should dequeue a message. Messages are dequeued in ascending priority order.

**Note:** To send an event to an agent, you must either use the default rule function provided by Oracle Workflow or include send processing in your custom rule function. See: *Standard API for an Event Subscription Rule Function: page 7 – 25*.

If you want an event message to become available to the recipient at a future date, rather than being available immediately as soon as it is propagated, you can set the SEND\_DATE attribute within the event message to the date you want. You should set the send date during subscription processing before the event is sent, either in a prior subscription or earlier in the rule function before the send processing. The event message is propagated to the To Agent but does not become available for dequeuing until the specified date.

### Documenting Identifying Information for a Subscription

If you use a program to create subscription definitions automatically, the program can set its own name and brief identifier as the owner name and owner tag for the subscriptions. The program can then use

this identifying information to locate the subscriptions that it owns. You can use the Edit Subscription page to update the owner name and owner tag manually if necessary.

## **Deferred Subscription Processing**

---

If you do not want subscriptions for an event to be executed immediately when the event occurs, you can defer the subscriptions. In this way you can return control more quickly to the calling application and let the Event Manager execute any costly subscription processing at a later time.

You can defer subscription processing by three different methods:

- Raise the event with a future date in the SEND\_DATE attribute. Use this method when you want to defer all subscription processing for a locally raised event until a particular effective date.
- Define subscriptions to the event with phase numbers of 100 or higher. Use this method when you want to defer processing of particular subscriptions for either local or external events.
- Set the dispatch mode of the Event Manager to deferred processing before raising the event. This method can be used to defer all subscription processing for a locally raised event. This method is not recommended, however, and should only be used in exceptional circumstances.

When subscription processing for an event is deferred by any of these methods, the event message is placed on the standard WF\_DEFERRED queue associated with the WF\_DEFERRED agent. You must schedule a listener to monitor the WF\_DEFERRED agent. See: Scheduling Listeners for Local Inbound Agents: page 13 – 56.

The listener dequeues event messages from the WF\_DEFERRED agent in priority order. The event messages retain their original source type, whether Local or External. The amount of time by which subscription processing for these events is deferred depends on the schedule defined for the listener, and, for future-dated events, on the specified effective date.

### **Deferring Subscription Processing Using a Future Send Date**

You can defer subscription processing for a local event until a particular future effective date by raising the event with that date in the SEND\_DATE attribute. For example, you could enter information for a new employee in a human resources application as soon as the

employee was hired, but defer payroll processing until the employee's start date.

When an event is raised with a future send date, the Event Manager immediately places the event message on the WF\_DEFERRED queue, without executing any of the subscriptions for the event. All subscriptions to the event are deferred, regardless of their phase number. The event remains in a WAIT state until the send date. When the send date arrives, the event message becomes available for dequeuing and will be dequeued the next time an agent listener runs on the WF\_DEFERRED queue. The amount of time by which subscription processing is deferred depends on the send date you specify as well as on the schedule defined for the listener.

When the listener dequeues the event message, the Event Manager checks for a subscription ID in the ERROR\_SUBSCRIPTION attribute. If the event message does not contain a subscription ID, meaning that all subscription processing for the event was deferred immediately after the event was raised, then the Event Manager proceeds to execute all subscriptions to the event, in ascending phase order.

**Note:** If an event was deferred when it was raised, the Event Manager executes all eligible subscriptions to the event when the event is dequeued from the WF\_DEFERRED queue, regardless of the subscription phase numbers. Subscriptions will not be deferred a second time, even if they have a phase number of 100 or higher.

## See Also

Raise: page 8 – 261

### **Deferring Subscription Processing Using Subscription Phase Numbers**

You can also use the phase number for a subscription to control whether the subscription is executed immediately or is deferred. The Event Manager treats subscriptions with a phase number of 100 or higher as deferred subscriptions. Both Local and External subscriptions can be deferred in this way.

**Note:** For this deferral method to take effect when an event is raised locally, the event must not be raised with a future send date, and the Event Manager must be in the normal synchronous mode for subscription processing. Otherwise, the event message will immediately be placed on the WF\_DEFERRED queue, and the Event Manager will not

execute any subscriptions until the event is dequeued from there.

When a triggering event is raised or received, the Event Manager executes subscriptions to that event in phase order until it encounters a subscription with a phase number of 100 or higher. The Event Manager sets that subscription into the `ERROR_SUBSCRIPTION` attribute within the event message, as well as setting the priority specified in the subscription properties into the `PRIORITY` attribute. Then the event message is placed on the standard `WF_DEFERRED` queue.

The amount of time by which subscription processing is deferred depends on the schedule defined for the agent listener monitoring the `WF_DEFERRED` agent. When the listener dequeues an event message, the Event Manager checks for a subscription ID in the `ERROR_SUBSCRIPTION` attribute. If a subscription ID is present, meaning that subscription processing was deferred from that subscription onwards, the Event Manager begins by executing that subscription, and then continues executing any other subscriptions to the event with the same or a higher phase number.

**Note:** The Event Manager resumes subscription processing at the phase number of the subscription set into the event message. It does not necessarily begin with the phase number 100, if there were no subscriptions with that phase number when the event was originally processed.

### **Deferring Subscription Processing Using the Event Manager Dispatch Mode**

If you raise an event from a local application, you can also choose to defer all subscription processing for that event every single time it is raised. To do so, call the `SetDispatchMode()` API with the mode `'ASYNC'`, indicating deferred (asynchronous) processing, just before calling the `Raise()` API. This method is not recommended, however, and should only be used in exceptional circumstances, since it requires hard-coding the deferral in your application. To retain the flexibility to modify subscription processing without intrusion into the application, you can simply raise the event with a future send date or mark some or all of the individual subscriptions for deferral using the subscription phase numbers.

When an event is raised after the dispatch mode is set to deferred processing, the Event Manager immediately places the event message on the `WF_DEFERRED` queue, without executing any of the subscriptions for the event. All subscriptions to the event are deferred, regardless of their phase number.

The amount of time by which subscription processing is deferred depends on the schedule defined for the agent listener monitoring the WF\_DEFERRED agent. When the listener dequeues the event message, the Event Manager checks for a subscription ID in the ERROR\_SUBSCRIPTION attribute. If the event message does not contain a subscription ID, meaning that all subscription processing for the event was deferred immediately after the event was raised, then the Event Manager proceeds to execute all subscriptions to the event, in ascending phase order.

**Note:** If an event was deferred when it was raised, the Event Manager executes all eligible subscriptions to the event when the event is dequeued from the WF\_DEFERRED queue, regardless of the subscription phase numbers. Subscriptions will not be deferred a second time, even if they have a phase number of 100 or higher.

## See Also

SetDispatchMode: page 8 – 274

### **Error Handling**

---

If a rule function returns a status code of WARNING or ERROR, indicating that a warning condition or an error occurred during subscription processing, the Event Manager places the event message on the standard WF\_ERROR queue associated with the WF\_ERROR agent. For a WARNING status, the Event Manager then continues subscription processing for the event. For an ERROR status, the Event Manager halts subscription processing for the event and rolls back any subscriptions already executed for the event.

You must schedule a listener to monitor the WF\_ERROR agent. When this listener dequeues the event message from the WF\_ERROR queue, the message is assigned a source type of Error. The Event Manager then searches for and executes any subscriptions by the local system to that event or to the Any event with the source type Error. If no subscriptions are found, the Event Manager executes any subscriptions by the local system to the Unexpected event with the source type Error.

Oracle Workflow provides one predefined subscription to the Unexpected event with the source type Error. This subscription performs the default error handling for any errored event for which you have not defined a custom Error subscription. The subscription sends the event message to the Default Event Error process in the System: Error item type.





**Attention:** You must not change or disable the definition of the Unexpected event or of the predefined Error subscription to that event. If you do, the Event Manager will not be able to perform default error handling for event and subscription processing.

The Default Event Error process sends a notification to the system administrator. For a warning condition, no response is required. For an error, the process allows the system administrator to abort or retry the event subscription processing. See: Unexpected Event: page 14 – 12 and Default Event Error Process: page 6 – 34.

You can set up custom error handling for a particular event by defining a subscription to that event with a source type of Error and specifying the custom processing you want to execute as the subscription action. In this case, the Event Manager will not perform the default error handling, since the errored event will no longer be an unexpected event. Instead, your custom error handling will replace the default error handling.

**Note:** If a rule function raises an exception, the Event Manager rolls back all subscription processing for the event and raises the error to the calling application. In this case the event message is not placed on the WF\_ERROR queue.

## See Also

Standard API for an Event Subscription Rule Function: page 7 – 25

Scheduling Listeners for Local Inbound Agents: page 13 – 56

### ► To Define an Event Subscription

1. Use a web browser to connect to the following URL:

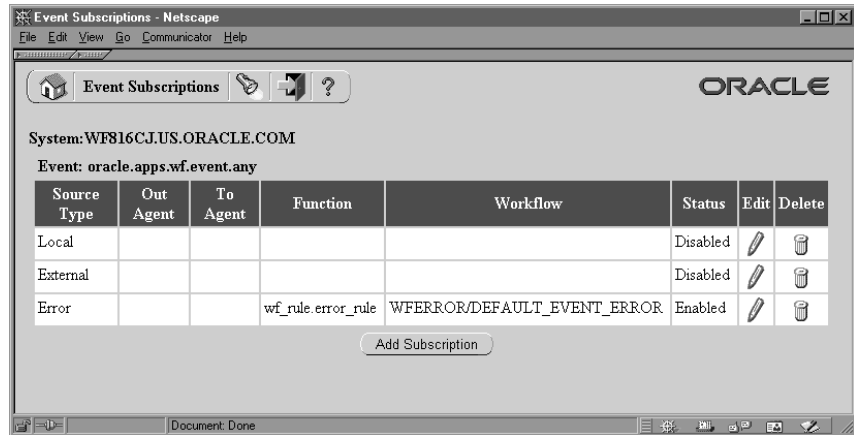
```
<webagent>/wf_event_html.listsubscriptions
```

Replace *<webagent>* with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



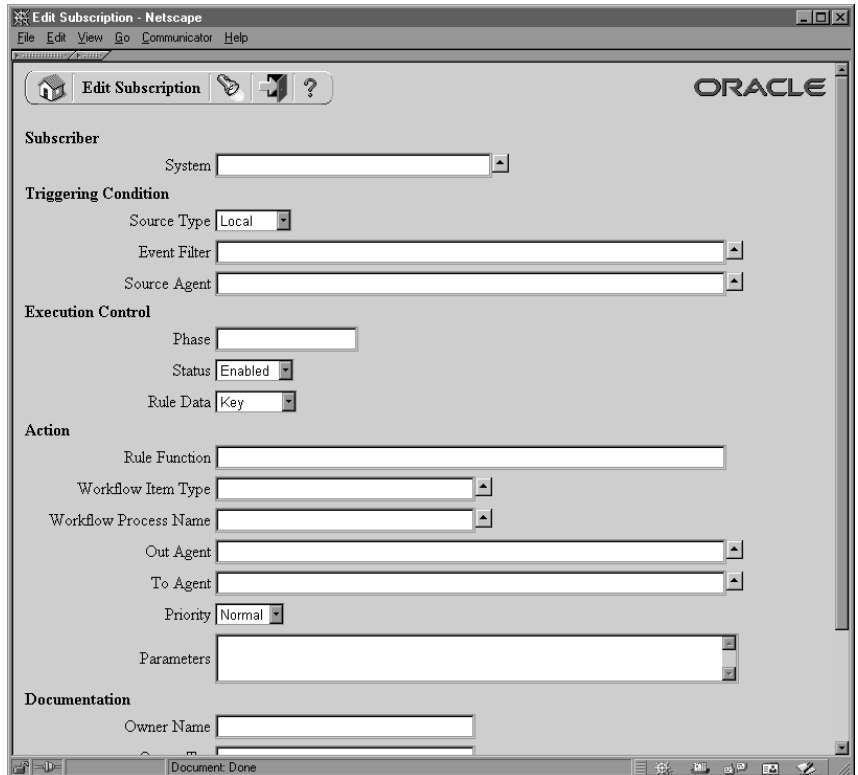
**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

**Note:** You can also access the Event Subscriptions web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 9 – 2.



2. The Event Subscriptions page appears, displaying a list of existing subscriptions grouped by the subscribing system and triggering event. The Event Subscriptions page summarizes the source type, out agent, to agent, function, workflow, and status of each subscription.

Choose the Add Subscription button to open the Edit Subscription page.



3. In the Subscriber region, enter the system where the subscription executes. Click on the System field's up arrow icon to display a list of systems from which to choose. See: Using a List of Values: page 13 – 22.
4. In the Triggering Condition region, specify the type of source system to which the subscription applies in the Source Type field.
  - Local—The subscription applies only to events raised on the subscribing system.
  - External—The subscription applies only to events received by an inbound agent on the subscribing system.
 

**Note:** All event messages received by an inbound agent on the subscribing system are considered to have an External source, whether the sending agent is located on a remote system or on the local system.
  - Error—The subscription applies to only to errored events dequeued from the WF\_ERROR queue.

5. Enter the event to which the subscription applies in the Event Filter field. You can specify an individual event or an event group. Click on the Event Filter field's up arrow icon to display a list of events from which to choose. See: Using a List of Values: page 13 – 22.
6. Enter an optional Source Agent to which the subscription applies. If you specify a source agent, then the subscription is executed only when the triggering event is received from that agent. Click on the Source Agent field's up arrow icon to display a list of agents from which to choose. See: Using a List of Values: page 13 – 22.

**Note:** In most cases, the Source Agent field is left blank.

7. In the Execution Control region, enter an optional Phase number for the subscription to specify the order in which subscriptions that apply to the same event are executed. The phase number also controls whether a subscription is executed immediately or is deferred.
8. Select Enabled or Disabled as the subscription status. If you disable a subscription, it still remains in the Event Subscriptions list for reference, but it can no longer be actively used to respond to events.
9. In the Rule Data field, specify the event information required by the subscription.
  - Key—The subscription requires only the event key.
  - Message—The subscription requires the complete event data.
10. In the Action region, define the subscription processing you want to perform when the triggering event occurs. Subscription processing can include:
  - Running a function on the event message.
  - Sending the event message to a workflow process.
  - Sending the event message to an agent.
11. If you want to run a function on the event message, enter the Rule Function to run. The rule function must be defined according to a standard API. See: Standard API for an Event Subscription Rule Function: page 7 – 25.

If you do not specify a rule function, Oracle Workflow runs a default rule function to send the event message to the workflow process and the agent that you specify.

**Note:** If you enter a rule function other than the default, Oracle Workflow does not automatically send the event

message to the specified workflow and agent. You must explicitly include the send processing in your custom rule function instead. You can still enter workflow and agent information in the Action region for your function to reference, however.

12. If you want to send the event message to a workflow process, enter the item type that the process belongs to in the Workflow Item Type field and the name of the process in the Workflow Process Name field. Click on each field's up arrow icon to display a list of values from which to choose. See: Using a List of Values: page 13 – 22.

**Note:** The list of values for the Workflow Process Name field includes only the runnable processes within the item type you specify.

13. If you want to send the event message to an agent, enter either the Out Agent that you want to send the outbound message, or the To Agent that you want to receive the inbound message, or both. Click on each field's up arrow icon to display a list of values from which to choose. See: Using a List of Values: page 13 – 22.

- If you specify both a To Agent and an Out Agent, Oracle Workflow places the event message on the Out Agent's queue for propagation, addressed to the To Agent.
- If you specify a To Agent without an Out Agent, Oracle Workflow selects an outbound agent on the subscribing system whose queue type matches the queue type of the To Agent. The event message is then placed on this outbound agent's queue for propagation, addressed to the To Agent.
- If you specify an Out Agent without a To Agent, Oracle Workflow places the event message on the Out Agent's queue without a specified recipient. The Out Agent must use either a multi-consumer queue with a subscriber list or a single-consumer queue.

**Note:** The Out Agent must be located on the subscribing system. The list of values for the Out Agent field includes only agents with a direction of Out.

The list of values for the To Agent field includes only agents with a direction of In.

14. If you want to send the event message to an agent, select Normal, High, or Low as the priority with which the recipient should dequeue the message.

- Optionally enter any additional parameters for the rule function in the Parameters field. Use spaces to separate the parameters, and specify the name and value for each parameter in the following format:

`<name1>=<value1> <name2>=<value2> ... <nameN>=<valueN>`

**Note:** If you send the event message to a workflow process and you want to specify additional parameters to set as item attributes for the process, you can enter these parameters in the Parameters field for a subscription and use `WF_RULE.SetParametersIntoParameterList()` in the subscription rule function to set the subscription parameters into the event message parameter list. The event parameters will then be set as item attributes for the workflow process when the process receives the event. See: `SetParametersIntoParameterList`: page 8 – 289.

- In the Documentation region, you can optionally identify the program or application that owns the subscription by entering the program name in the Owner Name field and the program ID in the Owner Tag field. The Owner Name and Owner Tag are not required if you are defining a subscription manually in the Edit Subscription page. However, if you use a program to create subscription definitions automatically, the Event Manager displays the owner information set by that program in these fields. You can use the Edit Subscription page to update this information manually if necessary.
- Enter an optional description for the subscription.
- Choose the Submit button to save the subscription and return to the Event Subscriptions page. The Event Subscriptions page displays an updated list of subscriptions.

You can also choose the Cancel button to return to the Event Subscriptions page without saving the subscription.

### ► To Find Event Subscriptions

- Use a web browser to connect to the following URL:

`<webagent>/wf_event_html.findsubscription`

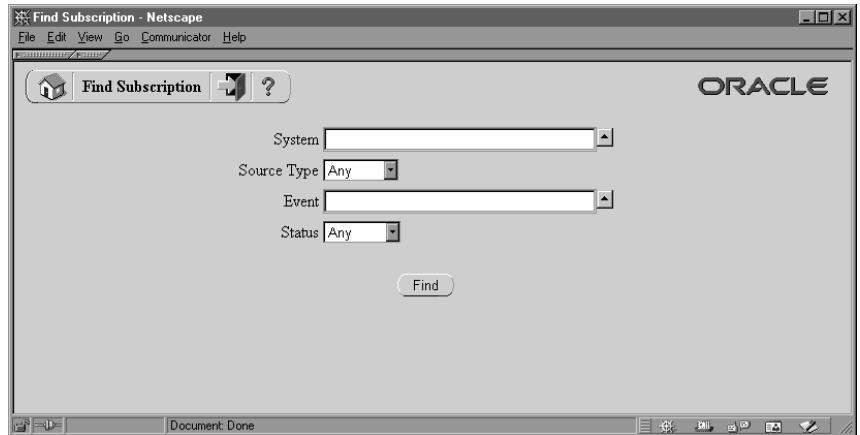
Replace `<webagent>` with the base URL of the web agent configured for Oracle Workflow in your Web server. See: `Setting Global User Preferences`: page 2 – 14.



**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you

will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

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

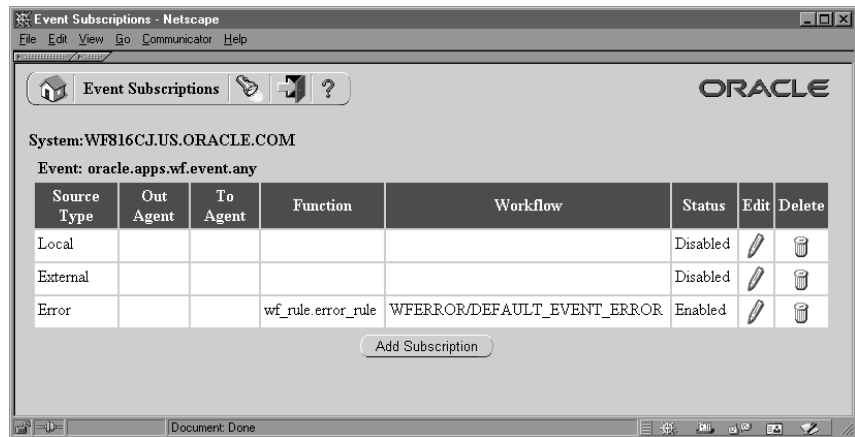


2. The Find Subscription page appears. The Find Subscription page lets you enter search criteria to locate specific event subscriptions. The search criteria are:
  - System—select the system for which you want to display subscriptions. Click on the field's up arrow icon to display a list of systems from which to choose. See: *Using a List of Values*: page 13 – 22.
  - Source Type—choose Local, External, or Error as the type of source system for which you want to display subscriptions, or choose Any to display subscriptions for any type of source system.
  - Event—select the event for which you want to display subscriptions. Click on the field's up arrow icon to display a list of events from which to choose. See: *Using a List of Values*: page 13 – 22.
  - Status—choose Enabled or Disabled as the status of the subscriptions you want to display, or choose Any to display subscriptions of any status.
3. Choose the Find button. The Event Subscriptions page appears, displaying a list of subscriptions that match your search criteria.

If your search criteria included a system or an event, you can choose the Add Subscription button to open the Edit Subscription page with the system and event information you specified automatically entered in the System and Event Filter fields, respectively. You can begin defining a new subscription in this way even if your search did not find any existing subscriptions matching your criteria.

► **To Update or Delete an Event Subscription**

1. Locate the subscription you want in the Event Subscriptions page. You can use the Find Subscription page to find the subscription that you want and display the subscription in the Event Subscriptions page. See: To Find Event Subscriptions: page 13 – 50.



2. To update a subscription, choose the pencil icon in the Edit column for that subscription. The Edit Subscription page appears. Make your changes to the subscription definition and save your work. See: To Define an Event Subscription: page 13 – 45.
3. To delete a subscription, choose the trash icon in the Delete column for that subscription, and choose OK in the confirmation window that appears. You can also choose Cancel in the confirmation window to return to the Event Subscriptions page without deleting the subscription.



---

## Setting Up Message Propagation

After you define your events, systems, agents, and subscriptions, you must set up the Business Event System for message propagation. Use the Check Setup web page in the Event Manager to perform the following steps:

1. Check the Business Event System setup.
2. Schedule listeners for local inbound agents.
3. Schedule propagations for local outbound agents.

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications, you should use the Concurrent Manager to schedule listeners for your local inbound agents. Use the Check Setup web page only to review and verify your setup and to schedule propagations for your local outbound agents.

You should recheck your setup whenever you make changes to your agents that affect the physical implementation required for propagation. See: Agents: page 13 – 22.



**Attention:** Oracle Workflow sets the status of the local system to Enabled by default. After you finish setting up the Business Event System, you can use the Global Workflow Preferences web page to set the system status that you want for event processing. See: Setting Global User Preferences: page 2 – 14.

---

### Checking the Business Event System Setup

Use the Check Setup web page to verify that the required parameters and components have been set up to enable message propagation for the Business Event System.

► **To Check the Business Event System Setup**

1. Use a web browser to connect to the following URL:

`<webagent>/wf_setup.check_all`

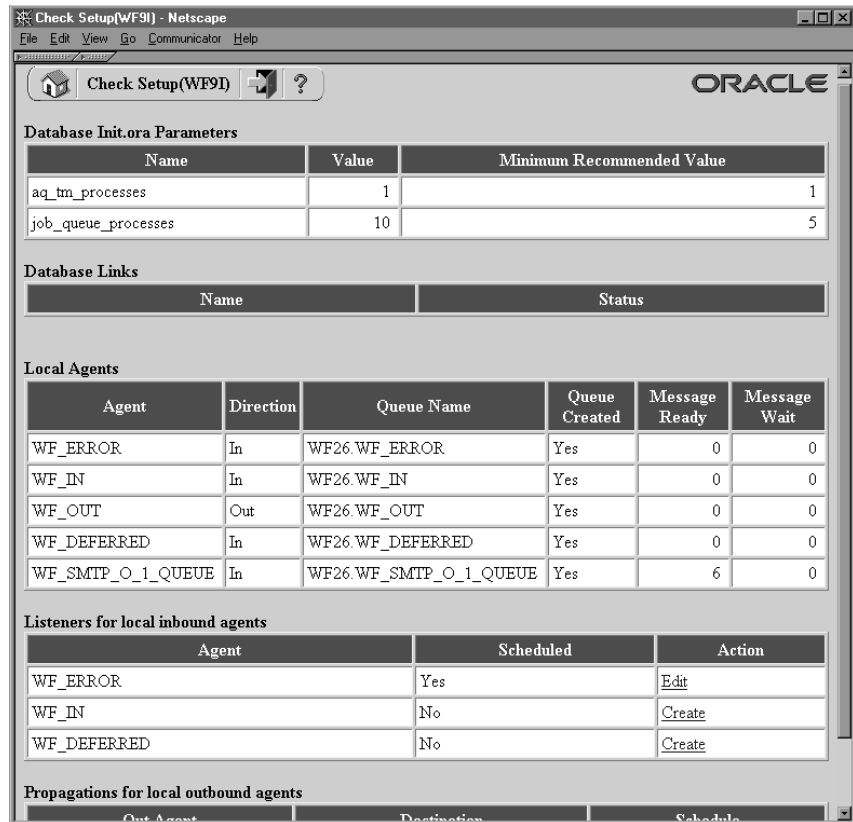
Replace `<webagent>` with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



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

have workflow administrator privileges to access the Event Manager web pages.

**Note:** You can also access the Check Setup web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 9 – 2.



2. The Check Setup page appears, displaying the propagation settings and components for your local system.
3. Use the Database Init.ora Parameters region to verify your settings for the database initialization parameters related to AQ. The Check Setup page displays the actual value defined for each parameter as well as the minimum recommended value for Oracle Workflow.

To modify any of these parameters when using Oracle8i, change the settings in the init.ora file for your database. Then you must restart your database to make the changes effective.

If you are using Oracle 9i, you can either modify the parameters in the init.ora file and restart the database, or you can use the ALTER SYSTEM statement to dynamically modify the values for AQ\_TM\_PROCESSES and JOB\_QUEUE\_PROCESSES for the duration of the instance.

- **AQ\_TM\_PROCESSES**—This parameter enables the time manager process in Oracle Advanced Queuing (AQ). The time manager process is required by Oracle Workflow to monitor delay events in queues, as in the case of the Oracle Workflow standard Wait activity. The minimum recommended number of time manager processes for Oracle Workflow is one.
- **JOB\_QUEUE\_INTERVAL**—If you are using Oracle8i, specify the job queue interval to determine how frequently each SNP job queue process in your instance wakes up. Oracle Workflow requires the job queue interval to be less than or equal to the latency parameter defined for your AQ propagation schedules, to allow queues to be rechecked for messages with the specified latency. The recommended job queue interval for Oracle Workflow is five seconds.

**Note:** Because the JOB\_QUEUE\_INTERVAL parameter is desupported in Oracle9i, the Check Setup page does not display this parameter if you are using Oracle9i, and you do not need to set a value for it.

- **JOB\_QUEUE\_PROCESSES**—This parameter defines the number of SNP job queue processes for your instance. Oracle Workflow requires job queue processes to handle propagation of Business Event System event messages by AQ queues. You must start at least one job queue process to enable message propagation. The minimum recommended number of processes for Oracle Workflow is two.

**Note:** You can use another initialization parameter, EVENT, for detailed database level tracing of issues related to AQ. Add the following line to your init.ora file:

```
event = "24040 trace name context forever, level 10"
```

Then restart your database to make this change effective. Be aware that using this parameter may generate large trace files.

4. Use the Database Links region to verify your database links. The Check Setup page displays the name and status of each database link that is referenced in an agent's address. You should create any required database links that do not yet exist. See: Creating Database Links: page 2 – 96.

**Note:** Ensure that the database link names used in your agents' addresses are exactly the same as the database link names specified when the database links were created.

5. Use the Local Agents region to verify the queues that are set up for the agents defined on your local system. The Check Setup web page displays the name and direction of each agent, the name of the queue assigned to the agent, whether or not the queue has been created, how many messages on the queue have been processed and are ready to be consumed, and how many messages are still waiting to be processed. You should create any required queues that do not yet exist. See: Agents: page 13 – 22 and Setting Up Queues: page 2 – 97.

**Note:** In addition to Business Event System agents, the Local Agents region also displays the standard agent defined for the Notification Mailer SMTP queue, WF\_SMTP\_O\_1\_QUEUE. You can review the information for this agent to check the number of notification messages on the Notification Mailer queue. The WF\_SMTP\_O\_1\_QUEUE agent is not used by the Business Event System, however. See: Standard Agents: page 13 – 25.

6. Use the Listeners for Local Inbound Agents region to schedule listeners to receive inbound event messages. See: Scheduling Listeners for Local Inbound Agents: page 13 – 56.
7. Use the Propagations for Local Outbound Agents region to schedule propagations to send event messages. See: Scheduling Propagations for Local Outbound Agents: page 13 – 61.

## See Also

*Oracle Reference*

*Oracle Application Developer's Guide – Advanced Queuing*

### **Scheduling Listeners for Local Inbound Agents**

---

Use the Check Setup web page to schedule listeners for the inbound agents on your local system. The Business Event System requires listeners to be scheduled to receive inbound event messages.

When you schedule a listener for an agent, it starts monitoring the agent's queue on the run day you specify, dequeuing any inbound event messages. When an event message is received, the Event Manager searches for and executes any active subscriptions by the local system to that event with a source type of External, and also any active

subscriptions by the local system to the Any event with a source type of External.

The listener exits after all event messages on the agent's queue have been dequeued. Oracle Workflow reruns the listener indefinitely, at the interval you specify.

After you schedule a listener, you can modify its schedule by updating its settings. You can also stop a listener altogether by deleting it.

You must schedule listeners for the standard WF\_DEFERRED and WF\_ERROR agents to enable deferred subscription processing and error handling for the Business Event System, respectively. Also, if you want to use the standard WF\_IN agent for event message propagation, schedule a listener for that agent as well.

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications, you should use the Concurrent Manager to schedule listeners for your local inbound agents by submitting the Workflow Agent Listener concurrent program. Use the Check Setup web page only to review and verify your setup and to schedule propagations for your local outbound agents.

If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager to submit and manage the Workflow Agent Listener concurrent program. For more information, please refer to the Oracle Applications Manager online help.

If you are using the standalone version of Oracle Workflow available with Oracle9i Release 2, you can use the standalone Oracle Workflow Manager component available through Oracle Enterprise Manager to submit and manage Workflow agent listener database jobs. For more information, please refer to the Oracle Workflow Manager online help.

**Note:** You must not run an agent listener on the standard WF\_SMTP\_O\_1\_QUEUE agent. This agent is defined for the Notification Mailer SMTP queue and is not used by the Business Event System. See: Standard Agents: page 13 – 25.

## See Also

Agents: page 13 – 22

Listen(): page 8 – 270

Workflow Agent Listener Concurrent Program: page 8 – 272

► **To Schedule a Listener for a Local Inbound Agent**

1. Use a web browser to connect to the following URL:

```
<webagent>/wf_setup.check_all
```

Replace *<webagent>* with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

**Note:** You can also access the Check Setup web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 9 – 2.

Check Setup(WF9I) - Netscape

File Edit View Go Communicator Help

Check Setup(WF9I) ?

ORACLE

**Database Init.ora Parameters**

Name	Value	Minimum Recommended Value
aq_tm_processes	1	1
job_queue_processes	10	5

**Database Links**

Name	Status
------	--------

**Local Agents**

Agent	Direction	Queue Name	Queue Created	Message Ready	Message Wait
WF_ERROR	In	WF26.WF_ERROR	Yes	0	0
WF_IN	In	WF26.WF_IN	Yes	0	0
WF_OUT	Out	WF26.WF_OUT	Yes	0	0
WF_DEFERRED	In	WF26.WF_DEFERRED	Yes	0	0
WF_SMTP_O_1_QUEUE	In	WF26.WF_SMTP_O_1_QUEUE	Yes	6	0

**Listeners for local inbound agents**

Agent	Scheduled	Action
WF_ERROR	Yes	<a href="#">Edit</a>
WF_IN	No	<a href="#">Create</a>
WF_DEFERRED	No	<a href="#">Create</a>

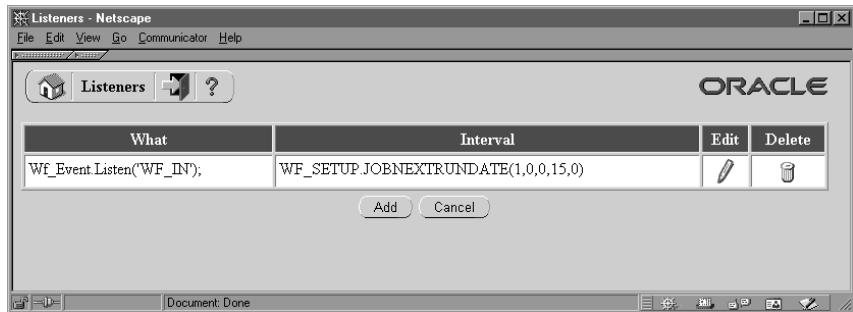
**Propagations for local outbound agents**

Out Agent	Destination	Schedule
-----------	-------------	----------

- The Check Setup page appears, displaying the propagation settings and components for your local system.
- Locate the agent you want in the Listeners for Local Inbound Agents region. The Check Setup page summarizes the agent name and Scheduled status for each local inbound agent.
- If no listeners are scheduled for the agent yet, the Scheduled status for the agent is No. Choose the Create link in the Action column for that agent to create the first listener for the agent. The Edit Listener page appears, displaying the name of the selected agent.

If at least one listener is already scheduled for the agent, the Scheduled status for the agent is Yes. Choose the Edit link in the Action column for that agent to create an additional listener. The Listeners page appears, displaying a list of existing listeners for the agent. The Listeners page summarizes each Listen procedure that runs for the agent as well as the interval at which the procedure is resubmitted.

Choose the Add button. The Edit Listener page appears, displaying the name of the selected agent. You can also choose the Cancel button to return to the Check Setup page without creating a new listener.



- In Edit Listener page, enter the date on which you want to start running the listener in the Run Day field. To start the listener on the current system date, leave this field blank.

**Note:** If you have set your user preferences to a date format that includes a time setting, then you can specify the time when you want to start the listener, as well as the date. Otherwise, you can only specify a date. See: Setting User Preferences: page 9 – 6

- In the Run Every fields, enter an interval to specify how often you want the listener to be run. You can specify the interval in days, hours, minutes, and seconds.
- Choose the Submit button to save the listener schedule. If you are scheduling the first listener for the agent, the Check Setup page

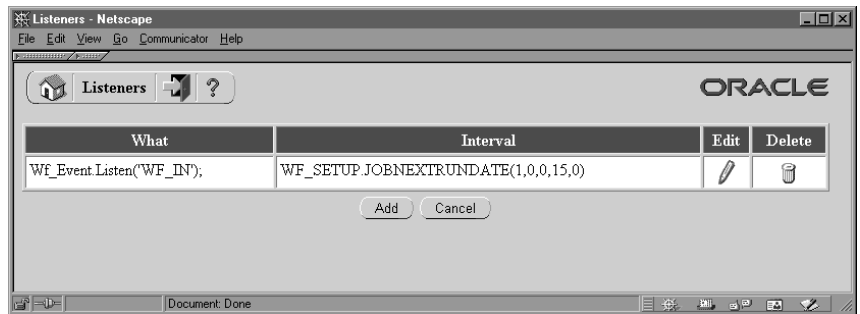


appears, displaying an updated Scheduled status. If you are scheduling an additional listener, the Listeners page appears, displaying an updated list of listeners.

You can also choose the Cancel button to return to the Check Setup page or the Listeners page without saving the listener.

### ► To Update or Delete a Listener

1. Use the Listeners for Local Inbound Agents region of the Check Setup page to locate the inbound agent that you want. See: To Schedule a Listener for a Local Inbound Agent: page 13 – 58.
2. If the Scheduled status for the agent is Yes, at least one listener is already scheduled for the agent. Choose the Edit link in the Action column to open the Listeners page, displaying a list of existing listeners.



3. To update a listener, choose the pencil icon in the Edit column for that listener. The Edit Listener page appears. Make your changes to the listener schedule and save your work. See: To Schedule a Listener for a Local Inbound Agent: page 13 – 58.
4. To delete a listener, choose the trash icon in the Delete column for that listener, and choose OK in the confirmation window that appears. You can also choose Cancel in the confirmation window to return to the Listeners page without deleting the listener.

### Scheduling Propagations for Local Outbound Agents

---

Use the Check Setup web page to schedule propagations for the outbound agents on your local system. The Business Event System requires propagations to be scheduled to send outbound event messages.

When you send an event message to an agent, the Event Manager places the message on the queue associated with the outbound agent. The message is then asynchronously delivered to the inbound agent by propagation. The Check Setup page lets you verify whether the required propagations are scheduled and schedule AQ propagations for agents that use the SQLNET protocol.

For agents that use other protocols, you must provide external propagation logic. See: Agents: page 13 – 22

If you want to use the standard WF\_OUT agent for event message propagation, ensure that you schedule propagations for that agent.

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager to review the propagation schedules for your local outbound agents. For more information, please refer to the Oracle Applications Manager online help.

If you are using the standalone version of Oracle Workflow available with Oracle9i Release 2, you can use the standalone Oracle Workflow Manager component available through Oracle Enterprise Manager to review the propagation schedules for your local outbound agents. For more information, please refer to the Oracle Workflow Manager online help.

## See Also

Agents: page 13 – 22

### ► To Schedule a Propagation for a Local Outbound Agent

1. Use a web browser to connect to the following URL:

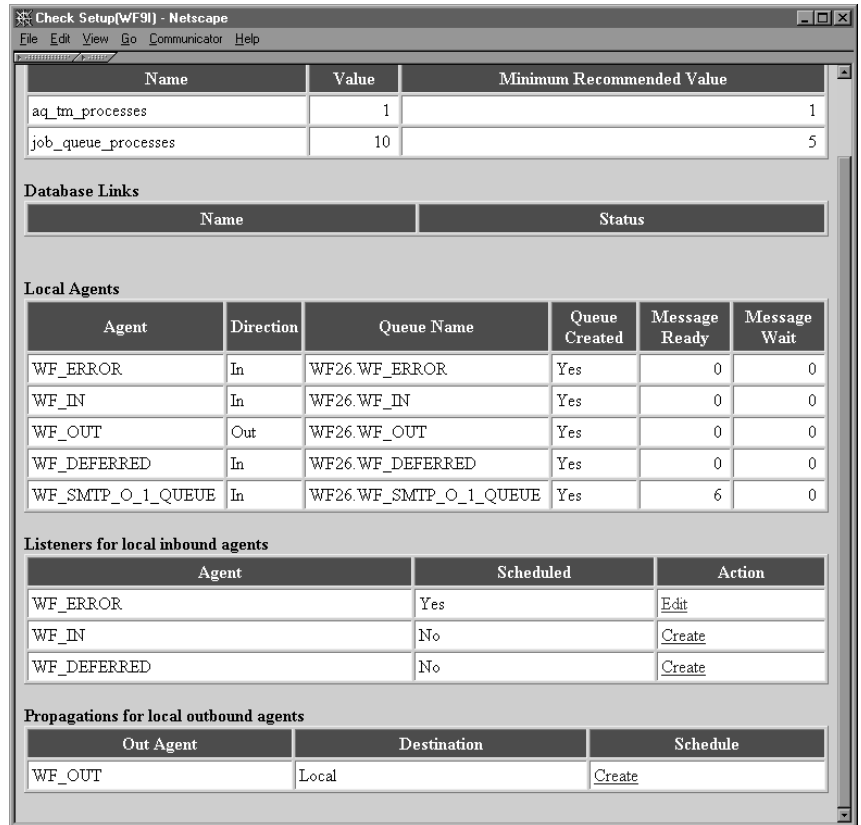
```
<webagent>/wf_setup.check_all
```

Replace *<webagent>* with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

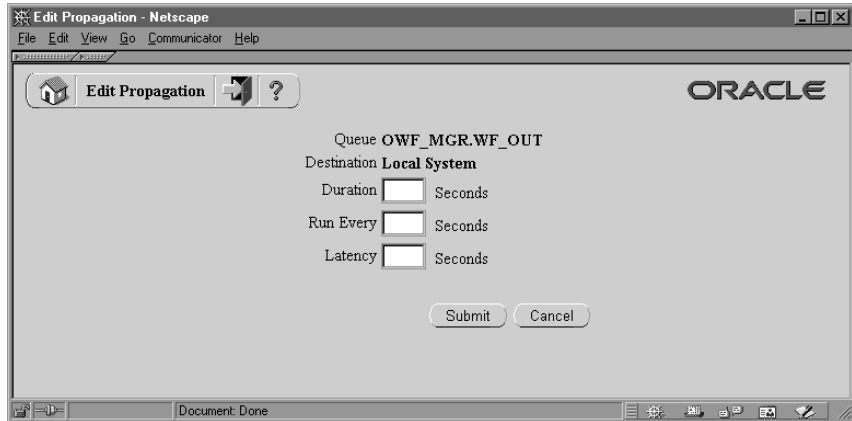
**Note:** You can also access the Check Setup web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 9 – 2.



2. The Check Setup page appears, displaying the propagation settings and components for your local system.
3. The Propagations for Local Outbound Agents region displays a list of the combinations of local outbound agents and database links that may require propagations. This list matches each of your local outbound agents with each database link to a remote system that appears in the addresses of the inbound agents you have defined. Each local outbound agent is also listed with a Local destination in the Database Link column for propagation to inbound agents on the local system. See: Agents: page 13 – 22

Locate the outbound agent and database link combination for which you want to schedule a propagation.

- If no propagation is scheduled yet for the agent and database link you want, choose the Create link in the Schedule column to schedule the propagation. The Edit Propagation page appears, displaying the name of the outbound agent's queue, as well as the database link name for a remote destination or Local System for a local destination.



- In the Duration field, enter the duration of the propagation window, in seconds.
- In the Run Every field, enter an interval in seconds to specify how often you want a propagation window to occur.

**Note:** The run interval must be longer than the duration of the propagation window.

- In the Latency field, enter a latency time in seconds to specify how long you want to wait, after all messages have been propagated, before rechecking the queue for new messages to the destination.

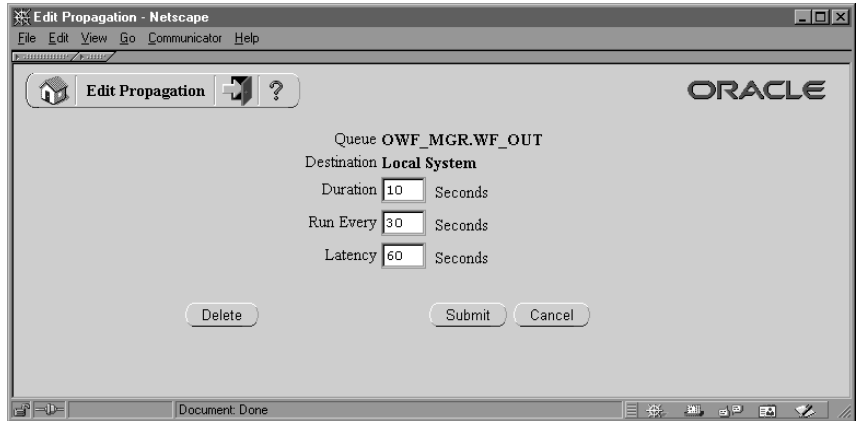
The latency represents the maximum wait time during the propagation window for a message to be propagated after it is enqueued. To propagate messages as soon as possible after they are enqueued, enter a latency of zero. The default latency is 60 seconds.

**Note:** To enable AQ to enforce the latency time, the job queue interval setting in your database initialization parameters must be less than or equal to the latency value. See: *Checking the Business Event System Setup*: page 13 – 53.

- Choose the Submit button to save the propagation schedule and return to the Check Setup page. You can also choose the Cancel button to return to the Check Setup page without saving the propagation.

► **To Update or Delete a Propagation**

1. Use the Propagations for Local Outbound Agents region of the Check Setup page to locate the outbound agent and database link combination that you want. See: To Schedule a Propagation for a Local Outbound Agent: page 13 – 62.
2. If a propagation is already scheduled, choose the Edit link in the Schedule column to open the Edit Propagation page, displaying the propagation settings.



3. To update the propagation, make your changes to the settings and save your work. See: To Schedule a Propagation for a Local Outbound Agent: page 13 – 62.
4. To delete the propagation, choose the Delete button. You can also choose the Cancel button to return to the Check Setup page without changing or deleting the propagation.

**Note:** You may not be able to delete a propagation until the propagation window closes.

---

## Raising Events

In addition to raising events from your applications or through workflows, you can raise events that do not require additional parameters manually using the Raise Event web page for testing purposes. When you raise an event, the Event Manager searches for and executes any active subscriptions by the local system to that event

with a source type of Local, and also any active subscriptions by the local system to the Any event with a source type of Local.

### ► To Raise an Event

1. Use a web browser to connect to the following URL:

`<webagent>/wf_event_html.entereventdetails`

Replace `<webagent>` with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

**Note:** You can also access the Raise Event web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 9 – 2.

2. The Raise Event page appears.
3. In the Event Name field, select the event that you want to raise.

4. Enter an event key that uniquely identifies this instance of the event.
5. Optionally enter event data to describe what occurred in the event.

**Note:** The maximum length of the data you can enter in the Event Data field is 32 kilobytes. If the event data exceeds 32 Kb, you should assign a Generate function in the event definition to generate the event data, rather than entering the data directly in the Event Data field. See: *To Define an Event: page 13 – 5.*

You can also choose to raise the event using the WF\_EVENT.Raise API instead. This method lets you provide the event data as a CLOB storing up to four gigabytes of data. See: *Raise: page 8 – 261.*

6. Choose the Submit button to raise the event to the Event Manager. You can also choose the Cancel button to return to the Oracle Workflow home page without raising the event.

If you choose the Submit button, Oracle Workflow raises the event and displays a confirmation message with the event name and event key. Choose the OK button to return to the Raise Event page.

---

## Signing Up Systems

Before you can send business events from one system to another, you must sign up the destination system with the source system as a potential recipient of event messages. Signing up a system means defining the destination system as well as its inbound agents in the Event Manager of the source system, so that event messages from the source system can be addressed to the destination agents.

Usually, both systems should be signed up with each other, so that each system can both send messages to and receive messages from the other system.

To sign up a destination system for receiving event messages from a source system, perform the following steps:

1. Retrieve the local system and inbound agent definitions, which together make up the system identifier information, from the destination system. You can use the System Identifier web page on the destination system to generate an XML document containing the system identifier information. See: *To Retrieve System Identifier Information: page 13 – 68.*

**Note:** If you do not have access to the Oracle Workflow installation on the destination system, ask the workflow administrator for that system to perform this step.

2. Add the destination system identifier information to the Event Manager in the source system. You can use the System Signup web page on the source system to add the information by raising the System Signup event with the XML document from the destination system as the event data. When the System Signup event is raised on the source system, Oracle Workflow executes a predefined subscription that adds the system identifier information to the Event Manager in that system. See: *To Sign Up a System*: page 13 – 69.

**Note:** If you do not have access to the Oracle Workflow installation on the source system, ask the workflow administrator for that system to perform this step.

### ► To Retrieve System Identifier Information

1. Use a web browser to connect to the following URL on the system you want to sign up as a destination system:

```
<webagent>/wf_event_html.getsystemidentifier
```

Replace *<webagent>* with the base URL of the web agent configured for Oracle Workflow in your Web server. See: *Setting Global User Preferences*: page 2 – 14.



**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

If you do not have access to the Oracle Workflow installation on the destination system, ask the workflow administrator for that system to perform this step.

**Note:** You can also access the System Identifier web page from the Oracle Workflow home page. See: *Accessing the Oracle Workflow Home Page*: page 9 – 2.

2. Oracle Workflow produces the system identifier XML document, which contains the definitions of the local system and its inbound agents. Save this document as a text file. You can then copy the document and enter it as the event data for the System Signup event when you sign this system up with a source system. See: *To Sign Up a System*: page 13 – 69.



## ► To Sign Up a System

1. Use a web browser to connect to the following URL on the source system where you want to sign up a destination system:

```
<webagent>/wf_event_html.entereventdetails?p_event_name=oracle.apps.wf.event.system.signup
```

Replace *<webagent>* with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

If you do not have access to the Oracle Workflow installation on the source system, ask the workflow administrator for that system to perform this step.

**Note:** You can also access the System Signup web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 9 – 2.

The screenshot shows a Netscape browser window titled "System Signup - Netscape". The browser's address bar is empty. The page content includes a navigation bar with "System Signup" and a help icon. The Oracle logo is in the top right corner. The main form area contains the following elements:

- Event Name:** A dropdown menu with the value "oracle.apps.wf.event.system.signup".
- Event Key:** An empty text input field.
- Event Data:** A large, empty text area for entering details.
- Buttons:** "Submit" and "Cancel" buttons at the bottom of the form.

The browser's status bar at the bottom shows "Document Done".

2. The System Signup page appears, displaying the internal name of the System Signup event.
3. Enter an event key that uniquely identifies this instance of the event.
4. Copy the XML document containing the destination system identifier information into the Event Data field. See: To Retrieve System Identifier Information: page 13 – 68.
5. Choose the Submit button to raise the System Signup event to the Event Manager. A confirmation message is displayed. When the System Signup event is raised, Oracle Workflow executes a predefined subscription that adds the system identifier information from the event data to the Event Manager. See: System Signup Event: page 14 – 9.

You can also choose the Cancel button to return to the Oracle Workflow home page without raising the System Signup event.

---

## Synchronizing Systems

Synchronizing systems means replicating all the Event Manager objects that are defined on the source system to the target system. You can use the Synchronize Event Systems event to synchronize systems with each other.

### ► To Synchronize Systems

1. Sign up the source and target systems with each other. See: Signing Up Systems: page 13 – 67.
2. On the source system, modify the predefined subscription to the Seed Event Group with the Local source type.
  - Specify the inbound agent on the target system that you want to receive the event message, or specify a workflow process that sends the event message to the target system.

**Note:** If you want to send the event message to more than one target system, you can either define additional subscriptions or specify a workflow process that sends the event message to multiple systems.
  - Enable the subscription.

- Note:** If you do not have access to the Oracle Workflow installation on the source system, ask the workflow administrator for that system to perform this step.
3. On the target system, enable the predefined subscription to the Seed Event Group with the External source type.

**Note:** If you do not have access to the Oracle Workflow installation on the target system, ask the workflow administrator for that system to perform this step.
  4. On the source system, raise the Synchronize Event Systems event (oracle.apps.wf.event.all.sync) using the Raise Event web page. Enter a unique event key, but leave the Event Data field blank. See: Raising Events: page 13 – 65

**Note:** If you do not have access to the Oracle Workflow installation on the source system, ask the workflow administrator for that system to perform this step.

When the Synchronize Event Systems event is raised on the source system, it triggers the subscription to the Seed Event Group with the Local source type. The Event Manager generates the event message, which contains the definitions of all the Event Manager objects on the local system, including events, event groups, systems, agents, and subscriptions. Then the event message is sent to the specified inbound agent on the target system, or to the specified workflow process that sends the event message to the target system.

When the Synchronize Event Systems event is received on the target system, it triggers the subscription to the Seed Event Group with the External source type. Oracle Workflow loads the object definitions from the event message into the Event Manager on the target system, creating new definitions or updating existing definitions as necessary.

### **Automatic Replication**

---

After you enable the predefined subscriptions in steps 2 and 3, these subscriptions will also replicate any subsequent changes you make to Event Manager object definitions on the source system. Whenever you create, update, or delete events, event group members, systems, agents, or subscriptions, Oracle Workflow raises the corresponding predefined events. These events trigger the Local subscription to the Seed Event Group on the source system, which sends the object definition data to the target system. The External subscription to the Seed Event Group

on the target system receives the data and adds, updates, or deletes the object definition in the Event Manager there.

If you do not want to continue automatically replicating changes on the source system to the target system, you can either disable the subscriptions after you finish synchronizing the systems, or disable the predefined events corresponding to those changes.

### **Master/Copy Systems**

---

If you choose, you can treat one system as a master system that replicates its own Event Manager object definitions to its associated copy systems, but does not accept any object definition changes from those systems. To set up master/copy replication, perform the steps to synchronize the target copy systems with the source master system, as usual. Then, to prevent object definitions from being sent from the copy systems, ensure that the Local subscription to the Seed Event Group on the copy systems is disabled. To prevent object definitions from being received into the master system, ensure that the External subscription to the Seed Event Group on the master system is disabled as well.

### **See Also**

Predefined Workflow Events: page 14 – 2

Synchronize Event Systems Event: page 14 – 5

Seed Event Group: page 14 – 6

To Define an Event Subscription: page 13 – 45

---

## **Reviewing Local Queues**

You can use the Event System Local Queues page to review the local queues used by the Business Event System as well as the messages currently being held on those queues.

**Note:** You can also use the Check Setup page to verify the setup of your local queues. See: Checking the Business Event System Setup: page 13 – 53.

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager to review the statuses of the messages being held on

local queues. For more information, please refer to the Oracle Applications Manager online help.

If you are using the standalone version of Oracle Workflow available with Oracle9i Release 2, you can use the standalone Oracle Workflow Manager component available through Oracle Enterprise Manager to review the statuses of the messages being held on local queues. For more information, please refer to the Oracle Workflow Manager online help.

**Note:** In addition to Business Event System agents, the Event System Local Queues page also displays the standard agent defined for the Notification Mailer SMTP queue, WF\_SMTP\_O\_1\_QUEUE. You can review the information for this agent to check the number of notification messages on the Notification Mailer queue. The WF\_SMTP\_O\_1\_QUEUE agent is not used by the Business Event System, however. You cannot view message details for this queue because it does not use WF\_EVENT\_T as its payload type. See: Standard Agents: page 13 – 25.

### ► To Review Local Queues

1. Use a web browser to connect to the following URL:

`<webagent>/wf_event_html.eventqueuedisplay`

Replace `<webagent>` with the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.



**Attention:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager web pages.

**Note:** You can also access the Event System Local Queues web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 9 – 2.

Protocol	Agent Name	Direction	Message Count	View Detail
SQLNET	WF_DEFERRED	IN	0	
SQLNET	WF_ERROR	IN	0	
SQLNET	WF_IN	IN	0	
SQLNET	WF_OUT	OUT	0	
SQLNET	WF_SMTP_O_1_QUEUE	IN	6	Not Available

- The Event System Local Queues page appears, displaying a list of the local queues used by the Business Event System. For each queue, the Event System Local Queues page summarizes the protocol and name of the agent with which the queue is associated, whether the agent is used for inbound or outbound communication on the system, and the number of messages currently being held on the queue.
- You can review message details for queues that use the standard WF\_EVENT\_T datatype as their payload type. To review message details for a queue, choose the flashlight icon in the View Detail column for that queue.

**Note:** You cannot view message details for queues that do not use WF\_EVENT\_T as their payload type.



4. The Find Standard Event Queue Messages page appears, displaying the name of the queue you selected. The Find Standard Event Queue Messages page lets you enter search criteria to locate specific event messages. The search criteria are:
  - Event Name—enter the internal event name for the event messages you want to display.
  - Event Key—enter the event key for the event messages you want to display.
  - Status—choose Ready, Wait, Processed, or Expired as the status of the event messages you want to display, or choose Any to display messages of any status.
5. Choose the Go button. The Local Queue Messages page appears, displaying a list of event messages on the queue you selected that match your search criteria. For each message, the Local Queue Messages page summarizes the event name, event key, correlation ID, event parameters, From System that sent the message, To System that received the message, send date, error message, error stack, and the message status.

The screenshot shows a Netscape browser window titled "Local Queue Messages - Netscape". The browser's address bar contains "Local Queue Messages" and the Oracle logo is visible in the top right corner. Below the browser interface is a table with the following columns: Event Name, Event Key, Correlation ID, Parameters, From System, To System, Send Date, Error Message, Error Stack, Message State, and Event Data (XML Format). The table contains five rows of data, all with a "Ready" message state and a flashlight icon in the "Event Data (XML Format)" column.

Event Name	Event Key	Correlation ID	Parameters	From System	To System	Send Date	Error Message	Error Stack	Message State	Event Data (XML Format)
oracle.apps.ecx.bes.qa.t5	5734	ecx_generate_5a		WF_OUT@IV51	WF_IN@IV51	09-OCT-2001 11:59:56			Ready	
oracle.apps.ecx.bes.qa.t5	5734	ecx_gena2a_5c		WF_OUT@IV51	WF_IN@IV51	09-OCT-2001 12:00:02			Ready	
oracle.apps.ecx.bes.qa.t5	5734	ecx_gena2a_5d		WF_OUT@IV51	WF_IN@IV51	09-OCT-2001 12:00:05			Ready	
oracle.apps.ecx.bes.qa.t5	5734	ecx_gena2a_5f		WF_OUT@IV51	WF_IN@IV51	09-OCT-2001 12:00:11			Ready	
oracle.apps.ecx.bes.qa.t5	5734	ecx_gena2a_5g		WF_OUT@IV51	WF_IN@IV51	09-OCT-2001 12:00:14			Ready	

6. To review the event data for a message as an XML document, choose the flashlight icon in the Event Data (XML Format) column for that message.
7. To review the event data for a message as a text document, choose the flashlight icon in the Event Data (Text Format) column for that message.



---

## Workflow Agent Ping/Acknowledge

You can test your Business Event System setup using the Workflow Agent Ping/Acknowledge workflow. This workflow sends a ping event message to each inbound agent on the local system or on external systems, and waits to receive an acknowledgement event message from each of the agents. If the workflow completes successfully, then the basic Business Event System setup for communication with these agents is complete.

### How Workflow Agent Ping/Acknowledge Works

---

Use the Launch Processes web page to launch the Workflow Agent Ping/Acknowledge workflow. This workflow consists of two processes, the Master Ping process and the Detail Ping process. To ping all inbound agents, select the Master Ping process, and enter a unique item key. See: Testing Workflow Definitions: page 12 – 2.

When you launch the Master Ping process, the Workflow Engine identifies all the inbound agents that you have defined on the local system or on external systems and launches a Detail Ping process for each agent. The master process then waits for each detail process to complete.

The Detail Ping process begins by sending a Ping Agent event to the inbound agent identified by the master process. The detail process places a Ping Agent event message on a queue associated with an outbound agent on the local system. The event message is addressed to the inbound agent and contains a correlation ID that identifies the detail process to which it belongs. AQ propagation transmits the event message from the outbound queue to the queue associated with the specified inbound agent.

On the receiving system, the listener for the inbound agent dequeues the Ping Agent message the next time it runs. When the event message is dequeued, the Event Manager searches for and executes any active subscriptions to the Ping Agent event or the Any event on that system that have a source type of External.

When the predefined External subscription to the Ping Agent event is executed, its rule function places an Acknowledge Ping event message on a queue associated with an outbound agent on that system. The event message is addressed to an inbound agent on the originating system and includes the correlation ID from the Ping Agent event message. AQ propagation transmits the Acknowledge Ping event message from the outbound queue to the queue associated with the specified inbound agent.

On the originating system, the listener for the inbound agent dequeues the Acknowledge Ping message the next time it runs. When the event message is dequeued, the Event Manager searches for and executes any active subscriptions to the Acknowledge Ping event or the Any event on that system that have a source type of External.

When the predefined External subscription to the Acknowledge Ping event is executed, its rule function, which is the default rule function, sends the event message to the Detail Ping process. The Workflow Engine uses the correlation ID to match the message with the running detail process to which it belongs. After receiving the event message, the Detail Ping process completes.

Finally, after all the detail processes are complete, the master process also completes.

You can use the Workflow Monitor to check the progress of the Workflow Agent Ping/Acknowledge workflow. You can also use the Event System Local Queues page to confirm the processing of the Ping Agent and Acknowledge Ping event messages. See: Workflow Monitor: page 11 – 2 and Reviewing Local Queues: page 13 – 72.

The amount of time needed to complete the Workflow Agent Ping/Acknowledge workflow depends on how often the listeners run to dequeue messages from the inbound agents. See: Scheduling Listeners for Local Inbound Agents: page 13 – 56.

## See Also

Ping Agent Events: page 14 – 8

---

## The Workflow Agent Ping/Acknowledge Item Type

The Workflow Agent Ping/Acknowledge process is associated with an item type called Workflow Agent Ping/Acknowledge. Currently there are two workflow processes associated with Workflow Agent Ping/Acknowledge: Master Ping Process and Detail Ping Process.

To view the details of the Workflow Agent Ping/Acknowledge item type in the Workflow Builder, choose Open from the File menu. Then connect to the database and select the Workflow Agent Ping/Acknowledge item type, or connect to a file called wfping.wft in the `<ORACLE_HOME>\wf\Data\<language>` subdirectory on your file system.

If you examine the property page of Workflow Agent Ping/Acknowledge, you see that it has a persistence type of Temporary and persistence number of days of 0. This means that the runtime data associated with any work items for this item type are eligible for purging as soon as they complete.

The Workflow Agent Ping/Acknowledge item type also has several attributes associated with it. These attributes reference information in the Workflow application tables. The attributes are used and maintained by function activities as well as event activities throughout the process. The following table lists the Workflow Agent Ping/Acknowledge item type attributes.

Display Name	Description	Type	Length/Format/ Lookup Type
To Agent	The inbound agent that receives the event message, in the format <code>&lt;agent&gt;@&lt;system&gt;</code>	Text	
Event Name	The internal name of the event	Text	
Out Agent	The outbound agent that sends the event message, in the format <code>&lt;agent&gt;@&lt;system&gt;</code>	Text	
Event Key	The event key that uniquely identifies the specific instance of the event	Text	
Event Message	The event message	Event	

Table 13 – 8 (Page 1 of 1)

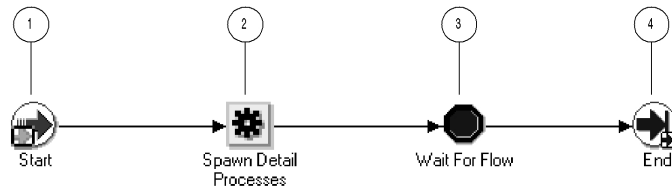
---

## Summary of the Master Ping Process

To view the properties of the Master Ping process, select the process in the navigator tree, and then choose Properties from the Edit menu. This process activity is runnable, indicating that it can be initiated as a top level process to run.

When you display the Process window for the Master Ping process, you see that the process consists of four unique activities. To examine the activities of the process in more detail, we have numbered each

node for easy referencing below. The numbers themselves are not part of the process diagram.



The Workflow Agent Ping/Acknowledge workflow begins when you launch the Master Ping Process using the Launch Processes web page. You can optionally provide a to agent, event name, out agent, event key, and event message. See: Testing Workflow Definitions: page 12 – 2.

The workflow begins at node 1 with the Start activity. At node 2, the master process spawns a detail process for each inbound agent that you have defined on the local system or on external systems. The detail process pings the agent by sending it a Ping Agent event and waits to receive an acknowledgement in the form of an Acknowledge Ping event.

Node 3 is a Wait for Flow activity that waits for all the detail processes to complete. When all the detail processes have completed, the master process ends.

---

## Master Ping Process Activities

Following is a description of each activity in the process, listed by the activity's display name.

### Start (Node 1)

---

This Standard function activity marks the start of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	None

## Spawn Detail Processes (Node 2)

---

This function activity identifies all the inbound agents that you have defined on the local system or external systems, and spawns a Detail Ping process for each agent. The function sets the Ping Agent event (oracle.apps.wf.event.test.ping) as the event to be sent to the Detail Ping processes.

<b>Function</b>	<i>WF_EVENT_PING_PKG.LAUNCH_PROCESSES</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	None
<b>Item Attributes Set by Function</b>	Event Name, To Agent

## Wait for Flow (Node 3)

---

This Standard function activity pauses the flow until the corresponding detail processes complete a specified activity.

<b>Function</b>	<i>WF_STANDARD.WAITFORFLOW</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Spawn Detail Processes

## End (Node 4)

---

This Standard function activity marks the end of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Wait for Flow

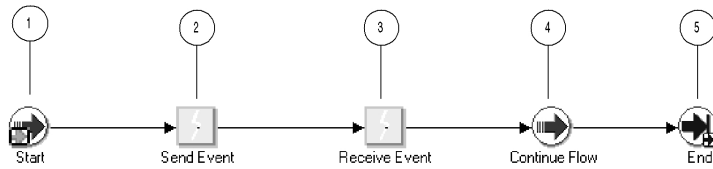
---

## Summary of the Detail Ping Process

To view the properties of the Detail Ping process, select its process activity in the navigator tree, and then choose Properties from the Edit menu. This process activity is runnable, indicating that it can be initiated as a top level process to run.

When you display the Process window for the Detail Ping process, you see that the process consists of five unique activities. To examine the activities of the process in more detail, we have numbered each node

for easy referencing below. The numbers themselves are not part of the process diagram.



The Detail Ping process begins when it is launched by the Master Ping process. See: Summary of the Master Ping Process: page 13 – 79.

The workflow begins at node 1 with the Start activity. At node 2, the process sends a Ping Agent event to the selected inbound agent. At node 3, the process waits to receive an Acknowledge Ping event back from the agent. When the acknowledgement is received, the master process can continue. The detail process ends at this point.

---

## Detail Ping Process Activities

Following is a description of each activity in the process, listed by the activity's display name.

### Start (Node 1)

---

This Standard function activity marks the start of the process.

**Function** *WF\_STANDARD.NOOP*

**Result Type** None

**Prerequisite Activities** None

### Send Event (Node 2)

---

This event activity sends the Ping Agent event (oracle.apps.wf.event.test.ping) from an outbound agent on the local system to the inbound agent identified by the master process. The event message includes a correlation ID that identifies the detail process to which it belongs.

<b>Event Action</b>	Send
<b>Prerequisite Activities</b>	None
<b>Item Attributes Retrieved by Activity</b>	Event Message, Event Name, Event Key, To Agent

### **Receive Event (Node 3)**

---

This event activity receives the Acknowledge Ping event (oracle.apps.wf.event.test.ack) that is returned to the originating system from the system that received the Ping Agent event. The Acknowledge Ping event message contains the correlation ID, which the Workflow Engine uses to match the event message with the detail process to which it belongs.

<b>Event Action</b>	Receive
<b>Event Filter</b>	oracle.apps.wf.event.test.ack
<b>Prerequisite Activities</b>	Send Event
<b>Item Attributes Set by Activity</b>	Event Name, Event Key, Event Message

### **Continue Flow (Node 4)**

---

This Standard function activity marks the position in the detail process where, upon completion, the corresponding halted master process will continue.

<b>Function</b>	<i>WF_STANDARD.CONTINUEFLOW</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Receive Event

### **End (Node 5)**

---

This Standard function activity marks the end of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Continue Flow





CHAPTER

# 14

## Predefined Workflow Events

**T**his chapter tells you how to use Oracle Workflow's predefined events.

---

## Predefined Workflow Events

Oracle Workflow provides several predefined events for significant occurrences within the Business Event System. You can define subscriptions to these events for replication, validation, or other purposes.

All the predefined events are enabled by default. You can disable many of the events if necessary.

Some predefined events are referenced by default subscriptions that are created automatically when you install Oracle Workflow. The subscriber for all the default subscriptions is the local system. You can update, enable, or disable many of these subscriptions to perform the event processing that you want.



**Attention:** You must not change or disable the definition of the Unexpected event or of the predefined Error subscription to that event. If you do, the Event Manager will not be able to perform default error handling for events and subscription processing.

**Note:** Predefined events and default subscriptions are also provided with some Oracle Applications and Oracle Self-Service Web Applications. For more information on Oracle Applications-specific workflow events, consult the documentation or help for that specific Oracle Application product.

---

## Event Definition Events

### Event Created

---

Oracle Workflow raises this event whenever a new individual event or event group definition is created.

<b>Internal Name</b>	oracle.apps.wf.event.event.create
<b>Generate Function</b>	wf_event_functions_pkg.generate

### Event Updated

---

Oracle Workflow raises this event whenever an individual event or event group definition is updated.

<b>Internal Name</b>	oracle.apps.wf.event.event.update
----------------------	-----------------------------------

**Generate Function** wf\_event\_functions\_pkg.generate

#### **Event Deleted**

---

Oracle Workflow raises this event whenever an individual event or event group definition is deleted.

**Internal Name** oracle.apps.wf.event.event.delete

**Generate Function** wf\_event\_functions\_pkg.generate

---

## **Event Group Definition Events**

#### **Event Group Creation**

---

Oracle Workflow raises this event whenever a new event group member definition is created.

**Internal Name** oracle.apps.wf.event.group.create

**Generate Function** wf\_event\_functions\_pkg.generate

#### **Event Group Updated**

---

Oracle Workflow raises this event whenever an event group member definition is updated.

**Internal Name** oracle.apps.wf.event.group.update

**Generate Function** wf\_event\_functions\_pkg.generate

#### **Event Group Deleted**

---

Oracle Workflow raises this event whenever an event group member definition is deleted.

**Internal Name** oracle.apps.wf.event.group.delete

**Generate Function** wf\_event\_functions\_pkg.generate

---

## System Definition Events

### System Created

---

Oracle Workflow raises this event whenever a new system definition is created.

**Internal Name** oracle.apps.wf.event.system.create

**Generate Function** wf\_event\_functions\_pkg.generate

### System Updated

---

Oracle Workflow raises this event whenever a system definition is updated.

**Internal Name** oracle.apps.wf.event.system.update

**Generate Function** wf\_event\_functions\_pkg.generate

### System Deleted

---

Oracle Workflow raises this event whenever a system definition is deleted.

**Internal Name** oracle.apps.wf.event.system.delete

**Generate Function** wf\_event\_functions\_pkg.generate

---

## Agent Definition Events

### Agent Created

---

Oracle Workflow raises this event whenever a new agent definition is created.

**Internal Name** oracle.apps.wf.event.agent.create

**Generate Function** wf\_event\_functions\_pkg.generate

### Agent Updated

---

Oracle Workflow raises this event whenever an agent definition is updated.

**Internal Name** oracle.apps.wf.event.agent.update

**Generate Function** wf\_event\_functions\_pkg.generate

### **Agent Deleted**

---

Oracle Workflow raises this event whenever an agent definition is deleted.

**Internal Name** oracle.apps.wf.event.agent.delete

**Generate Function** wf\_event\_functions\_pkg.generate

---

## **Event Subscription Definition Events**

### **Subscription Created**

---

Oracle Workflow raises this event whenever a new subscription definition is created.

**Internal Name** oracle.apps.wf.event.subscription.create

**Generate Function** wf\_event\_functions\_pkg.generate

### **Subscription Updated**

---

Oracle Workflow raises this event whenever a subscription definition is updated.

**Internal Name** oracle.apps.wf.event.subscription.update

**Generate Function** wf\_event\_functions\_pkg.generate

### **Subscription Deleted**

---

Oracle Workflow raises this event whenever a subscription definition is deleted.

**Internal Name** oracle.apps.wf.event.subscription.delete

**Generate Function** wf\_event\_functions\_pkg.generate

---

## **Synchronize Event Systems Event**

You can raise this event to synchronize the Event Manager data on the local system with another system. The event message for the Synchronize Systems event contains the definitions of all the Event

Manager objects on the local system. See: Synchronizing Systems: page 13 – 70.

<b>Internal Name</b>	oracle.apps.wf.event.all.sync
<b>Generate Function</b>	wf_event_functions_pkg.generate

---

## Seed Event Group

This event group contains events used for automatic replication of Business Event System objects from one system to another. The group includes all the event, event group, system, agent, and subscription definition events, as well as the Synchronize Event Systems event.

<b>Internal Name</b>	oracle.apps.wf.event.group.all
<b>Members</b>	oracle.apps.wf.event.event.create oracle.apps.wf.event.event.update oracle.apps.wf.event.event.delete oracle.apps.wf.event.group.create oracle.apps.wf.event.group.update oracle.apps.wf.event.group.delete oracle.apps.wf.event.system.create oracle.apps.wf.event.system.update oracle.apps.wf.event.system.delete oracle.apps.wf.event.agent.create oracle.apps.wf.event.agent.update oracle.apps.wf.event.agent.delete oracle.apps.wf.event.subscription.create oracle.apps.wf.event.subscription.update oracle.apps.wf.event.subscription.delete oracle.apps.wf.event.all.sync

Oracle Workflow provides two default subscriptions to the Seed Event Group. The first subscription can send the Event Manager data to an agent and to a workflow process when one of the group member events is raised locally. To use this subscription, you must add the agent or workflow to which you want to send the data, and enable the

subscription. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	Local
Event Filter	oracle.apps.wf.event.group.all
Status	Disabled
Rule Data	Message
Rule Function	wf_rule.default_rule
Priority	Normal

**Table 14 – 1 (Page 1 of 1)**

The second subscription can load the Event Manager data into the local system when one of the group member events is received from an external source. To use this subscription, you must enable it. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	oracle.apps.wf.event.group.all
Status	Disabled
Rule Data	Key
Rule Function	wf_event_functions_pkg.receive

**Table 14 – 2 (Page 1 of 1)**

## See Also

To Define an Event Subscription: page 13 – 45

Synchronizing Systems: page 13 – 70

## Ping Agent Events

### Ping Agent Event

---

The Detail Ping process in the Workflow Agent Ping/Acknowledge item type sends this event to ping inbound agents. You can use the Launch Processes web page to launch the Master Ping Process, which in turn launches the Detail Ping process. See: Workflow Agent Ping/Acknowledge: page 13 – 77.

**Internal Name** oracle.apps.wf.event.test.ping

**Generate Function** None

Oracle Workflow provides one default subscription to the Ping Agent event. This subscription sends the Acknowledge Ping event back to the originating system when the Ping Agent event is received from an external source. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	oracle.apps.wf.event.test.ping
Status	Enabled
Rule Data	Key
Rule Function	wf_event_ping_pkg.acknowledge

Table 14 – 3 (Page 1 of 1)

### Acknowledge Ping

---

Oracle Workflow sends this event back to the originating system when a Ping Agent event is received. See: Workflow Agent Ping/Acknowledge: page 13 – 77.

**Internal Name** oracle.apps.wf.event.test.ack

**Generate Function** None

Oracle Workflow provides one default subscription to the Acknowledge Ping event. This subscription sends the Acknowledge



Ping event to the Detail Ping process in the Workflow Agent Ping/Acknowledge item type when the event is received from an external source. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	oracle.apps.wf.event.test.ack
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.default_rule
Workflow Item Type	WFPING
Workflow Process Name	WFDTLPNG

Table 14 – 4 (Page 1 of 1)

## See Also

To Define an Event Subscription: page 13 – 45

---

## System Signup Event

You can raise this event from the System Signup web page on a source system to sign up a destination system for receiving event messages from the source system. See: To Sign Up a System: page 13 – 69.

**Internal Name** oracle.apps.wf.event.system.signup

**Generate Function** None

Oracle Workflow provides one default subscription to the System Signup event. This subscription loads the Event Manager data into the local system when the System Signup event is raised locally. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	Local
Event Filter	oracle.apps.wf.event.system.signup
Status	Enabled
Rule Data	Key
Rule Function	wf_event_functions_pkg.receive

Table 14 – 5 (Page 1 of 1)

## See Also

To Define an Event Subscription: page 13 – 45

---

## Any Event

This event is raised implicitly when any other event is raised locally or received from an external source. You can define a subscription to the Any event to implement processing that you want to execute whenever an event occurs.



**Attention:** You must not change or disable the definition of the Any event. If you do, the Event Manager will not be able to perform error handling for event and subscription processing.

**Internal Name**      oracle.apps.wf.event.any

**Generate Function**      None

Oracle Workflow provides three default subscriptions to the Any event. The first subscription can be triggered when an event is raised locally. To use this subscription, you must define the action for the subscription and enable it. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	Local
Event Filter	oracle.apps.wf.event.any
Phase	100
Status	Disabled
Rule Data	Key
Priority	Normal

**Table 14 – 6 (Page 1 of 1)**

The second subscription can be triggered when an event is received from an external source. To use this subscription, you must define the action for the subscription and enable it. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	oracle.apps.wf.event.any
Phase	100
Status	Disabled
Rule Data	Key
Priority	Normal

**Table 14 – 7 (Page 1 of 1)**

The third subscription sends the event message to the Default Event Error process in the System: Error item type and raises an exception when an event is received from an Error source (that is, when it is dequeued from the WF\_ERROR queue). To use this subscription, you must enable it. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	Error
Event Filter	oracle.apps.wf.event.any
Phase	100
Status	Disabled
Rule Data	Key
Rule Function	wf_rule.error_rule
Workflow Item Type	WF_ERROR
Workflow Process Name	DEFAULT_EVENT_ERROR
Priority	Normal

Table 14 – 8 (Page 1 of 1)

## See Also

To Define an Event Subscription: page 13 – 45

---

## Unexpected Event

Oracle Workflow executes subscriptions to this event when an event is raised locally or received from an external source, but no subscription exists on that event.

**Internal Name** oracle.apps.wf.event.unexpected

**Generate Function** None

Oracle Workflow provides three default subscriptions to the Unexpected event. The first subscription can send the event message to the Default Event Error process in the System: Error item type when an unexpected event is raised locally. To use this subscription, you must enable it.



**Attention:** If you want to enable this subscription, be careful to consider all the events that can be raised on your local system and trigger the subscription. Many local events may be

raised to which you do not want to subscribe. Additionally, if a large number of events are raised on the local system, enabling this subscription may flood the Business Event System.

The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	Local
Event Filter	oracle.apps.wf.event.unexpected
Status	Disabled
Rule Data	Key
Workflow Item Type	WF_ERROR
Workflow Process Name	DEFAULT_EVENT_ERROR

Table 14 – 9 (Page 1 of 1)

The second subscription sends the event message to the Default Event Error process in the System: Error item type when an unexpected event is received from an external source. This subscription allows your local system to handle any event messages received from external systems that you were not expecting.

The Default Event Error process notifies the system administrator, who can retry or abort subscription processing for the event. For example, the system administrator can optionally define a subscription to process the event and then retry the event.

The External subscription to the Unexpected event is enabled by default. You can disable it if necessary.



**Attention:** If you want to disable this subscription, be careful to consider all the consequences for handling unexpected event messages from external sources. If the subscription is disabled, these event messages will remain on the inbound queue where they are received and may be undetected for some time.

The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	oracle.apps.wf.event.unexpected
Status	Enabled
Rule Data	Key
Workflow Item Type	WF_ERROR
Workflow Process Name	DEFAULT_EVENT_ERROR

**Table 14 – 10 (Page 1 of 1)**

The third subscription sends the event message to the Default Event Error process in the System: Error item type when an unexpected event is received from an Error source (that is, when it is dequeued from the WF\_ERROR queue). This subscription is enabled by default.



**Attention:** You must not change or disable the definition of the predefined Error subscription to the Unexpected event. If you disable this subscription, then the Event Manager will not be able to perform error handling for any events for which you have not defined custom Error subscriptions.

The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	Error
Event Filter	oracle.apps.wf.event.unexpected
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.error_rule
Workflow Item Type	WF_ERROR

**Table 14 – 11 (Page 1 of 2)**

Subscription Property	Value
Workflow Process Name	DEFAULT_EVENT_ERROR
Priority	Normal

Table 14 – 11 (Page 2 of 2)

## See Also

To Define an Event Subscription: page 13 – 45

---

## User Entry Has Changed Event

The Workflow LDAP APIs raise this event when changed user information is retrieved from an LDAP directory. You can run these APIs to synchronize the Workflow directory service with Oracle Internet Directory if you implement OID integration. One event is raised for each changed user. See: Synchronizing Workflow Directory Services with Oracle Internet Directory: page 2 – 30 and Workflow LDAP APIs: page 8 – 144.

**Internal Name**      oracle.apps.wf.public.user.change  
**Generate Function**      wf\_entity\_mgr.gen\_xml\_payload

Oracle Workflow provides two default subscriptions to the User Entry Has Changed event. The first subscription loads the changed user data into the WF\_LOCAL\_USERS table for the standalone version of Oracle Workflow when the User Entry Has Changed event is raised locally. You should enable this subscription if you are using the standalone version of Oracle Workflow and you want to implement integration with Oracle Internet Directory. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	Local

Table 14 – 12 (Page 1 of 2)

Subscription Property	Value
Event Filter	oracle.apps.wf.public.user.change
Status	Disabled
Rule Data	Key
Rule Function	wf_sso.user_change

**Table 14 – 12 (Page 2 of 2)**

The second subscription loads the changed user data into the WF\_LOCAL\_USERS table for the version of Oracle Workflow embedded in Oracle Applications when the User Entry Has Changed event is raised locally. You should enable this subscription if you are using the version of Oracle Workflow embedded in Oracle Applications and you want to implement integration with Oracle Internet Directory. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<i>&lt;local system&gt;</i>
Source Type	Local
Event Filter	oracle.apps.wf.public.user.change
Status	Disabled
Rule Data	Key
Rule Function	find_user_pkg.user_change

**Table 14 – 13 (Page 1 of 1)**

## See Also

To Define an Event Subscription: page 13 – 45



---

## Workflow Send Protocol

The Workflow Send Protocol process is a sample workflow process that demonstrates receiving, sending, and acknowledging event messages. Depending on your requirements, you can copy or customize this process to accommodate your organization's specific needs.

The Workflow Send Protocol process receives an event message from a subscription, sends the event message to the inbound agent specified in the subscription, waits to receive an acknowledgement if required, and also sends an acknowledgement if required. You can use the process on one system to send a message to another system, for example, and you can also use the same process on the second system to send the acknowledgement message back to the first system.

The Workflow Send Protocol workflow consists of one process, the Workflow Event Protocol process. This process is launched when it receives an event message from an event subscription.

You can start the Workflow Send Protocol workflow by any of the following methods:

- Raise the Workflow Send Protocol event from the Raise event page. Enter a unique event key, and enter any valid XML document as the event data. A predefined subscription sends the event message to the Workflow Event Protocol process. See: Raising Events: page 13 – 65 and Workflow Send Protocol Events: page 14 – 24.
- The process can also be started when an agent receives the Workflow Send Protocol event from an external source. A predefined subscription sends the event message to the Workflow Event Protocol process. See: Workflow Send Protocol Events: page 14 – 24.
- Define your own subscription to send any event you choose to the Workflow Event Protocol process. In the subscription, specify the workflow item type as WFSNDPRT and the workflow process name as WFEVPRTC. Ensure that you either use the default rule function or include send processing in your custom rule function to send the event to the workflow. The Workflow Event Protocol process starts when the subscription is executed and the process receives the event message. See: To Define an Event Subscription: page 13 – 45.

---

## The Workflow Send Protocol Item Type

The Workflow Send Protocol process is associated with an item type called Workflow Send Protocol. Currently there is one workflow process associated with Workflow Send Protocol: the Workflow Event Protocol process.

To view the details of the Workflow Send Protocol item type in the Workflow Builder, choose Open from the File menu. Then connect to the database and select the Workflow Send Protocol item type, or connect to a file called wfsndprt.wft in the `<ORACLE_HOME>\wf\Data\<language>` subdirectory on your file system.

If you examine the property page of Workflow Send Protocol, you see that it has a persistence type of Temporary and persistence number of days of 0. This means that the runtime data associated with any work items for this item type are eligible for purging as soon as they complete.

The Workflow Send Protocol item type also has several attributes associated with it. These attributes reference information in the Workflow application tables. The attributes are used and maintained by function activities as well as event activities throughout the process. The following table lists the Workflow Send Protocol item type attributes.

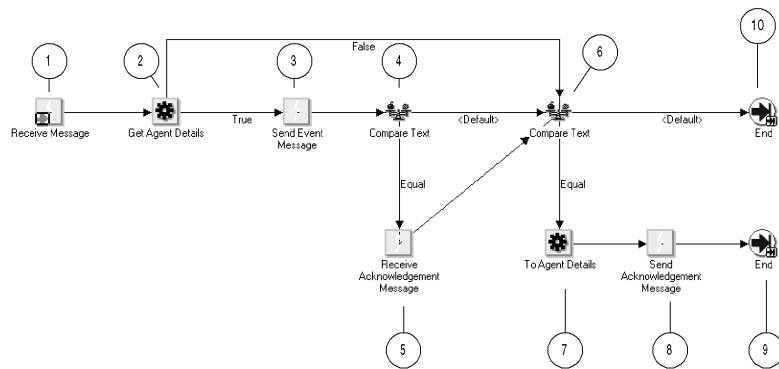
Display Name	Description	Type	Length/Format/ Lookup Type
Event Name	The internal name of the event	Text	
Event Key	The event key that uniquely identifies the specific instance of the event	Text	
Event Message	The event message	Event	
To Agent	The inbound agent that receives the event message, in the format <i>&lt;agent&gt;@&lt;system&gt;</i>	Text	
From Agent	The outbound agent that sends the event message, in the format <i>&lt;agent&gt;@&lt;system&gt;</i>	Text	
Acknowledge Required?	An option that specifies whether the event message that is sent requires an acknowledgement from the recipient	Text	
Send Acknowledgement?	An option that specifies whether to send an acknowledgement of a message that is received	Text	
Acknowledge Message	The acknowledgement message that is sent	Event	
Acknowledge To Agent	The inbound agent that receives the acknowledgement message, in the format <i>&lt;agent&gt;@&lt;system&gt;</i>	Text	
Subscription GUID	The globally unique identifier of the subscription	Text	

Table 14 – 14 (Page 1 of 1)

## Summary of the Workflow Event Protocol Process

To view the properties of the Workflow Event Protocol process, select the process in the navigator tree, and then choose Properties from the Edit menu. This process activity is runnable, indicating that it can be initiated as a top level process to run.

When you display the Process window for the Workflow Event Protocol process, you see that the process consists of eight unique activities, some of which are reused to make up the ten activity nodes that appear in the workflow diagram. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.



The Workflow Send Protocol workflow begins when the Event Manager sends an event message to the Workflow Event Protocol process. For example, when you raise the Workflow Send Protocol event locally or receive that event from an external source, predefined subscriptions send the event message to the Workflow Event Protocol process. See: [Workflow Send Protocol Events: page 14 – 24](#).

The workflow begins at node 1 with the Receive message activity. At node 2, the process attempts to retrieve the agent details for the intended outbound and inbound agents from the subscription.

If no inbound agent is specified, the process continues immediately to node 6 to determine whether to send an acknowledgement message.

If the subscription does specify a To Agent, the process sends the event message to that agent. Then the process determines whether the event message requires an acknowledgement from the recipient, based on a

subscription parameter. If an acknowledgement is required, the Workflow Engine waits to receive the acknowledgement message. Otherwise, the process continues immediately to node 6 to determine whether to send an acknowledgement message.

At node 6, the process determines whether it should send an acknowledgement of the original message that it received. If no acknowledgement needs to be sent, the process ends at this point. Otherwise, the process retrieves the agent details for the inbound agent where the acknowledgement must be sent and sends the acknowledgement message to that agent. Then the process ends.

---

## Workflow Event Protocol Process Activities

Following is a description of each activity in the process, listed by the activity's display name.

### Receive Message (Node 1)

---

This event activity receives the event message that is sent to the Workflow Event Protocol process by the Event Manager to start a new item.

<b>Event Action</b>	Receive
<b>Event Filter</b>	None
<b>Prerequisite Activities</b>	None
<b>Item Attributes Set by Activity</b>	Event Name, Event Key, Event Message

### Get Agent Details (Node 2)

---

This function activity attempts to retrieve the agent details from the subscription for the outbound agent that should send the message and the inbound agent that should receive the message. If no inbound agent is specified, the process continues immediately to node 6. If an inbound agent is specified, but no outbound agent is specified, the function selects a default outbound agent on the local system.

<b>Function</b>	<i>WF_STANDARD.GETAGENTS</i>
<b>Result Type</b>	Boolean

<b>Prerequisite Activities</b>	Receive Message
<b>Item Attributes Retrieved by Function</b>	Subscription GUID
<b>Item Attributes Set by Function</b>	To Agent, From Agent

### **Send Event Message (Node 3)**

---

This event activity sends the event message from an outbound agent on the local system to the specified inbound agent.

<b>Event Action</b>	Send
<b>Prerequisite Activities</b>	Get Agent Details
<b>Item Attributes Retrieved by Activity</b>	Event Message, From Agent, To Agent

### **Compare Text (Node 4)**

---

This Standard function activity compares two text values. At this node, the process checks the Acknowledge Required? item attribute to determine whether the event message sent at node 3 requires an acknowledgement from the recipient.

If the subscription that initiated the process included the parameter name and value pair ACKREQ=Y, the Workflow Engine sets the Acknowledge Required? item attribute to Y when the process is launched. In this case, the process continues from node 4 to node 5. Otherwise, the process continues directly from node 4 to node 6.

<b>Function</b>	<i>WF_STANDARD.COMPARE</i>
<b>Result Type</b>	Comparison
<b>Prerequisite Activities</b>	None
<b>Item Attributes Retrieved by Activity</b>	Acknowledge Required?

### **Receive Acknowledgement Message (Node 5)**

---

This event activity waits to receive the Workflow Send Protocol Acknowledgement event message that is returned to the Workflow Event Protocol process from the system that received the event message sent at node 3.

<b>Event Action</b>	Receive
<b>Event Filter</b>	oracle.apps.wf.event.wf.ack
<b>Prerequisite Activities</b>	Compare Text

### **Compare Text (Node 6)**

---

This Standard function activity compares two text values. At this node, the process checks the Send Acknowledgement? item attribute to determine whether to send an acknowledgement of the original message that it received.

If the original message requires an acknowledgement, the Workflow Engine sets the Send Acknowledgement? item attribute to Y when the process is launched. In this case, the process continues from node 6 to node 7. Otherwise, the process ends at node 10.

<b>Function</b>	<i>WF_STANDARD.COMPARE</i>
<b>Result Type</b>	Comparison
<b>Prerequisite Activities</b>	None
<b>Item Attributes Retrieved by Activity</b>	Send Acknowledgement?

### **To Agent Details (Node 7)**

---

This function activity selects an inbound agent on the originating system where the acknowledgement must be sent and retrieves the agent details for that agent.

<b>Function</b>	<i>WF_STANDARD.GETACKAGENT</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Compare Text
<b>Item Attributes Set by Function</b>	Acknowledge To Agent

### **Send Acknowledgement Message (Node 8)**

---

This event activity sends the Workflow Send Protocol Acknowledgement message from an outbound agent on the local system to the inbound agent identified at node 7.

<b>Event Action</b>	Send
<b>Prerequisite Activities</b>	To Agent Details
<b>Item Attributes Retrieved by Activity</b>	Acknowledge Message, Event Key, Acknowledge To Agent

### **End (Nodes 9 and 10)**

---

This Standard function activity marks the end of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	None

---

## **Workflow Send Protocol Events**

### **Workflow Send Protocol Event**

---

You can raise this event from the Raise Event web page to send the event message to an agent using the Workflow Event Protocol process in the Workflow Send Protocol item type. This workflow process lets you specify whether the message requires an acknowledgement from the recipient, and whether you want to send an acknowledgement of a message that you have received.

<b>Internal Name</b>	oracle.apps.wf.event.wf.send
<b>Generate Function</b>	None

Oracle Workflow provides two default subscriptions to the Workflow Send Protocol event. The first subscription sends the event message to the Workflow Event Protocol process in the Workflow Send Protocol item type when the Workflow Send Protocol event is raised locally. A subscription parameter specifies that the message requires an acknowledgement. This subscription is enabled by default. You can add an outbound agent and inbound agent to the subscription to specify



where you want the Workflow Event Protocol process to send the event message. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	Local
Event Filter	oracle.apps.wf.event.wf.send
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.workflow_protocol
Workflow Item Type	WFSNDPRT
Workflow Process Name	WFEVPRTC
Parameters	ACKREQ=Y

**Table 14 – 15 (Page 1 of 1)**

The second subscription sends the event message to the Workflow Event Protocol process in the Workflow Send Protocol item type when the Workflow Send Protocol event is received from an external source. This subscription is enabled by default. You can optionally add an outbound agent and inbound agent to the subscription to specify that you want the Workflow Event Protocol process to send the event message on to another agent. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	oracle.apps.wf.event.wf.send
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.workflow_protocol

**Table 14 – 16 (Page 1 of 2)**

Subscription Property	Value
Workflow Item Type	WFSNDPRT
Workflow Process Name	WFEVPRTC

**Table 14 – 16 (Page 2 of 2)**

### **Workflow Send Protocol Acknowledgement Event**

Oracle Workflow sends this event back to the originating system when an event message sent to or from the Workflow Event Protocol process requires an acknowledgement.

**Internal Name** oracle.apps.wf.event.wf.ack

**Generate Function** None

Oracle Workflow provides one default subscription to the Workflow Send Protocol Acknowledgement Event. This subscription sends the event message to the Workflow Event Protocol process in the Workflow Send Protocol item type when the Workflow Send Protocol Acknowledgement event is received from an external source. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	oracle.apps.wf.event.wf.ack
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.workflow_protocol
Workflow Item Type	WFSNDPRT
Workflow Process Name	WFEVPRTC

**Table 14 – 17 (Page 1 of 1)**

## See Also

To Define an Event Subscription: page 13 – 45



CHAPTER

# 15

## Demonstration Workflow Processes

**T**his 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 illustrates 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 15 – 5.
- 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 15 – 34.
- Document Review Process—illustrates document management integration (functionality reserved for future use) and looping. See: Document Process: page 15 – 49.
- 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 15 – 54.
- Event System Demonstration Processes—illustrate sending and receiving business events between two systems and using external Java function activities. See: Event System Demonstration: page 15 – 63.



**Attention:** If you are using the standalone version of Oracle Workflow, the Workflow Configuration Assistant installs all of the sample workflow processes listed above.

If you are using the version of Oracle Workflow embedded in Oracle Applications, AutoUpgrade 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
```

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

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.

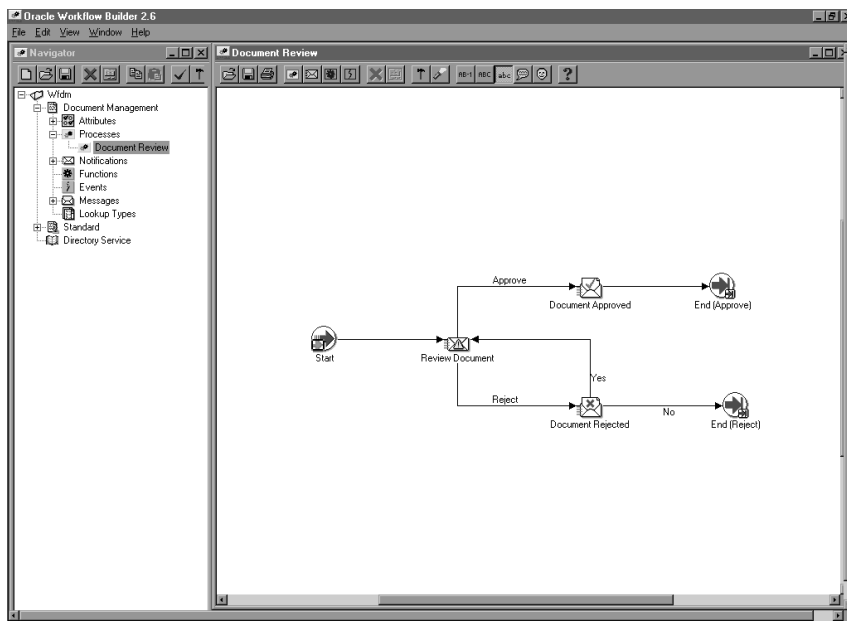
## See Also

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. Connect to your Oracle Workflow database and select the item type you want.

Alternatively, you can connect to any of the sample workflow definitions, 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 is initiated when you create a new requisition to purchase an item. The Requisition process is based on two tables that store approval hierarchy and spending authority information.

When you submit a requisition in this demonstration, the process sends a notification to the next manager in the approval hierarchy to approve the requisition. If the spending limit of the approving manager is less than the requisition amount, the process forwards the requisition to the next higher manager in the approval hierarchy until it finds a manager with the appropriate spending limit to approve the requisition. Each intermediate manager must approve the requisition to move it to the next higher manager. Once a manager with the appropriate spending limit approves the requisition, the process ends with a result of Approve.

The process can end with a result of rejected if:

- Any manager rejects the requisition.
- The requisition amount is greater than the highest spending limit.
- The requisition's requestor does not have a manager.

You can set up and initiate this example process if you are using the standalone version of Oracle Workflow. If you are using Oracle Workflow embedded in Oracle Applications, you should consider this process mainly as an example for explanation purposes and not for demonstration use. The files necessary to set up and run this demonstration are not provided with the version of Oracle Workflow embedded in Oracle Applications.



**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 Workflows Embedded in Oracle E-Business Suite: 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.

This sample workflow is based on the demonstration data model. The data model includes two tables with data: one table maintains an

employee approval hierarchy and the other maintains the spending limit of each employee. These two tables make up the database application that we use to approve a requisition. In addition, the data model also includes a directory service that identifies the Oracle Workflow users and roles in this sample implementation.

There are two ways you can initiate the Requisition process based on a fictitious requisition: run a script or submit a requisition using a web-based interface. Both methods require that you provide the name of the employee who prepared the requisition, the requisition amount, the requisition number, a requisition description, a requisition process owner and the name of the workflow process to initiate.

This section describes the Requisition process in detail to give you an understanding of what each activity in this workflow accomplishes.

---

## Installing the Requisition Data Model

The Requisition data model is installed only with the standalone version of Oracle Workflow. The data model is automatically installed for you by the Workflow Configuration Assistant. The files used in the installation are copied to the *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 the installation. See: Overview of Setting Up: page 2 – 6.

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.



**Attention:** For security reasons, the installation process automatically locks these user accounts after they are created. Before you can begin using the accounts, you must unlock them using a script called *wfdemoul.sql*. This script is located in the *wf/demo* subdirectory within your Oracle Home. Connect to the SYSTEM database account using SQL\*Plus and run the script using the following command:

```
sqlplus SYSTEM/<SYSTEM pwd> @wfdemoul
```

See your Oracle DBA if you need more information about the SYSTEM account and password.


- Calls a script called *wfdemoc.sql* to create two tables with seed data. These tables make up the demonstration database application that is workflow-enabled:
  - WF\_REQDEMO\_EMP\_HIERARCHY—maintains the employee approval hierarchy. The approval chain consists of these employee user IDs listed in ascending order with the employee having the most authority listed last: BLEWIS, KWALKER, CDOUGLAS, and SPIERSON.
  - WF\_REQDEMO\_EMP\_AUTHORITY—maintains the spending limit for each employee. The limit for each employee follows the employee’s user ID: BLEWIS:500, KWALKER:1000, CDOUGLAS:2000, and SPIERSON:3000.
- The script *wfdemoc.sql* also inserts seed data into the WF\_LOCAL\_USERS, WF\_LOCAL\_ROLES, WF\_LOCAL\_USER\_ROLES tables. The following table shows the users and roles that are seeded by the script.


User	ADMIN Role	MANAGERS Role	WORKERS Role	OTHERS Role
SYSADMIN	yes			
WFADMIN	yes			
BLEWIS			yes	
KWALKER			yes	
CDOUGLAS			yes	yes
SPIERSON		yes		yes

Table 15 – 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 – 58.

 **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 – 21.

- Calls the scripts *wfdemos.sql* and *wfdemob.sql* to create the PL/SQL spec and body for packages called WF\_REQDEMO and WF\_DEMO. These packages contain:
  - The PL/SQL stored procedures associated with the demonstration home page.
  - The PL/SQL stored procedures called by the function activities used in the Requisition Process workflow.
  - The PL/SQL procedure WF\_REQDEMO.Create\_Req called by the Oracle Workflow web agent to generate the web-based interface page for the Requisition process demonstration.
- Runs the Workflow Resource Generator to load messages from *wfdemo.msg* into the database. The messages are used by the web-based interface page for the Requisition process demonstration.
- Loads the Requisition Process workflow definition from *wfdemo.wft* into the database. You can view this process in Oracle Workflow Builder.

---

## Initiating the Requisition Workflow

You can use any of the following methods to initiate the Requisition workflow:

- Run the script *wfrund.sql*.

- 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 15 – 23.

► **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 – 12.

You can also access the Workflow Monitor to view the status of the workflow process. See: Using the Find Processes Web Page: page 11 – 9.

## ► 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 – 14.

Alternatively, you can enter the following URL to directly display the Requisition Approval web page:

```
<webagent>/wf_reqdemo.create_req
```

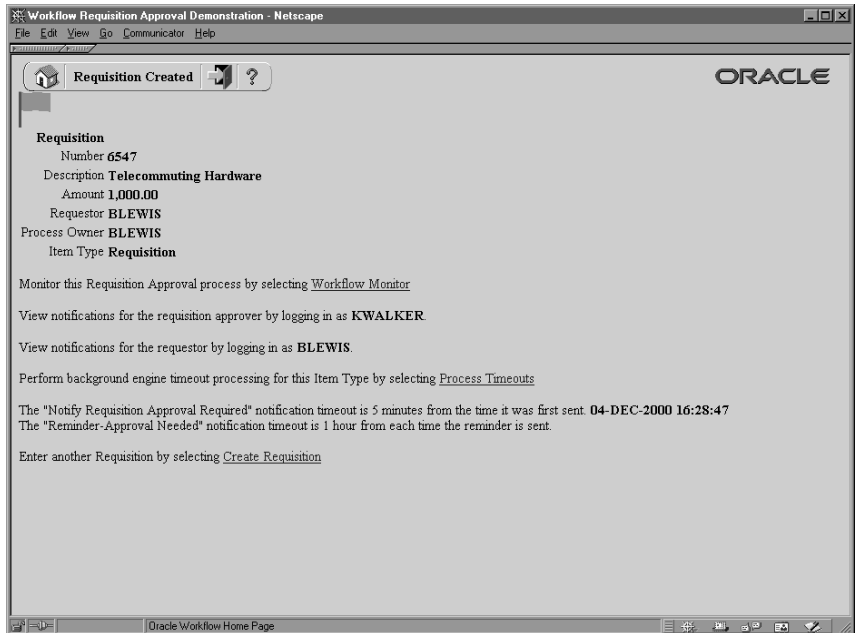


**Attention:** These are both secured pages, so if you have not yet logged on as a valid workflow user in the current web session, you will be prompted to do so before the page appears.

Employee	Spending Limit	Manager
BLEWIS	500	KWALKER
KWALKER	1000	CDOUGLAS
CDOUGLAS	2000	SPIERSON
SPIERSON	3000	

**Description**  
When you submit a requisition in this demonstration, the process sends a notification to the next manager in the approval hierarchy to approve the requisition. If the spending limit of the approving manager is less than the requisition amount, the process forwards the requisition to the next higher manager in the approval hierarchy until it finds a manager with the appropriate spending limit to approve the requisition. Each intermediate manager must approve the requisition to move it to the next higher manager. Once a manager with the appropriate spending limit approves the requisition, the process ends with a result of approved. The process can end with a result of rejected if any manager rejects the requisition, the requisition amount is greater than the highest spending limit, or the requisition's requestor does not have a manager.

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.

9. Select the Process Timeouts HTML link to have the background engine look for any timed out notifications and execute the next activity expected to run in the case of a time out.

Two messages appear below this link informing you of when a timeout may occur in the process.

10. Select the Create Requisition HTML link if you wish to enter and submit another requisition in the Requisition Demonstration web page.

---

## The Requisition Item Type

The Requisition process is associated with an item type called Requisition. Currently there are two workflow processes associated with Requisition: Requisition Approval and Notify Approver.

If you examine the property page of Requisition, you see that it has a persistence type of Temporary and persistence number of days of 0. This means that the run time data associated with any work items for this item type are eligible for purging as soon as they complete. You also see that it calls a selector function named *WF\_REQDEMO.SELECTOR*. This selector function is an example PL/SQL stored procedure that returns the name of the process to run when more than one process exists for a given item type. The selector function in this example returns *REQUISITION\_APPROVAL* or 'Requisition Approval' as the process to run.

The Requisition item type also has several attributes associated with it. These attributes reference information in the demonstration application tables. The attributes are used and maintained by function activities as well as notification activities throughout the process. The following table lists the Requisition item type attributes.



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 15 – 2 (Page 1 of 1)

---

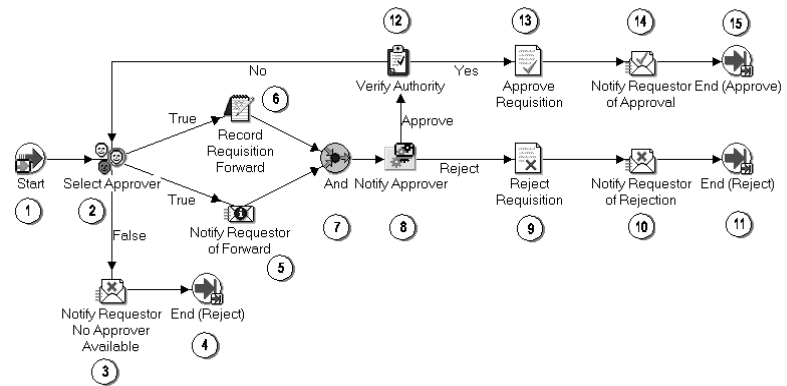
## Summary of the Requisition Approval Process

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.

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 – 26.

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 15 – 8.

The workflow begins at node 1 with the Start activity.

At node 2, the process attempts to select an approver for the requisition. If an approver cannot be found for the requisition, the requestor is notified and the process ends with the final process result of Reject. If an approver is found, then the requestor is notified of who that approver is and a function records in the application that the requisition is being forwarded to the approver. Both of these activities must complete before the approver is actually notified in node 8.

Node 8 is a subprocess that requests the approver to approve the requisition by a specified period of time and if the approver does not respond by that time, the subprocess performs a timeout activity to keep sending a reminder to the approver until the approver responds. If the approver rejects the requisition, the requisition gets updated as rejected in node 9, and the requestor is notified in node 10. The process ends at this point with a result of Reject.

If the approver approves the requisition, the process transitions to node 12 to verify that the requisition amount is within the approver's spending limit. If it is, the process approves the requisition in node 13, and notifies the requestor in node 14. The process ends in this case with a result of Approve.

---

## Requisition Process Activities

Following is a description of each activity listed by the activity's display name. You can create all the components for an activity in the graphical Oracle Workflow Builder except for the PL/SQL stored procedures that the function activities call. Function activities can execute functions external to the database by integration with Oracle Advanced 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 15 – 26 and Example Notification Activities: page 15 – 31.

### Start (Node 1)

---

This is a Standard function activity that simply marks the start of the process.

<b>Function</b>	<code>WF_STANDARD.NOOP</code>
<b>Result Type</b>	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

<b>Result Type</b>	None
<b>Prerequisite Activities</b>	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.

<b>Function</b>	<i>WF_REQDEMO.Forward_Req</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	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.

<b>Function</b>	<i>WF_STANDARD.ANDJOIN</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	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 15 – 19.

<b>Result Type</b>	Approval
<b>Prerequisite Activities</b>	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.

<b>Function</b>	<i>WF_REQDEMO.Reject_Req</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	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.

<b>Message</b>	Requisition Rejected
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	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.

<b>Function</b>	<i>WF_REQDEMO.VerifyAuthority</i>
<b>Result Type</b>	Yes/No
<b>Prerequisite Activities</b>	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.

<b>Function</b>	<i>WF_REQDEMO.Approve_Req</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	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.

<b>Message</b>	Requisition Approved
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	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.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Start

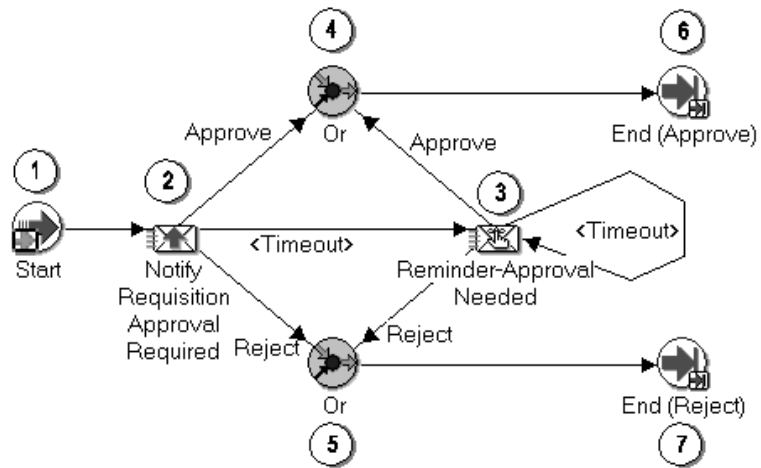
---

## **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** None

### Activities

### Notify Requisition Approval Required (Node 2)

---

This activity notifies the approver that the requisition needs to be approved or rejected. This activity must be completed within 5 minutes, otherwise it times out.

The message includes 'Send' attributes that display the requisition number, requisition description, requisition amount, previous approver name, and preparer name for the requisition when the notification is sent.

The message includes 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.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	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/'
                                           ||Itemtype||':'||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 – 19.

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 – 21.

**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 – 23.

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 – 48.

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 – 26.

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 – 28.

---

## 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 – 3.

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 – 57.

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 – 57.

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` with `itemtype` and `itemkey` concatenated as an argument. The function parses this string and creates 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 initiate this example process if you are using the standalone version of Oracle Workflow. If you are using Oracle Workflow embedded in Oracle Applications, you should consider this process mainly as an example for explanation purposes and not for demonstration use. The files necessary to set up and run this demonstration are not provided with the version of Oracle Workflow embedded in Oracle Applications.



**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 Workflows Embedded in Oracle E-Business Suite*: 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 installed only with the standalone version of Oracle Workflow. The data model is automatically installed for you by the Workflow Configuration Assistant. The files used in the installation are copied to the *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 the installation. See: Overview of Setting Up: page 2 – 6.

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.
- 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.
- 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 15 – 7.



**Attention:** For security reasons, the installation process automatically locks the database accounts for the users in the demonstration directory service after they are created. Before you can begin using the accounts, you must unlock them using a script called *wfdemoul.sql*. See: Installing the Requisition Data Model: page 15 – 7.

---

## 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 15 – 36.
- 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 – 14.

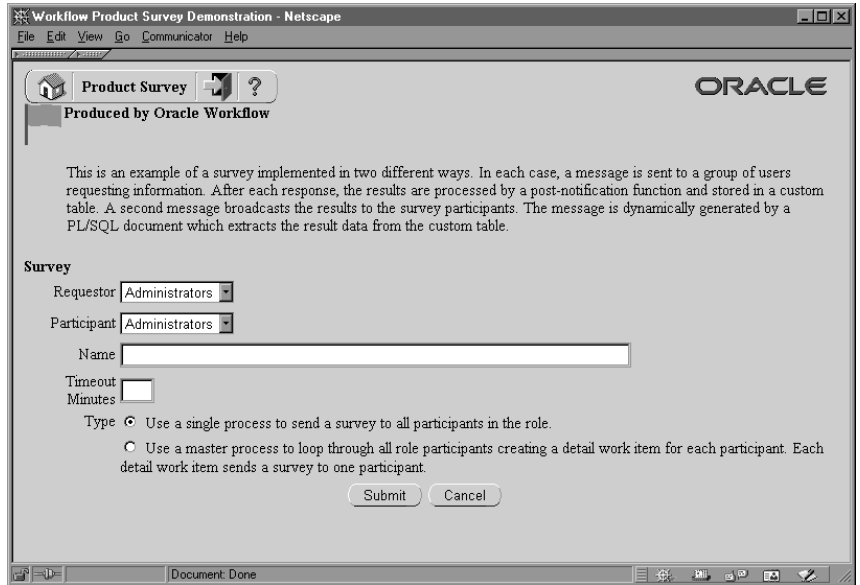
Alternatively, you can enter the following URL to directly display the Product Survey web page:

```
<webagent>/wf_surveydemo.create_survey
```

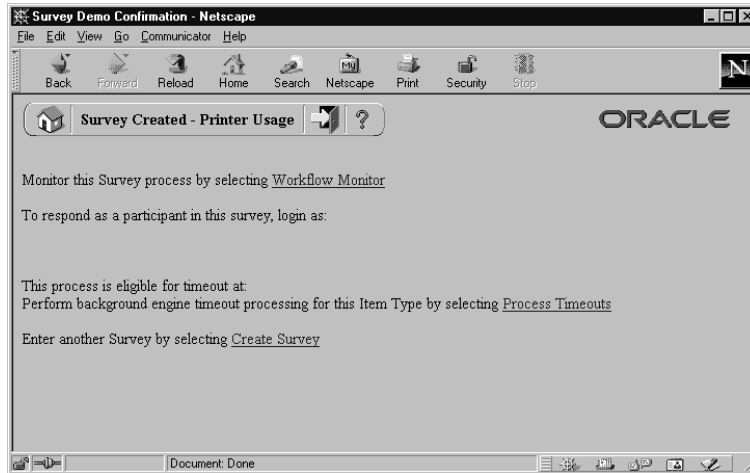


**Attention:** These are both secured pages, so if you have not yet logged on as a valid workflow user in the current web session, you will be prompted to do so before the page appears.





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. The following table lists the 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 15 – 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.

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.

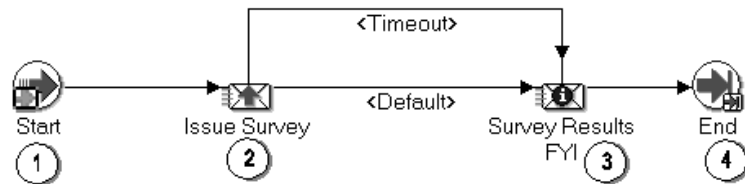
The Survey – Single Process workflow begins when you submit a survey using the Product Survey web page accessible from the Workflow Demonstrations home page. You must provide a survey requestor role, survey participant role, survey name, a timeout value in minutes, and check "Use a single process to 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.



---

## Survey – Single 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 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.

The Details property page of the process activity indicates that Survey – Master/Detail Process has an associated error item type and error process called WFEERROR and DEFAULT\_ERROR, respectively. The DEFAULT\_ERROR process of item type WFEERROR 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."

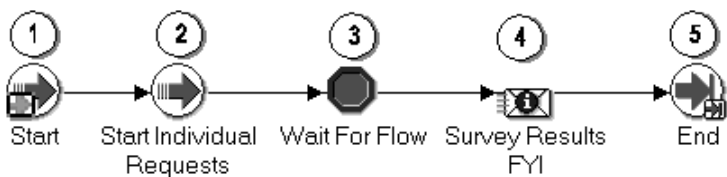
**Note:** If you choose to initiate the survey process from the Launch Processes web page, you should select "Survey – Master/Detail Process" as the process name.

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.



---

## Survey – Master/Detail 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

### **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, but instead it must be called by a higher level process.

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.

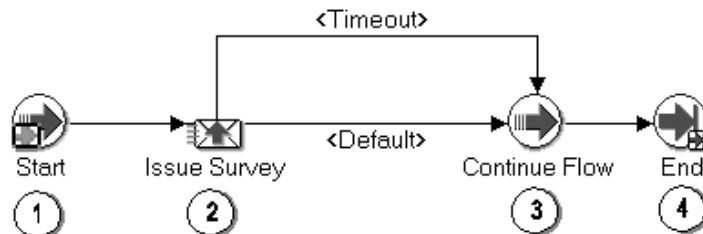
When you display the Process window for Detail Survey Process, you see that the process consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

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



**Attention:** Document management functionality is reserved for future use. This description of the Document Review Process is provided for reference only.

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.

The process definition of the Document Review Process is automatically installed for you by the Workflow Configuration Assistant for the standalone version of Oracle Workflow or by AutoUpgrade for the version of Oracle Workflow embedded in Oracle Applications.

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 15 – 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

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. The following table lists the 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 15 – 4 (Page 1 of 1)

---

## Summary of the Document Review Process

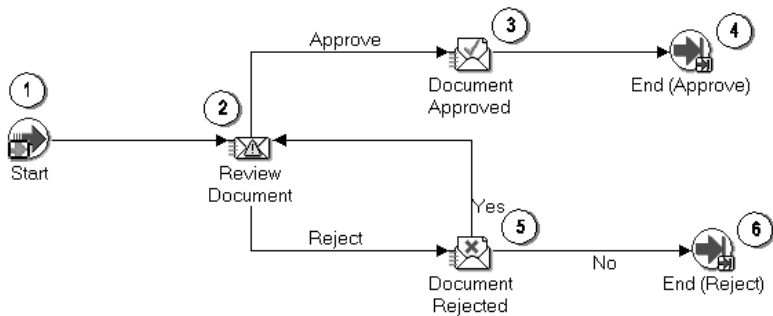
To view the properties of the Document Review process, select the process in the navigator tree, then choose Properties from the Edit menu. The Document Review process has a result type of Approval, indicating that when the process completes, it has a result of Approve or Reject (the lookup codes in 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.

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.

The process definition of the Error Check Process is automatically installed for you by the Workflow Configuration Assistant for the standalone version of Oracle Workflow or by AutoUpgrade for the version of Oracle Workflow embedded in Oracle Applications.

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 15 – 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. The following table lists the 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 15 – 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.

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.

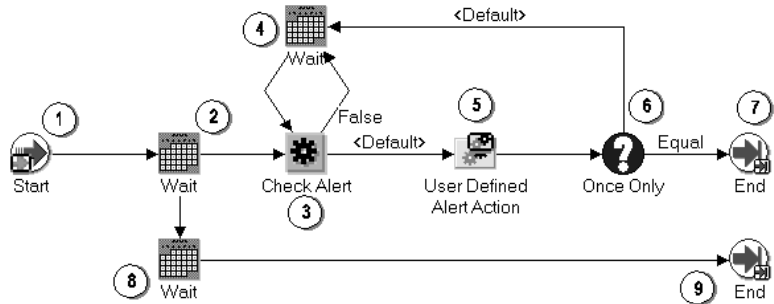
The Error Check workflow begins when you initiate the process from the Launch Processes web-based interface. You must provide the Item Key, User Key, Process Owner, Alert Recipient, Start date of the process, End date of the process, and the Frequency (day of week, day of month, time of day, days, or 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, but instead it must be called by a higher level process.

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

---

## Event System Demonstration

The Event System Demonstration is an example of using events to transmit business documents between two systems. You initiate the demonstration process by entering a purchase order on one system. Oracle Workflow generates a purchase order XML document and sends this document to a second system. The second system processes the purchase order and sends back to the first system three XML documents representing a purchase order acknowledgement, an advanced shipment notice, and an invoice.

You can initiate this example process if you are using the standalone version of Oracle Workflow. If you are using Oracle Workflow embedded in Oracle Applications, you should consider this process mainly as an example for explanation purposes and not for demonstration use. The files necessary to set up and run this demonstration are not provided with the version of Oracle Workflow embedded in Oracle Applications.



**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 Workflows Embedded in Oracle E-Business Suite: page B - 2.

Before running the Event System Demonstration, you must set up a Buyer system and a Supplier system to use in the demonstration.

**Note:** You can either set up two separate systems, or you can use the same system as both the Buyer and the Supplier.

Then you can initiate an Event System Demonstration process on the Buyer system from the Workflow Demonstrations web page. When you initiate the process, you must specify an order number, item number, item description, deliver date, total amount, and order requestor role for the purchase order.

When you submit the purchase order, the order information is inserted into a database table and a purchase order event is raised with the order number as the event key. Raising the purchase order event triggers two subscriptions to this event that have the source type Local. The first subscription adds a correlation ID to the event message. The correlation ID consists of the prefix PO followed by the event key (the order number).

Because the second subscription to this event requires the complete event data, the Event Manager runs the Generate function for the event to produce a valid purchase order XML document. The second

subscription sends the event to the Buyer: Top Level PO process in the Event System Demonstration item type. The Workflow Engine creates a new instance of this process with the correlation ID as the item key.

During the purchase order processing in the Buyer: Top Level PO process, a standard external Java function activity retrieves the name of the order requestor from the purchase order XML document, so that the process can send notifications to the requestor. The purchase order event message is sent to the Supplier system, and then the process waits to receive responses from the supplier.

When the purchase order event arrives at the Supplier system, two subscriptions to this event with the source type External are triggered. The first subscription changes the correlation ID in the event message to consist of the prefix *SO* followed by the event key (the order number). The second subscription sends the event to the Supplier: Top Level Order process in the Event System Demonstration item type. The Workflow Engine creates a new instance of this process with the new correlation ID as the item key.

During the purchase order processing in the Supplier: Top Level Order process, standard external Java function activities retrieve the item number and item description from the purchase order XML document. The process sends event messages containing the following XML documents back to the Buyer system:

- purchase order acknowledgement
- advanced shipment notice
- invoice

On the Buyer system, these events trigger subscriptions with the source type External. For each event, there are two subscriptions: one that adds a correlation ID consisting of the prefix *PO* followed by the event key (the order number), and another that sends the event message to the Buyer: Top Level PO process, using the correlation ID to match the message with the running process to which it belongs. The process receives the event messages and notifies the order requestor when each one arrives. After all three response documents have been received, the process completes.

---

## Installing the Event System Demonstration Data Model

The Event System data model is installed only with the standalone version of Oracle Workflow. The data model is automatically installed for you by the Workflow Configuration Assistant. The files used in the

installation are copied to the *demo* and *demo/<language>* subdirectories of your Oracle Workflow server directory structure.



**Attention:** For the Event System Demonstration to work properly, you should perform the steps required to set up Oracle Workflow after the installation. See: Overview of Setting Up: page 2 – 6.

The installation does the following:

- Calls a script called *wfevdemc.sql* to create two tables called WF\_EVENTDEMO\_ITEMS and WF\_EVENTDEMO\_PO. The table WF\_EVENTDEMO\_ITEMS contains the items that can be selected for the purchase order. The Event System Demonstration process updates the table WF\_EVENTDEMO\_PO with information from the purchase order.
- Calls the scripts *wfevdems.sql* and *wfevdemb.sql* to create the PL/SQL spec and body for a package called WF\_EVENTDEMO. This package contains:
  - The PL/SQL stored procedures called by the function activities used in the Event System Demonstration workflow.
  - The PL/SQL procedures WF\_EVENTDEMO.Create\_Order and WF\_EVENTDEMO.Track\_Order called by the Oracle Workflow web agent to generate the web-based interface pages for the Event System Demonstration process demonstration.
- Loads the Event System Demonstration workflow definition from *wfevdeme.wft* into the database. You can view this process in the Oracle Workflow Builder.
- The data model for the Event System Demonstration process also includes the same demonstration directory service that is used by the sample Requisition process. See: Installing the Requisition Data Model: page 15 – 7.



**Attention:** For security reasons, the installation process automatically locks the database accounts for the users in the demonstration directory service after they are created. Before you can begin using the accounts, you must unlock them using a script called *wfdemoul.sql*. See: Installing the Requisition Data Model: page 15 – 7.

---

## Initiating the Event System Demonstration Workflow

The Event System Demonstration requires two Workflow-enabled systems, a Buyer system and a Supplier system. You can either set up two separate systems, or you can use the same system as both the Buyer and the Supplier. Before you can run the Event System Demonstration, you must set up the system or systems that you want to use. See: *To Set Up the Event System Demonstration Workflow*: page 15 – 66.

After the systems are set up, you can initiate the Event System Demonstration workflow using the *Events: Buyer Workbench* demonstration web page on the Buyer system. Then use the *Events: Track Order* demonstration web page on the Supplier system to continue processing the Event System Demonstration workflow. See: *To Initiate the Event System Demonstration Workflow from the Buyer Workbench*: page 15 – 67 and *To Continue the Event System Demonstration Workflow on the Supplier System*: page 15 – 69.

### ► To Set Up the Event System Demonstration Workflow

1. If you are using two separate installations of Oracle Workflow, designate one of the installations as the Buyer system and the other as the Supplier system.
2. If you are using two separate installations of Oracle Workflow, sign the two systems up with each other to exchange event messages. See: *Signing Up Systems*: page 13 – 67.
3. If you are using two separate installations of Oracle Workflow, on the Buyer system, locate the predefined subscription to the `demo.oracle.apps.wf.po.create` event with the source type `Local` and the phase 2. Edit this subscription by selecting the standard `WF_IN` agent on the Supplier system as the To agent. See: *To Define an Event Subscription*: page 13 – 45.
4. If you are using two separate installations of Oracle Workflow, on the Supplier system, locate the predefined subscription to the `demo.oracle.apps.wf.po.create` event with the source type `External` and the phase 2. Edit this subscription by selecting the standard `WF_IN` agent on the Buyer system as the To agent. See: *To Define an Event Subscription*: page 13 – 45.
5. Ensure that the Java Function Activity Agent is running on your systems. See: *Setting Up the Java Function Activity Agent*: page 2 – 86.

6. Optionally schedule a background engine to run every 10 to 30 seconds on your systems. See: Setting Up Background Engines: page 2 – 43.

You can also run the background engine manually during the demonstration by choosing the Process Order link on the Events: Track Order page.

► **To Initiate the Event System Demonstration Workflow from the Buyer Workbench**

1. On the Buyer system, enter the following URL in a web browser to access the Workflow Demonstration web page:

```
<webagent>/wf_demo.home
```

*<webagent>* represents the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.

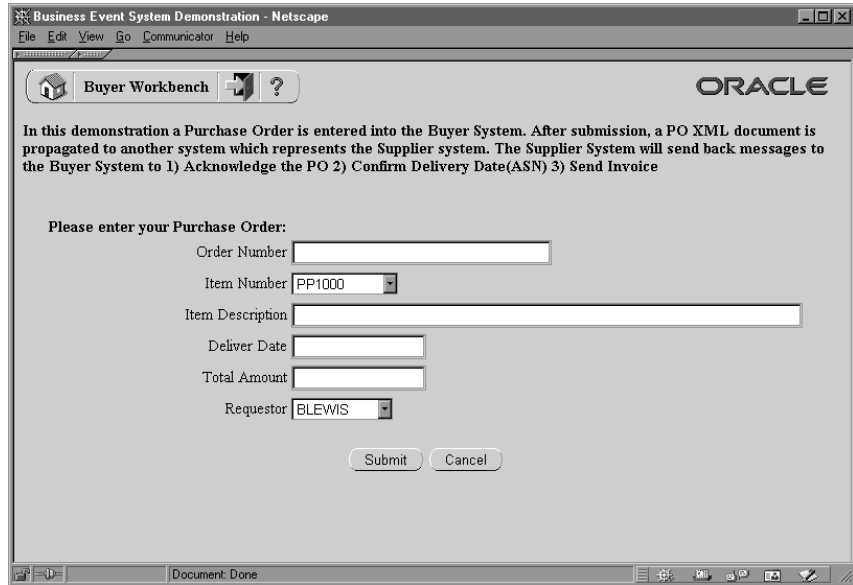
Choose the Events: Buyer Workbench link to display the Buyer Workbench web page.

Alternatively, you can enter the following URL to directly display the Buyer Workbench web page:

```
<webagent>/wf_eventdemo.create_order
```

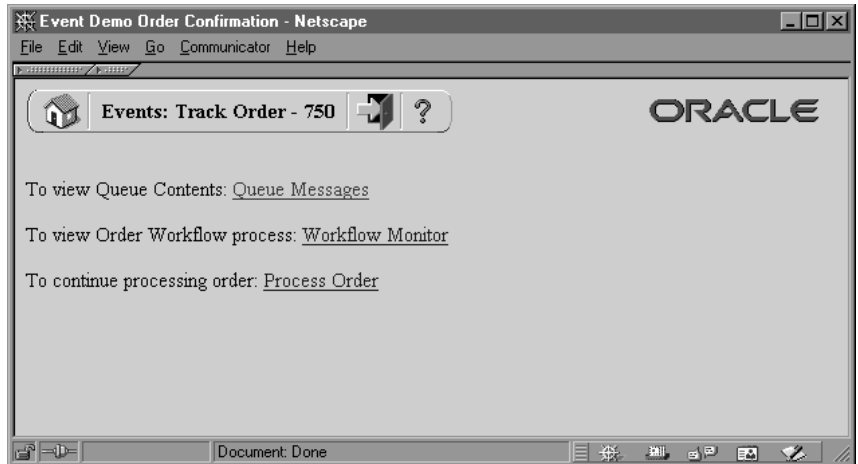


**Attention:** These are both secured pages, so if you have not yet logged on as a valid workflow user in the current web session, you will be prompted to do so before the page appears.



2. Enter a unique order number for the purchase order. The order number becomes the event key for the B2B Purchase Order event that is raised.
3. Select an item number.
4. Enter an item description of 40 characters or less.
5. Enter a deliver date in the format DD-MON-YYYY.
6. Enter the total amount for the purchase order. The amount should be a number without formatting.
7. Select a role name as the order requestor.
8. Choose the Submit button to submit the purchase order and display the Events: Track Order page. You can also choose the Cancel button to return to the Workflow Demonstrations page without submitting the purchase order.





9. In the Events: Track Order page, choose the Queue Messages link to display the Event System Local Queues page and review the messages on the Business Event System queues. See: Reviewing Local Queues: page 13 – 72.
10. Choose the Workflow Monitor link to view the progress of the Buyer: Top Level PO workflow process in the Workflow Monitor. See: Workflow Monitor: page 11 – 2.
11. Choose the Process Order link to run the background engine and process the deferred activity so that the buyer workflow can continue processing the purchase order.

► **To Continue the Event System Demonstration Workflow on the Supplier System**

1. On the Supplier system, enter the following URL in a web browser to access the Workflow Demonstration web page:

`<webagent>/wf_demo.home`

`<webagent>` represents the base URL of the web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences: page 2 – 14.

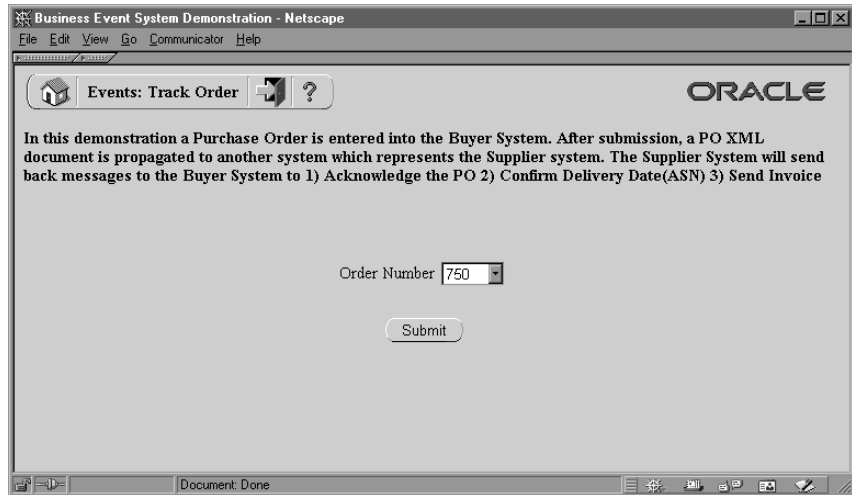
Choose the Events: Track Order link to display the Events: Track Order web page.

Alternatively, you can enter the following URL to directly display the Buyer Workbench web page:

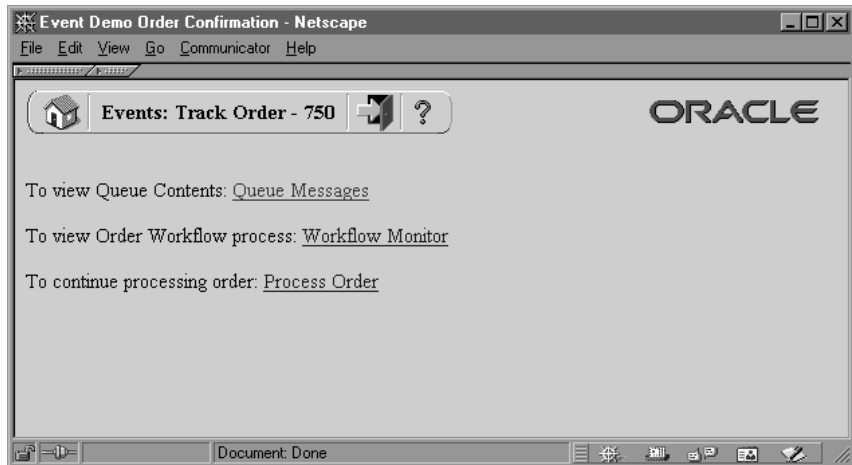
`<webagent>/wf_eventdemo.track_order`



**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 the order number for your purchase order and choose the Submit button.



3. In the Events: Track Order page that appears, choose the Queue Messages link to display the Event System Local Queues page and review the messages on the Business Event System queues. See: Reviewing Local Queues: page 13 – 72.

4. Choose the Workflow Monitor link to view the progress of the Supplier: Top Level Order process in the Workflow Monitor. See: Workflow Monitor: page 11 – 2.
5. Choose the Process Order link to run the background engine and continue processing the purchase order. Perform this step twice to process both deferred activities in the supplier workflow.

---

## The Event System Demonstration Item Type

The Event System demonstration is associated with an item type called Event System Demonstration. Currently there are eleven workflow processes associated with Event System Demonstration:

- Buyer: Top Level PO
- Buyer: Send PO to Supplier
- Buyer: Receive Supplier PO Acknowledgement
- Buyer: Advanced Shipment Notice
- Buyer: Receive Supplier Invoicing
- Supplier: Top Level Order
- Supplier: Get Order Details
- Supplier: Credit Check
- Supplier: Stock Check
- Supplier: Advanced Shipment Notice
- Supplier: Send Supplier Invoice

To view the details of the Event System Demonstration item type in the Workflow Builder, choose Open from the File menu. Then connect to the database and select the Event System Demonstration item type, or connect to a file called wfefdeme.wft in the `<ORACLE_HOME>\wf\Data\<language>` subdirectory on your file system.

If you examine the property page of Event System Demonstration, you see that it has a persistence type of Temporary and persistence number of days of 0. This means that the runtime data associated with any work items for this item type are eligible for purging as soon as they complete.

The Event System Demonstration item type also has several attributes associated with it. These attributes reference information in the Workflow application tables. The attributes are used and maintained

by function, notification, and event activities throughout the process. The following table lists the Event System Demonstration item type attributes.

Display Name	Description	Type	Length/Format/ Lookup Type/ Default Value
Event Name	The internal name of the original event	Text	
Event Key	The event key that uniquely identifies the specific instance of the original event	Text	
Event Message	The event message for the original event	Event	
Purchase Order Status	The purchase order status	Text	CREATED
PO Acknowledge Event	The name of the purchase order acknowledgement event	Text	
PO Acknowledge Event Key	The event key that uniquely identifies the specific instance of the purchase order acknowledgement event	Text	
PO ASN Event	The name of the advanced shipment notice event	Text	
PO ASN Event Key	The event key that uniquely identifies the specific instance of the advanced shipment notice event	Text	
PO ASN Event Message	The advanced shipment notice event message	Event	
Order Requestor	The name of the person who requested the order	Text	BLEWIS
To Agent/System 1	The inbound agent on the Buyer system that receives the event message, in the format <i>&lt;agent&gt;@&lt;system&gt;</i>	Text	

**Table 15 – 6 (Page 1 of 2)**

Display Name	Description	Type	Length/Format/ Lookup Type/ Default Value
From Agent/System 1	The outbound agent on the Buyer system that sends the event message, in the format <code>&lt;agent&gt;@&lt;system&gt;</code>	Text	
To Agent/System 2	The inbound agent on the Supplier system that receives the event message, in the format <code>&lt;agent&gt;@&lt;system&gt;</code>	Text	
From Agent/System 2	The outbound agent on the Supplier system that sends the event message, in the format <code>&lt;agent&gt;@&lt;system&gt;</code>	Text	
Item Number	The item number	Text	
Item Description	The item description	Text	
Subscription GUID	The globally unique identifier of the subscription	Text	

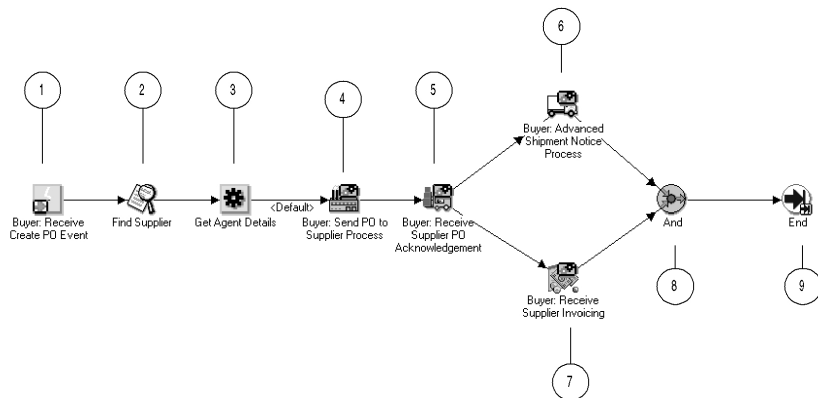
Table 15 – 6 (Page 2 of 2)

---

## Summary of the Buyer: Top Level PO Process

To view the properties of the Buyer: Top Level PO process, select the process in the navigator tree, then choose Properties from the Edit menu. This process activity is runnable, indicating that it can be initiated as a top level process to run.

When you display the Process window for the Buyer: Top Level PO process, you see that the process consists of nine unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.



The Buyer: Top Level PO workflow begins when you submit a purchase order from the Buyer Workbench demonstration page, raising the B2B Purchase Order event. Predefined subscriptions add a correlation ID consisting of the prefix PO followed by the event key (the order number), and send the event to the Buyer: Top Level PO process. See: To Initiate the Event System Demonstration Workflow from the Buyer Workbench: page 15 – 67.

The process begins at node 1 with the Buyer: Receive Create PO Event activity. The process finds a supplier for the purchase order and retrieves the agent details for the intended outbound and inbound agents to send the purchase order to the supplier.

Node 4 is a subprocess that retrieves the name of the order requestor from the purchase order event message, sends the purchase order to the supplier, and notifies the order requestor that the purchase order has been sent to the supplier.

Node 5 is a subprocess that waits to receive a purchase order acknowledgement event message from the supplier. If the acknowledgement is not received within a specified period of time, the subprocess performs a timeout activity to keep notifying the order requestor that the supplier has not responded until the acknowledgement is received. When the acknowledgement is received, the subprocess notifies the order requestor of the acknowledgement.

After the purchase order acknowledgement is received, the top level process waits to receive an advanced shipment notice and an invoice from the supplier. Node 6 is a subprocess that receives the advanced shipment notice and notifies the order requestor that the order has been shipped. Node 7 is a subprocess that receives the invoice and notifies

the order requestor that the invoice has arrived. After both of these subprocesses complete, the process ends.

---

## Buyer: Top Level PO Process Activities

Following is a description of each activity in the process, listed by the activity's display name.

### Buyer: Receive Create PO Event (Node 1)

---

This event activity receives the B2B Purchase Order event message that is sent to the Buyer: Top Level PO process by the Event Manager to start a new item.

<b>Event Action</b>	Receive
<b>Event Filter</b>	demo.oracle.wf.b2b.po.create
<b>Prerequisite Activities</b>	None
<b>Item Attributes Set by Activity</b>	Event Name, Event Key, Event Message

### Find Supplier (Node 2)

---

Currently this activity does nothing. It represents a point in the process where you can integrate a function that locates a supplier for the purchase order, based on the item being ordered or other criteria.

<b>Function</b>	<i>WF_EVENTDEMO.FINDSUPPLIER</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Buyer: Receive Create PO Event

### Get Agent Details (Node 3)

---

This function activity retrieves the agent details from the subscription that sent the event message to the workflow process. The activity retrieves details for the outbound agent on the Buyer system that should send the message and the inbound agent on the Supplier system that should receive the message.

<b>Function</b>	<i>WF_STANDARD.GETAGENTS</i>
<b>Result Type</b>	Boolean

<b>Prerequisite Activities</b>	Find Supplier
<b>Item Attributes Retrieved by Function</b>	Subscription GUID
<b>Item Attributes Set by Function</b>	From Agent/System1, To Agent/System2

#### **Buyer: Send PO to Supplier Process (Node 4)**

---

This activity is a subprocess that retrieves the name of the order requestor from the purchase order event message, sends the purchase order to the supplier, and notifies the order requestor that the purchase order has been sent to the supplier. To view the subprocess, double-click on Buyer: Send PO to Supplier under the Processes branch in the navigator tree. See: Summary of the Buyer: Send to PO Subprocess: page 15 – 78.

<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Get Agent Details

#### **Buyer: Receive Supplier PO Acknowledgement (Node 5)**

---

This activity is a subprocess that waits to receive a purchase order acknowledgement event message from the supplier. If the acknowledgement is not received within a specified period of time, the subprocess performs a timeout activity to keep notifying the order requestor that the supplier has not responded until the acknowledgement is received. When the acknowledgement is received, the subprocess notifies the order requestor of the acknowledgement.

To view the subprocess, double-click on Buyer: Receive Supplier PO Acknowledgement under the Processes branch in the navigator tree. See: Summary of the Buyer: Receive Supplier PO Acknowledgement Subprocess: page 15 – 80.

<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Buyer: Send PO to Supplier



### **Buyer: Advanced Shipment Notice Process (Node 6)**

---

This activity is a subprocess that receives an advanced shipment notice from the supplier and notifies the order requestor that the order has been shipped. To view the subprocess, double-click on Buyer: Advanced Shipment Notice under the Processes branch in the navigator tree. See: Summary of the Buyer: Advanced Shipment Notice Subprocess: page 15 – 83.

<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Buyer:Receive Supplier PO Acknowledgement

### **Buyer: Receive Supplier Invoicing (Node 7)**

---

This activity is a subprocess that receives an invoice from the supplier and notifies the order requestor that the invoice has arrived. To view the subprocess, double-click on Buyer: Receive Supplier Invoicing under the Processes branch in the navigator tree. See: Summary of the Buyer: Receive Supplier Invoicing Subprocess: page 15 – 85.

<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Buyer:Receive Supplier PO Acknowledgement

### **And (Node 8)**

---

This Standard function activity merges two or more parallel branches in the flow only when the activities in all of those branches complete.

<b>Function</b>	<i>WF_STANDARD.ANDJOIN</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Buyer: Advanced Shipment Notice, Buyer: Receive Supplier Invoicing

### **End (Node 9)**

---

This Standard function activity marks the end of the process.

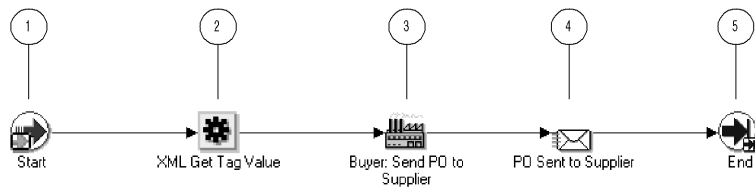
<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	And

---

## Summary of the Buyer: Send PO to Supplier Subprocess

To view the properties of the Buyer: Send PO to Supplier subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Buyer: Send PO to Supplier subprocess, you see that the subprocess consists of five unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.



The subprocess begins at node 1 with the Start activity. At node 2, the process retrieves the name of the order requestor from the purchase order event message. Then the process sends the purchase order to the supplier and notifies the order requestor that the purchase order has been sent. The subprocess ends at this point.

---

## Buyer: Send PO to Supplier Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

### Start (Node 1)

---

This Standard function activity marks the start of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	None

## XML Get Tag Value (Node 2)

---

This Standard external Java function activity retrieves the value of a particular XML tag set in the event message. At this node, the process retrieves the name of the order requestor from the purchase order event message.

<b>Function</b>	<i>oracle.apps.fnd.wf.XMLGetTagValue</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Start
<b>Item Attributes Retrieved by Function</b>	Event Message
<b>Item Attributes Set by Function</b>	Order Requestor

## Buyer: Send PO to Supplier (Node 3)

---

This event activity sends the purchase order event message to the supplier, using the outbound agent on the Buyer system and the inbound agent on the Supplier system identified by the Get Agent Details activity in the Buyer: Top Level PO process.

<b>Event Action</b>	Send
<b>Prerequisite Activities</b>	XML Get Tag Value
<b>Item Attributes Retrieved by Activity</b>	Event Message, From Agent/System1, To Agent/System2

## PO Sent to Supplier (Node 4)

---

This activity notifies the order requestor that the purchase order has been sent to the supplier. The message includes a 'Send' attribute that displays the purchase order number when the notification is sent.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in the item type attribute named Order Requestor.

<b>Message</b>	PO Sent to Supplier
<b>Result Type</b>	None

<b>Prerequisite Activities</b>	Buyer: Send PO to Supplier
--------------------------------	----------------------------

#### **End (Node 5)**

---

This Standard function activity marks the end of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
-----------------	-------------------------

<b>Result Type</b>	None
--------------------	------

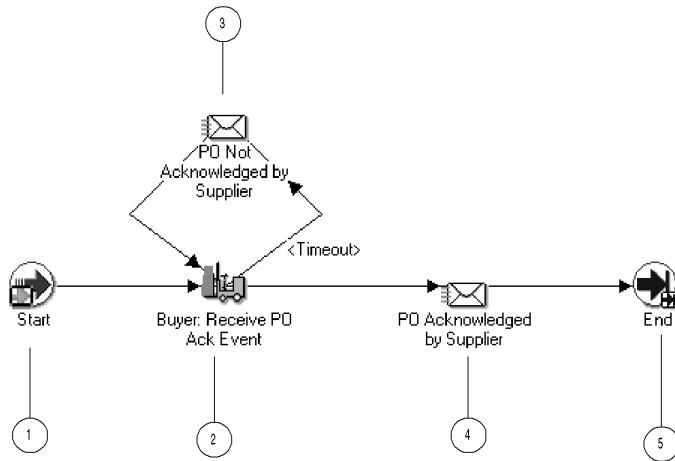
<b>Prerequisite Activities</b>	PO Sent to Supplier
--------------------------------	---------------------

---

## **Summary of the Buyer: Receive Supplier PO Acknowledgement Subprocess**

To view the properties of the Buyer: Receive Supplier PO Acknowledgement subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Buyer: Receive Supplier PO Acknowledgement subprocess, you see that the subprocess consists of five unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.



The subprocess begins at node 1 with the Start activity. At node 2, the process waits to receive a purchase order acknowledgement event message from the supplier. If the acknowledgement is not received within a specified period of time, the subprocess performs a timeout activity to keep notifying the order requestor that the supplier has not responded until the acknowledgement is received. When the acknowledgement is received, the subprocess notifies the order requestor of the acknowledgement. The subprocess ends at this point.

## Buyer: Receive Supplier PO Acknowledgement Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

### Start (Node 1)

This Standard function activity marks the start of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	None

### **Buyer: Receive PO Ack Event (Node 2)**

---

This event activity waits to receive the B2B Purchase Order Acknowledgement event message from the supplier. The activity must be completed within one day; otherwise it times out.

<b>Event Action</b>	Receive
<b>Event Filter</b>	demo.oracle.wf.b2b.po.ack
<b>Prerequisite Activities</b>	Start
<b>Item Attributes Set by Activity</b>	Event Name, Event Key, Event Message

### **PO Not Acknowledged by Supplier (Node 3)**

---

This activity occurs only if the Buyer: Receive PO Ack Event activity times out before being completed. This activity notifies the order requestor that the supplier has not yet acknowledged the purchase order. The message includes a 'Send' attribute that displays the purchase order number when the notification is sent.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in the item type attribute named Order Requestor.

<b>Message</b>	PO Not Acknowledged
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Buyer: Receive PO Ack Event

### **PO Acknowledged by Supplier (Node 4)**

---

This activity notifies the order requestor that the purchase order has been acknowledged by the supplier. The message includes a 'Send' attribute that displays the purchase order number when the notification is sent.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in the item type attribute named Order Requestor.

<b>Message</b>	PO Acknowledged
<b>Result Type</b>	None

**Prerequisite Activities** Buyer: Receive PO Ack Event

### End (Node 5)

---

This Standard function activity marks the end of the process.

**Function** *WF\_STANDARD.NOOP*

**Result Type** None

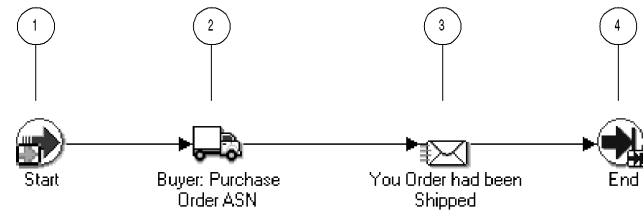
**Prerequisite Activities** PO Acknowledged by Supplier

---

## Summary of the Buyer: Advanced Shipment Notice Subprocess

To view the properties of the Buyer: Advanced Shipment Notice subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Buyer: Advanced Shipment Notice subprocess, you see that the subprocess consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.



The subprocess begins at node 1 with the Start activity. At node 2, the process waits to receive an advanced shipment notice from the supplier. When the advanced shipment notice is received, the process notifies the order requestor that the order has been shipped. The subprocess ends at this point.

---

## Buyer: Advanced Shipment Notice Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

### Start (Node 1)

---

This Standard function activity marks the start of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	None

### Buyer: Purchase Order ASN (Node 2)

---

This event activity waits to receive the B2B Advanced Shipment Notice event message from the supplier.

<b>Event Action</b>	Receive
<b>Event Filter</b>	demo.oracle.wf.b2b.po.asn
<b>Prerequisite Activities</b>	Start
<b>Item Attributes Set by Activity</b>	Event Name, Event Key, Event Message

### Your Order has been Shipped (Node 3)

---

This activity notifies the order requestor that the purchase order advanced shipment notice has been received, indicating that the order has been shipped by the supplier. The message includes a 'Send' attribute that displays the purchase order number when the notification is sent.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in the item type attribute named Order Requestor.

<b>Message</b>	PO Advanced Shipment Notice
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Buyer: Purchase Order ASN



## End (Node 4)

---

This Standard function activity marks the end of the process.

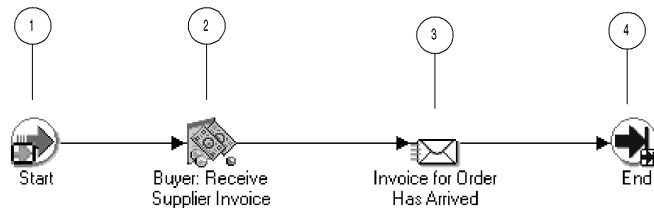
<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Your Order has been Shipped

---

## Summary of the Buyer: Receive Supplier Invoicing Subprocess

To view the properties of the Buyer: Receive Supplier Invoicing subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Buyer: Receive Supplier Invoicing subprocess, you see that the subprocess consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.



The subprocess begins at node 1 with the Start activity. At node 2, the process waits to receive an invoice from the supplier. When the invoice is received, the process notifies the order requestor that the invoice has arrived. The subprocess ends at this point.

---

## Buyer: Receive Supplier Invoicing Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

### Start (Node 1)

---

This Standard function activity marks the start of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	None

### Buyer: Receive Supplier Invoice (Node 2)

---

This event activity waits to receive the B2B Invoice event message from the supplier.

<b>Event Action</b>	Receive
<b>Event Filter</b>	demo.oracle.wf.b2b.po.invoice
<b>Prerequisite Activities</b>	Start
<b>Item Attributes Set by Activity</b>	Event Name, Event Key, Event Message

### Invoice for Order Has Arrived (Node 3)

---

This activity notifies the order requestor that the invoice for the purchase order has been received from the supplier. The message includes a 'Send' attribute that displays the purchase order number when the notification is sent.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in the item type attribute named Order Requestor.

<b>Message</b>	Supplier PO Invoice
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Buyer: Receive Supplier Invoice

## End (Node 4)

This Standard function activity marks the end of the process.

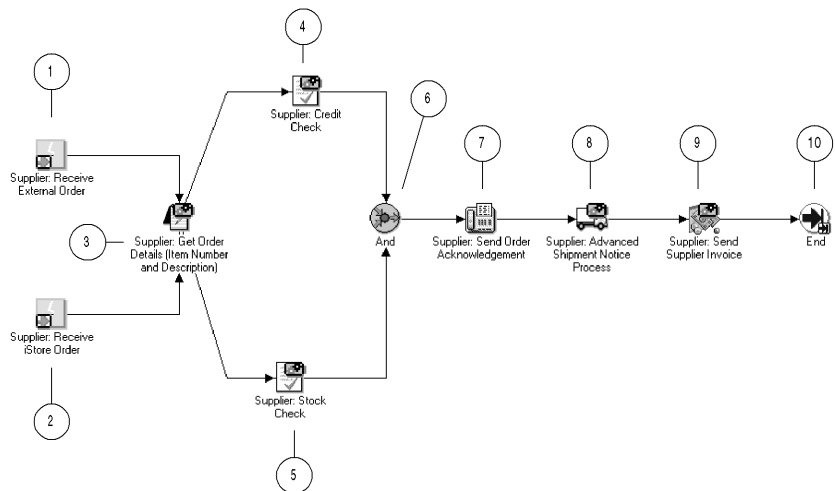
<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Invoice for Order Has Arrived

---

## Summary of the Supplier: Top Level Order Process

To view the properties of the Supplier: Top Level Order process, select the process in the navigator tree, then choose Properties from the Edit menu. This process activity is runnable, indicating that it can be initiated as a top level process to run.

When you display the Process window for the Supplier: Top Level Order process, you see that the process consists of ten unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.



The Supplier: Top Level Order workflow begins when the Supplier system receives the B2B Purchase Order event from the Buyer system. Predefined subscriptions change the correlation ID to consist of the prefix *SO* followed by the event key (the order number), and send the event to the Supplier: Top Level Order process.

The process begins by receiving a purchase order either at node 1 or at node 2, depending on the source of the purchase order.

Node 3 is a subprocess that retrieves the agent details for the intended outbound and inbound agents to send responses to the buyer, as well as retrieving the item number and item description.

Node 4 is a subprocess that performs a credit check if the total cost of the order is greater than 2000. Node 5 is a subprocess that performs a stock check to determine if the ordered item is in stock. When both the credit check and the stock check are complete, the process sends an acknowledgement of the purchase order to the buyer.

Node 8 is a subprocess that checks the shipping details for the order and then sends an advanced shipment notice to the buyer. Node 9 is a subprocess that creates an invoice and sends the invoice to the buyer. After sending the invoice, the process ends.

---

## Supplier: Top Level Order Process Activities

Following is a description of each activity in the process, listed by the activity's display name.

### **Supplier: Receive External Order (Node 1)**

---

This event activity receives the B2B Purchase Order event message that is sent to the Supplier: Top Level Order process by the Event Manager when the event is received from the buyer.

<b>Event Action</b>	Receive
<b>Event Filter</b>	demo.oracle.wf.b2b.po.create
<b>Prerequisite Activities</b>	None
<b>Item Attributes Set by Activity</b>	Event Name, Event Key, Event Message

### **Supplier: Receive iStore Order (Node 2)**

---

This event activity receives a purchase order event message that is sent when an order is placed through the supplier's iStore application. Currently this event is not defined. The activity represents a point in the process where you can integrate the reception of purchase orders from another source.

<b>Event Action</b>	Receive
<b>Event Filter</b>	demo.oracle.wf.istore.po.create
<b>Prerequisite Activities</b>	None
<b>Item Attributes Set by Activity</b>	Event Name, Event Key, Event Message

### **Supplier: Get Order Details (Node 3)**

---

This activity is a subprocess that retrieves the agent details for the intended outbound and inbound agents to send responses to the buyer, as well as retrieving the item number and item description of the ordered item.

To view the subprocess, double-click on Supplier: Get Order Details under the Processes branch in the navigator tree. See: Summary of the Supplier: Get Order Details Subprocess: page 15 – 91.

<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Supplier: Receive External Order or Supplier: Receive iStore Order

### **Supplier: Credit Check (Node 4)**

---

This activity is a subprocess that performs a credit check if the total cost of the order is greater than \$2000.

To view the subprocess, double-click on Supplier: Credit Check under the Processes branch in the navigator tree. See: Summary of the Supplier: Credit Check Subprocess: page 15 – 94.

<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Supplier: Get Order Details

### **Supplier: Stock Check (Node 5)**

---

This activity is a subprocess that performs a stock check to determine if the ordered item is in stock.

To view the subprocess, double-click on Supplier: Stock Check under the Processes branch in the navigator tree. See: Summary of the Supplier: Stock Check Subprocess: page 15 – 96.

<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Supplier: Get Order Details

### **And (Node 6)**

---

This Standard function activity merges two or more parallel branches in the flow only when the activities in all of those branches complete.

<b>Function</b>	<i>WF_STANDARD.ANDJOIN</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Supplier: Credit Check, Supplier: Stock Check

### **Supplier: Send Order Acknowledgement (Node 7)**

---

This event activity sends the B2B Purchase Order Acknowledgement event message to the buyer, using the outbound agent on the Supplier system and the inbound agent on the Buyer system identified by the Get Agent Details activity in the Supplier: Get Order Details subprocess.

<b>Event Action</b>	Send
<b>Prerequisite Activities</b>	And
<b>Item Attributes Retrieved by Activity</b>	Event Message, From Agent/System2, To Agent/System1

### **Supplier: Advanced Shipment Notice (Node 8)**

---

This activity is a subprocess that checks the shipping details for the order and then sends an advanced shipment notice to the buyer.

To view the subprocess, double-click on Supplier: Advanced Shipment Notice under the Processes branch in the navigator tree. See: Summary of the Supplier: Advanced Shipment Notice Subprocess: page 15 – 98.

<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Supplier: Send Order Acknowledgement

### **Supplier: Send Supplier Invoice (Node 9)**

---

This activity is a subprocess that creates an invoice for the purchase order and sends the invoice to the buyer.

To view the subprocess, double-click on Supplier: Send Supplier Invoice under the Processes branch in the navigator tree. See: Summary of the Supplier: Send Supplier Invoice Subprocess: page 15 – 100.

<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Supplier: Advanced Shipment Notice

### **End (Node 10)**

---

This Standard function activity marks the end of the process.

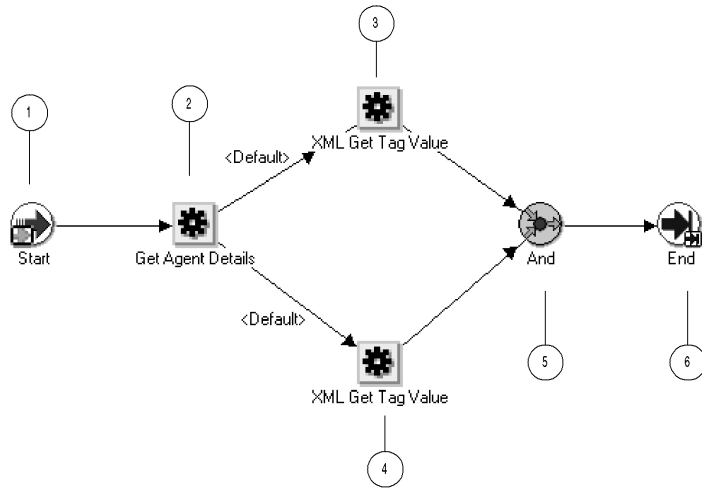
<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Supplier: Send Supplier Invoice

---

## **Summary of the Supplier: Get Order Details Subprocess**

To view the properties of the Supplier: Get Order Details subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Supplier: Get Order Details subprocess, you see that the subprocess consists of five unique activities, one of which is reused to make up the six activity nodes that appear in the workflow diagram. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.



The subprocess begins at node 1 with the Start activity. At node 2, the process retrieves the agent details for the intended outbound and inbound agents to send responses to the buyer. Next, the process retrieves the item number and item description of the ordered item from the purchase order. When both the item number and item description have been retrieved, the subprocess ends.

---

## Supplier: Get Order Details Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

### Start (Node 1)

---

This Standard function activity marks the start of the process.

**Function** *WF\_STANDARD.NOOP*

**Result Type** None

**Prerequisite Activities** None

### Get Agent Details (Node 2)

---

This function activity retrieves the agent details from the subscription that sent the event message to the workflow process. The activity



retrieves details for the outbound agent on the Supplier system that should send the response messages and the inbound agent on the Buyer system that should receive the response messages.

<b>Function</b>	<i>WF_STANDARD.GETAGENTS</i>
<b>Result Type</b>	Boolean
<b>Prerequisite Activities</b>	Start
<b>Item Attributes Retrieved by Function</b>	Subscription GUID
<b>Item Attributes Set by Function</b>	From Agent/System2, To Agent/System1

### **XML Get Tag Value (Node 3)**

---

This Standard external Java function activity retrieves the value of a particular XML tag set in the event message. At this node, the process retrieves the item number from the purchase order event message.

<b>Function</b>	<i>oracle.apps.fnd.wf.XMLGetTagValue</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Get Agent Details
<b>Item Attributes Retrieved by Function</b>	Event Message
<b>Item Attributes Set by Function</b>	Item Number

### **XML Get Tag Value (Node 4)**

---

This Standard external Java function activity retrieves the value of a particular XML tag set in the event message. At this node, the process retrieves the item description from the purchase order event message.

<b>Function</b>	<i>oracle.apps.fnd.wf.XMLGetTagValue</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Get Agent Details

**Item Attributes Retrieved by Function**      Event Message

**Item Attributes Set by Function**      Item Description

---

#### **And (Node 5)**

This Standard function activity merges two or more parallel branches in the flow only when the activities in all of those branches complete.

**Function**      *WF\_STANDARD.ANDJOIN*

**Result Type**      None

**Prerequisite Activities**      XML Get Tag Value (node 3), XML Get Tag Value (node 4)

---

#### **End (Node 6)**

This Standard function activity marks the end of the process.

**Function**      *WF\_STANDARD.NOOP*

**Result Type**      None

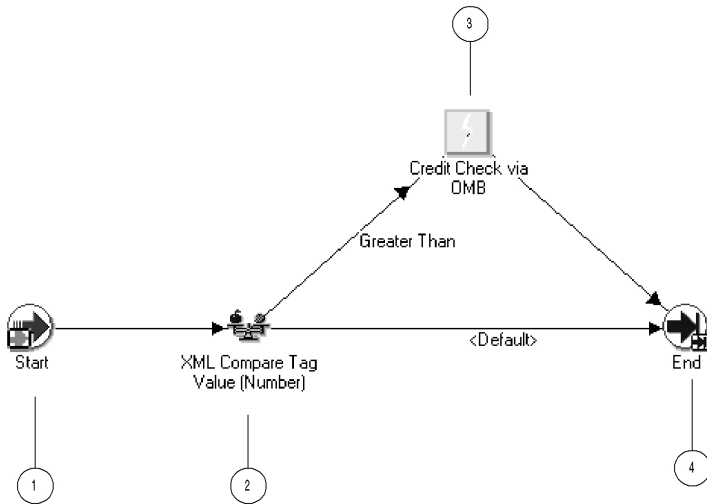
**Prerequisite Activities**      And

---

## **Summary of the Supplier: Credit Check Subprocess**

To view the properties of the Supplier: Credit Check subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Supplier: Credit Check subprocess, you see that the subprocess consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.



The subprocess begins at node 1 with the Start activity. At node 2, the process checks whether the total cost of the order is greater than 2000. If the total cost is equal to or less than 2000, the process ends at this point. If the total cost is greater than 2000, the process initiates a credit check on the buyer and then ends.

## Supplier: Credit Check Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

### Start (Node 1)

This Standard function activity marks the start of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	None

### **XML Compare Tag Value (Number) (Node 2)**

---

This Standard external Java function activity compares the value of a particular XML tag set in the event message with a test value. At this node, the process compares the total cost from the purchase order event message with the test value 2000.

<b>Function</b>	<i>oracle.apps.fnd.wf.XMLCompareTag</i>
<b>Result Type</b>	Comparison
<b>Prerequisite Activities</b>	Start
<b>Item Attributes Retrieved by Function</b>	Event Message

### **Credit Check (Node 3)**

---

This activity occurs only if the total cost of the purchase order is greater than 2000. This event activity raises a credit check event. Currently this event is not defined. The activity represents a point in the process where you can integrate a credit check.

<b>Event Action</b>	Raise
<b>Prerequisite Activities</b>	XML Compare Tag Value (Number)
<b>Item Attributes Retrieved by Activity</b>	Event Key

### **End (Node 4)**

---

This Standard function activity marks the end of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	XML Compare Tag Value (Number)

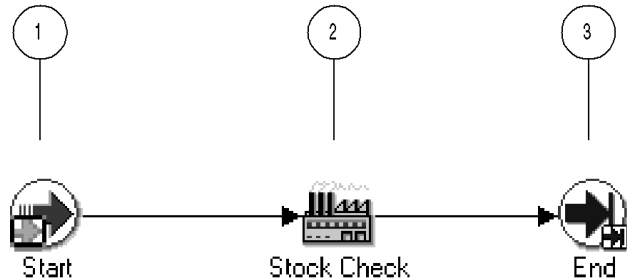
---

## **Summary of the Supplier: Stock Check Subprocess**

To view the properties of the Supplier: Stock Check subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it

cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Supplier: Stock Check subprocess, you see that the subprocess consists of three unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.



The subprocess begins at node 1 with the Start activity. At node 2, the process performs a stock check to determine if the ordered item is in stock. The subprocess ends at this point.

---

## Supplier: Stock Check Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

### Start (Node 1)

---

This Standard function activity marks the start of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	None

### **Stock Check (Node 2)**

---

Currently, this activity does nothing. It represents a point in the process where you can integrate a function that performs a stock check.

<b>Function</b>	<i>WF_EVENTDEMO.STOCKCHECK</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Start

### **End (Node 3)**

---

This Standard function activity marks the end of the process.

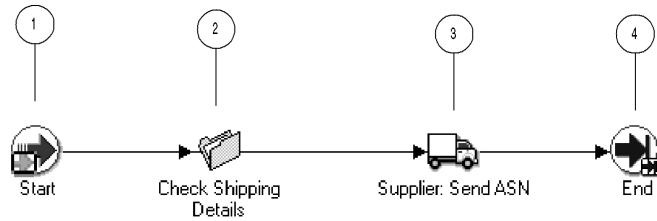
<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Stock Check

---

## **Summary of the Supplier: Advanced Shipment Notice Subprocess**

To view the properties of the Supplier: Advanced Shipment Notice subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Supplier: Advanced Shipment Notice subprocess, you see that the subprocess consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.



The subprocess begins at node 1 with the Start activity. At node 2, the process checks the shipping details for the purchase order. Then the process sends an advanced shipment notice to the buyer. The subprocess ends at this point.

---

## Supplier: Advanced Shipment Notice Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

### Start (Node 1)

---

This Standard function activity marks the start of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	None

### Check Shipping Details (Node 2)

---

Currently, this activity does nothing. It represents a point in the process where you can integrate a function that checks the shipping details for the purchase order.

<b>Function</b>	<i>WF_EVENTDEMO.CHECKSHIPPING</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Start

### **Supplier: Send ASN (Node 3)**

---

This event activity sends the B2B Advanced Shipment Notice event message to the buyer, using the outbound agent on the Supplier system and the inbound agent on the Buyer system identified by the Get Agent Details activity in the Supplier: Get Order Details subprocess.

<b>Event Action</b>	Send
<b>Prerequisite Activities</b>	Check Shipping Details
<b>Item Attributes Retrieved by Activity</b>	Event Message, From Agent/System2, To Agent/System1

### **End (Node 4)**

---

This Standard function activity marks the end of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Supplier: Send ASN

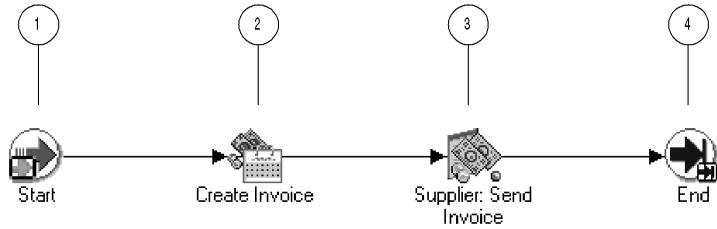
---

## **Summary of the Supplier: Send Supplier Invoice Subprocess**

To view the properties of the Supplier: Send Supplier Invoice subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Supplier: Send Supplier Invoice subprocess, you see that the subprocess consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.





The subprocess begins at node 1 with the Start activity. At node 2, the process creates an invoice for the items ordered in the purchase order. Then the process sends the invoice to the buyer. The subprocess ends at this point.

---

## Supplier: Send Supplier Invoice Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

### **Start (Node 1)**

---

This Standard function activity marks the start of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	None

### **Create Invoice (Node 2)**

---

Currently, this activity does nothing. It represents a point in the process where you can integrate a function that creates an invoice for the items ordered in the purchase order.

<b>Function</b>	<i>WF_EVENTDEMO.CREATEINVOICE</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Start

### Supplier: Send Invoice (Node 3)

---

This event activity sends the B2B Invoice event message to the buyer, using the outbound agent on the Supplier system and the inbound agent on the Buyer system identified by the Get Agent Details activity in the Supplier: Get Order Details subprocess.

<b>Event Action</b>	Send
<b>Prerequisite Activities</b>	Create Invoice
<b>Item Attributes Retrieved by Activity</b>	Event Message, Event Key, From Agent/System2, To Agent/System1

### End (Node 4)

---

This Standard function activity marks the end of the process.

<b>Function</b>	<i>WF_STANDARD.NOOP</i>
<b>Result Type</b>	None
<b>Prerequisite Activities</b>	Supplier: Send Invoice

---

## B2B Purchase Order Event

This event is raised on the Buyer system when you submit a purchase order from the Buyer Workbench demonstration page.

<b>Internal Name</b>	demo.oracle.wf.b2b.po.create
<b>Generate Function</b>	wf_eventdemo.generatexml

Oracle Workflow provides four default subscriptions to the B2B Purchase Order event. The first subscription adds a correlation ID to the event message when the B2B Purchase Order event is raised locally. The correlation ID consists of the prefix PO followed by the event key (the order number). This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	Local
Event Filter	demo.oracle.wf.b2b.po.create
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	wf_eventdemo.derivecorrelationid

**Table 15 – 7 (Page 1 of 1)**

The second subscription sends the event message to the Buyer: Top Level PO process in the Event System Demonstration item type when the B2B Purchase Order event is raised locally. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	Local
Event Filter	demo.oracle.wf.b2b.po.create
Phase	2
Status	Enabled
Rule Data	Message
Rule Function	wf_rule.workflow_protocol
Workflow Item Type	WFEVDEME
Workflow Process Name	RCVPOCRT
Out Agent	WF_OUT@<local system>
To Agent	WF_IN@<local system>

**Table 15 – 8 (Page 1 of 1)**

The third subscription modifies the correlation ID in the event message when the B2B Purchase Order event is received from an external

source. The correlation ID consists of the prefix so followed by the event key (the order number). This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.create
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	wf_eventdemo.derivecorrelationid

**Table 15 – 9 (Page 1 of 1)**

The fourth subscription sends the event message to the Supplier: Top Level Order process in the Event System Demonstration item type when the B2B Purchase Order event is received from an external source. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.create
Phase	2
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.workflow_protocol
Workflow Item Type	WFEVDEME
Workflow Process Name	SUPPORCV

**Table 15 – 10 (Page 1 of 2)**

Subscription Property	Value
Out Agent	WF_OUT@<local system>
To Agent	WF_IN@<local system>

Table 15 – 10 (Page 2 of 2)

---

## B2B Purchase Order Acknowledgement Event

This event is sent from the Supplier system to the Buyer system as a purchase order acknowledgement after the B2B Purchase Order event is received and processed.

**Internal Name**      demo.oracle.wf.b2b.po.ack

**Generate Function**      None

Oracle Workflow provides two default subscriptions to the B2B Purchase Order Acknowledgement event. The first subscription adds a correlation ID to the event message when the B2B Purchase Order Acknowledgement event is received from an external source. The correlation ID consists of the prefix `PO` followed by the event key (the order number). This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.ack
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	wf_eventdemo.derivecorrelationid

Table 15 – 11 (Page 1 of 1)

The second subscription sends the event message to the Buyer: Top Level PO process in the Event System Demonstration item type when

the B2B Purchase Order Acknowledgement event is received from an external source. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.ack
Phase	2
Status	Enabled
Rule Data	Key
Workflow Item Type	WFEVDEME
Workflow Process Name	RCVPOCRT

Table 15 – 12 (Page 1 of 1)

---

## B2B Advanced Shipment Notice Event

This event is sent from the Supplier system to the Buyer system as an advanced shipment notice after the B2B Purchase Order event is received and processed.

**Internal Name**      demo.oracle.wf.b2b.po.asn

**Generate Function**      None

Oracle Workflow provides two default subscriptions to the B2B Advanced Shipment Notice event. The first subscription adds a correlation ID to the event message when the B2B Advanced Shipment Notice event is received from an external source. The correlation ID consists of the prefix `PO` followed by the event key (the order number). This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.asn
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	wf_eventdemo.derivecorrelationid

**Table 15 – 13 (Page 1 of 1)**

The second subscription sends the event message to the Buyer: Top Level PO process in the Event System Demonstration item type when the B2B Advanced Shipment Notice event is received from an external source. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.asn
Phase	2
Status	Enabled
Rule Data	Key
Workflow Item Type	WFEVDEME
Workflow Process Name	RCVPOCRT

**Table 15 – 14 (Page 1 of 1)**

---

## B2B Invoice Event

This event is sent from the Supplier system to the Buyer system as an invoice after the B2B Purchase Order event is received and processed.

**Internal Name** demo.oracle.wf.b2b.po.invoice

**Generate** None

**Function**

Oracle Workflow provides two default subscriptions to the B2B Invoice event. The first subscription adds a correlation ID to the event message when the B2B Invoice event is received from an external source. The correlation ID consists of the prefix PO followed by the event key (the order number). This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.invoice
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	wf_eventdemo.derivecorrelationid

**Table 15 – 15 (Page 1 of 1)**

The second subscription sends the event message to the Buyer: Top Level PO process in the Event System Demonstration item type when the B2B Invoice event is received from an external source. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Property	Value
System	<local system>
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.invoice
Phase	2
Status	Enabled
Rule Data	Key

**Table 15 – 16 (Page 1 of 2)**



Subscription Property	Value
Workflow Item Type	WFEVDEME
Workflow Process Name	RCVPOCRT

**Table 15 – 16 (Page 2 of 2)**



CHAPTER

# 16



## Workflow Administration Scripts

**T**his 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 16 – 5.
- Enable/disable a language—wfnlena.sql: page 16 – 12
- Run a workflow process—wfrun.sql: page 16 – 15.
- Start a background engine—wfbkg.sql: page 16 – 6.
- Show activities deferred for the next background engine execution—wfbkgchk.sql: page 16 – 7.
- Display a status report for an item
  - wfstatus.sql: page 16 – 16.
  - wfstat.sql: page 16 – 16.
- Show a notification's status—wfntfsh.sql: page 16 – 12.
- Reset the protection level for objects—wfprot.sql: page 16 – 12.
- Handle errored activities—wfretry.sql: page 16 – 13.
- Check for version and process definition errors
  - wfverchk.sql: page 16 – 17.
  - wfverupd.sql: page 16 – 17.
  - wfstdchk.sql: page 16 – 16.
- Check for invalid hanging foreign keys—wfrefchk.sql: page 16 – 13.
- Check the directory service data model—wfdirchk.sql: page 16 – 10.
- Clean up Workflow Queues in the system tables—wfaqclean.sql: page 16 – 13.
- 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 16 – 7.
- wfchacta.sql: page 16 – 7.
- wfchita.sql: page 16 – 8.
- wfchitt.sql: page 16 – 8.
- wfchluc.sql: page 16 – 8.
- wfchlut.sql: page 16 – 9.
- wfchmsg.sql: page 16 – 9.
- wfchmsga.sql: page 16 – 9.
- Remove data from Oracle Workflow tables
  - wfrmall.sql: page 16 – 14.
  - wfrmitms.sql: page 16 – 15.
  - wfrmitt.sql: page 16 – 15.
  - wfrmtype.sql: page 16 – 15.
  - wfrmita.sql: page 16 – 14.

**Note:** In Oracle Applications, a standard concurrent program called "Purge Obsolete Workflow Runtime Data" is also available. See: FNDWFPR: page 16 – 5.

If you are using the standalone version of Oracle Workflow available with Oracle9i Release 2, you can use the standalone Oracle Workflow Manager component available through Oracle Enterprise Manager to submit and manage Workflow purge database jobs. For more information, please refer to the Oracle Workflow Manager online help.

- Purge and repopulate the outbound message queue for the Notification Mailer—wfmqupd.sql: page 16 – 12.



**Warning:** This script should only be used when Oracle Support determines it to be necessary. Do not run this script unless directed by Oracle Support.

- Display the version of the Oracle Workflow server—wfver.sql: page 16 – 16.
- Stop the Java Function Activity Agent—wfvstop.sql: page 16 – 11.

- Enqueue an event message on a queue using an override agent—wfevtenq.sql: page 16 – 10.
- Run a listener to monitor an agent for inbound event messages—wfagtlst.sql: page 16 – 6.

**Note:** In Oracle Applications, a standard concurrent program called “Workflow Agent Listener” is also available. See: FNDWFLST: page 16 – 4.

If you are using the standalone version of Oracle Workflow available with Oracle9i Release 2, you can use the standalone Oracle Workflow Manager component available through Oracle Enterprise Manager to submit and manage Workflow agent listener database jobs. For more information, please refer to the Oracle Workflow Manager online help.

---

## FNDWFLST

For Oracle Workflow embedded in Oracle Applications, use the standard concurrent program FNDWFLST “Workflow Agent Listener” to monitor an agent for inbound event messages.

Navigate to the Submit Requests form in Oracle Applications to submit the Workflow Agent Listener 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 must supply the name of the agent that you want to monitor as a parameter for the Workflow Agent Listener concurrent program.

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager to submit and manage the Workflow Agent Listener concurrent program. For more information, please refer to the Oracle Applications Manager online help.

---

## 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.
- **Persistence Type**—The persistence type to be purged, either 'TEMP' for Temporary or 'PERM' for Permanent. The default is 'TEMP'.

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager to submit and manage the Purge Obsolete Workflow Runtime Data concurrent program. For more information, please refer to the Oracle Applications Manager online help.

---

## 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 – 38* and *wflena.sql: page 16 – 12*.

Use the script as follows:

```
sqlplus <user/pwd> @WFNLADD
```

---

## Wfagtlst.sql

Use wfagtlst.sql to run a listener to monitor an agent for inbound event messages. When a message is received, the Event Manager searches for and executes any active subscriptions by the local system to the event with a source type of External, and also any active subscriptions by the local system to the Any event with a source type of External.

Use the script as follows:

```
sqlplus <user/pwd> @wfagtlst <agent_name>
```

Replace *<agent\_name>* with the internal name of the agent that you want to monitor for inbound event messages.

**Note:** You should use this script primarily for debugging purposes.

### See Also

Listen: page 8 – 270

---

## 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 its 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 – 41

Setting up Background Workflow Engines: page 2 – 43



---

## Wfbkgchk.sql

Use wfbkgchk.sql to get a list of all activities waiting to be processed by the background engine the next time it runs.

Use the script as follows:

```
sqlplus <user/pwd> @wfbkgchk
```

## See Also

Background: page 8 – 41

Setting up Background Workflow Engines: page 2 – 43

---

## 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 16 – 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 16 – 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 16 – 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\_attr>* 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 16 – 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 16 – 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 16 – 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 16 – 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 16 – 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 e-mail 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
```

---

## Wfevtmq.sql

Use wfevtmq.sql to enqueue an event message on a local queue using an override agent. This script constructs an event message using the event name, event key, event data, From Agent, and To Agent you specify. Then the event message is enqueued on the queue associated with the override agent you specify, which can be different than the From Agent listed inside the event message. If no override agent is

specified, the event message is enqueued on the message's From Agent by default.

**Note:** This script can only enqueue an event message onto a queue for an agent on the local system.

Use the script as follows:

```
sqlplus <user/pwd> @wfevteng <overrideagent> <overridesystem>
<fromagent> <fromsystem> <toagent> <tosystem> <eventname>
<eventkey> <message>
```

Replace the variables with your parameters as follows:

- *<overrideagent>*—The agent on whose queue you want to enqueue the event message
- *<overridesystem>*—The system where the override agent is located
- *<fromagent>*—The From Agent that you want to list in the event message
- *<fromsystem>*—The system where the From Agent is located
- *<toagent>*—The To Agent that receives the event message
- *<tosystem>*—The system where the To Agent is located
- *<eventname>*—The internal name of the event
- *<eventkey>*—The event key that uniquely identifies the instance of the event
- *<message>*—The event data

---

## Wfjvstop.sql

Use wfjvstop.sql to stop the Java Function Activity Agent by placing a stop message on the 'Outbound' queue.

Use the script as follows:

```
sqlplus <user/pwd> @wfjvstop
```

---

## Wfmqupd.sql

Use wfmqupd.sql to purge the outbound message queue for the Notification Mailer and repopulate the queue from the WF\_NOTIFICATIONS table.

Use the script as follows:

```
sqlplus <user/pwd> @wfmqupd
```



**Warning:** This script should only be used when Oracle Support determines it to be necessary. Do not run this script unless directed by Oracle Support.

---

## 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 16 – 5.

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 – 77.

---

## 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.

---

## Wfrmtime.sql

Use wfrmtime.sql to delete runtime data associated with a given item type. This script prompts you for an item type to purge from a list of valid item types, then asks you to choose between deleting all runtime data associated with the specified item type or deleting only runtime data for the completed activities and items of the specified item type.

Use the script as follows:

```
sqlplus <user/pwd> @wfrmtime
```

---

## 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

**T**his 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 – 22.

**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 – 21

**Print Diagram**—Prints the process diagram displayed in the active process window. See: To Print a Process: page 5 – 20.

**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 – 24.

**Exit**—Exits Oracle Workflow Builder.

## **Edit Menu**

---

The Edit menu varies depending on whether you select the Navigator window or a process window. The following menu options appear only when you select the Navigator window and apply only to the Navigator window:

**New**—Creates a new item type, function activity, process activity, notification activity, event activity, message, lookup type, lookup code, or attribute by displaying its property page(s).

**Copy**—Copies the selected object in the navigator tree.

**Paste**—Pastes the object from the clipboard into the selected branch of the navigator tree.

**Delete**—Deletes the selected object from the navigator tree.

**Find**—Displays the Search window so you can enter search criteria to find an object in the navigator tree. See: To Find an Object in the Navigator Tree: page 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 – 20

## 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 – 21.

**Log -> Show**—Toggles between displaying and hiding the Log window. The Message Log window displays messages from the Workflow Builder that are not error-related.

**Log -> Detailed**—Toggles the debug mode of Oracle Workflow Builder on and off. When you check Detailed, you turn the debug mode on and cause Oracle Workflow Builder to write more extensive messages to the Log window. You should not check Detailed unless instructed to do so by your Oracle customer support representative, as this mode significantly slows down the Oracle Workflow Builder.

**Log -> To File**—Writes all future content of the Message Log window to a file. Select Log Show from the View menu to determine the location and name of the log file.

**Log -> Bring to Front**—Brings the Message Log window to the front as the active window.

**Grid Snap**—Toggles grid snap on or off for all process windows.

**Show Label in Designer submenu**—A submenu of options that let you control the information displayed in an activity's label. Choose either Instance Label, Internal Name, Display Name, Performer, or Comment.

**Show Label in Designer -> Instance Label**—Uses the node label as the label for each activity node in a process diagram. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.

**Show Label in Designer -> Internal Name**—Uses the internal name of an activity as the label for each activity node in a process



diagram. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.

**Show Label in Designer -> Display Name**—Uses the display name of an activity as the label for each activity node in a process diagram. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.

**Show Label in Designer -> Performer**—Uses the activity's performer as the label for each activity node in a process diagram. Function and process activities that do not have performers do not have a label. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.

**Show Label -> Comment**—Uses the activity's comment as the label for each activity node in a process diagram. Activities that do not have a comment do not have a label. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.

**Developer Mode**—Toggles the display between standard presentation mode and developer mode. In developer mode, all icons revert to the default icon for the specific object type/subtype, subprocess icons are distinct from top level process icons, and in the navigator tree, objects are shown and sorted by internal name. Note that attributes are shown by internal name but are not sorted.

If the Navigator window is the active window, then the following menu option also appears:

**Split Window**—Splits the Navigator window horizontally or vertically.

If a process window is the active window, then the following menu options also appear:

**Overview**—Displays the process Overview window. See: To Display a Process Overview: page 5 – 19.

**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.



**New Event**—Displays the event activity node property page for you to create a new event activity.



**Delete Selection**—Deletes the selected object.



**Properties**—Shows the property pages of the selected object.



**Developer Mode**—Toggles between Developer and Presentation display.



**Find**—Displays the process Overview window.



**Show Instance Labels**—Displays the instance label of the node as the node activity label in the Process window.



**Show Internal Names**—Displays the internal name of the node as the node activity label in the Process window.



**Show Display Names**—Displays the display name of the node as the node activity label in the Process window.



**Show Comments**—Displays the comments of the node as the node activity label in the Process window.



**Show Performers**—Displays the performer of the node as the node activity label in the Process window.



**Help**—Displays help on how to use Oracle Workflow.



# B

## Oracle Workflow Implementation in Other Oracle Products

**T**his appendix lists embedded workflows and Business Event System implementation in Oracle E-Business Suite and the Oracle9i platform, as well as Oracle's support policy towards the customization of these workflows, events, and subscriptions.

---

## Predefined Workflows Embedded in Oracle E–Business Suite

You can use Oracle Workflow to customize the predefined workflow processes listed below. A full description of each workflow is documented in its respective product's User's Guide or 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, Events, and Subscriptions: page B – 20

---

### **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 – 22.

## **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 Internet Expenses**

---

**AP Expense Report Process**—Oracle Internet Expenses uses the AP Expense Report workflow process to process the manager approval and accounting review of expense reports entered in Internet Expenses. The AP Expense Report process begins when a user submits an expense report, and finishes when an expense report is rejected, or when a manager has approved and accounting has reviewed an expense report. If approved and reviewed, the workflow process makes the expense report available for the Payables Invoice Import program. The AP Expense Report process notifies employees at key event points during the manager approval and accounting review processes.

**AP Credit Card Process**—The AP Credit Card workflow process notifies employees and managers of payments made to employees and payments made to credit card issuers for charges to travel and entertainment (T&E) credit cards. It also sends notifications to employees and managers for any outstanding credit card transactions that have not been submitted on an expense report, or have been submitted on an expense report but the expense report has not completed the AP Expense Report process. The payment notifications are generated when payments are made in Oracle Payables; the outstanding charges notifications are sent when you run the Credit Card Outstanding Charges Report program from Oracle Payables.

### **Procurement Cards**

The following procurement card workflow processes enable your self-service employees to verify and approve procurement card transactions.

**AP Procurement Card Employee Verification Workflow Process**—The AP Procurement Card Employee Verification Workflow process notifies and confirms procurement card transactions with card holders. This workflow process is initiated when you submit the Distribute

Employee Card Transaction Verifications program from Oracle Payables. The process notifies an employee of transactions charged to the employee's procurement card, and optionally requires the employee's manual verification.

**AP Procurement Card Manager Approval Transaction Process**—The AP Procurement Card Manager Approval Transaction workflow process notifies and confirms verified procurement card transactions with a card holder's manager. This workflow process is initiated when you submit the Distribute Manager Card Transactions Approvals program from Oracle Payables. The AP Procurement Card Employee Verification Workflow process must first complete for transactions before the manager workflow process is used. The process notifies managers, and optionally requires their manual approval, of procurement card transactions incurred by employees.

### **Oracle 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 managers 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 Internet Time**

---

**PA Timecard Approval Process**—This process is initiated when an employee submits a timecard in Oracle Internet Time. The workflow can be configured to either automatically approve all timecards or route the timecard through a pre-determined approval process. The approval process sends notifications to managers and employees, ensures timecards adhere to company policy, and checks manager approval levels. The status of submitted timecards can be monitored throughout the approval process.

### **Oracle Web Suppliers**

---

**Supplier Self-Service Registration Approval Process**—Oracle Web Suppliers allows a guest to log on and register as a supplier contact for a company. This process routes a notification to the appropriate account approver to verify and approve the registration. If the approver approves the registration, the Supplier Web User account is activated. If the account approver rejects the registration, the account is deactivated.

### **Oracle Configure To Order**

---

**CTO Change Order**—Sends a notification to the planner in the shipping organization detailing the changes made to an ATO order. It is started when a change to the order quantity, schedule ship date, request date, or schedule arrival date is made and a discrete job reservation or a flow schedule exists for the order. It is also started anytime a configuration change or order line cancellation is made, even if a reservation does not exist.

## Oracle Demand Planning

---

**MSD Demand Planning Cycle Process**—The MSD Demand Planning Cycle manages all background processing during a Demand Planning cycle. It is made up of four stages, each of which initiates a specific Workflow process to govern a task that is performed during that stage. Each stage is initiated from the Demand Planning Administrator page. Notifying the administrator and user community of relevant processing status as the workflow progresses through its many stages is a central benefit.

The Demand Planning processing cycle is made up of the following four stages, which correspond to processes:

- **Downloading data from the Planning Server**—Manages the transfer and transformation of data from the Demand Planning Server to the Express Server Demand Planning Engine for analytic processing.
- **Forecasting and distributing data to demand planners**—Runs a forecast engine and makes data available to the community of demand planners.
- **Collecting and consolidating data from demand planners**—Collects submitted data from demand planners and consolidates data in the Demand Planning Engine.
- **Uploading the consolidated data to the Planning Server**—Transfers transformed data back to the Planning Server.

## Oracle Engineering

---

**Engineering Change Orders Process**—Submits an engineering change order to the appropriate people for approval.

## Oracle General Ledger

---

**Journal Approval Process**—You can require journal batches to be approved before posting. Create an approval hierarchy and define authorization limits for each user. The Journal Approval process is initiated when you try to post a journal batch. The process automatically routes journals to the appropriate user for approval, based on the approval hierarchy.

**AutoAllocations Process**—When you generate step-down AutoAllocations, the workflow process initiates the AutoAllocation process and validates and generates the Mass Allocation and Recurring Journal batches that are defined in the AutoAllocation. The workflow process also determines whether journal approval is required for each

generated journal batch, submits the batches to the appropriate users for approval if required, and notifies the appropriate users of the approval results. If an error occurs during the AutoAllocation process, the designated user or users can choose to roll back the AutoAllocation process, which reverses any posted journals.

**Global Intercompany System**—The Global Intercompany System (formerly CENTRA) is an enhanced feature for Release 11*i*, and has been backported to Release 11. It provides an environment for multiple companies to exchange intercompany transactions. The workflow process notifies the receiver company when a sender company initiates an intercompany transaction and requires approval from the receiver, or when the sender company recalls or reverses an intercompany transaction. The workflow process notifies the sender company when a receiver company approves or rejects an intercompany transaction that the sender had initiated. In addition, a threshold amount can be set to limit the volume of notifications. The workflow process is initiated when the sender submits, recalls, or reverses an intercompany transaction, or when the receiver rejects or accepts an intercompany transaction.

## **Oracle Grants Accounting**

---

**Grants Accounting Workflow Process**—The Grants Accounting Workflow process notifies key members that an installment has been activated or that a report is due. The Budget Subprocess notifies the budget approver or award manager that a budget has been submitted for approval.

The workflow process is initiated at the following points:

- installment is activated
- report is due
- budget is submitted
- budget is approved/baselined

## **Oracle Grants Proposal**

---

**Proposal Approval Process**—The Proposal Approval Process is initiated when a proposal is submitted for approval.

Notifications are sent to approvers and the workflow process waits for the response from each approver before proceeding to the next approver in the hierarchical proposal approval map.

The proposal is approved if all approvers approve the proposal. The proposal is rejected if any approvers reject it. The person submitting the

proposal for approval is notified of the approval status at every stage during the approval process.

**Notify Approval Subprocess**—The Notify Approval Subprocess is initiated during the Proposal Approval Process when the next approver in the hierarchical approval map is selected.

The Notify Approval Subprocess notifies the approver that a proposal is pending for approval. The approver can approve or reject the proposal.

If the approver fails to approve or reject the proposal within a given time frame, the approver receives periodic reminders. Organizations can set the timeout, which defines the time frame in which the reminders are sent. By default, the timeout is not set.

**Notify Proposal Members Process**—The Notify Proposal Members Process sends notifications to personnel on the proposal.

### **Oracle Labor Distribution**

---

**Effort Report Notification Process**—Workflow functionality in Labor Distribution automatically routes effort reports throughout the organization and delivers electronic notifications to users regarding effort reports that require their attention or processes that are completed.

The Effort Report Notification workflow process includes the following subprocesses:

- approval
- notification

The Effort Report Notification workflow process is initiated in Labor Distribution when an effort report is created.

Notification is sent to approvers of the effort report. When the effort report is approved, the effort report is sent to a supervisor for certification. The creator of the effort report can monitor the status of the effort report.

**Distribution Adjustment Approval Notification Process**—Workflow functionality in Labor Distribution automatically routes distribution adjustments approval notifications throughout the organization and delivers electronic notifications to users regarding distribution adjustments that require their attention or processes that are completed. The process is initiated when a distribution batch is submitted.

### **Oracle Public Sector Budgeting**

---

**Distribute Worksheet Workflow Process**—The Distribute Worksheet Workflow Process distributes worksheets and notifies users that a

worksheet has been distributed. The process is initiated when distributing a worksheet.

**Submit Worksheet Workflow Process**—The Submit Worksheet Workflow Process submits worksheets. Based on user-defined parameters, the process performs constraint validations, worksheet operations, copying, and merging. The process moves worksheets from one budget stage to the next and routes the worksheets through an approval process for required approvals. The process also freezes and unfreezes worksheets. Notifications are sent to users who initiate a process and to approvers.

The process is initiated at the following points:

- validating a worksheet constraint
- freezing a worksheet
- unfreezing a worksheet
- moving a worksheet to the next stage
- copying a worksheet
- merging a worksheet
- submitting a worksheet

**Distribute Budget Revision Workflow Process**—The Distribute Budget Revision Workflow Process distributes budget revisions and notifies users that budget revisions have been distributed. The process is initiated when distributing a budget revision.

**Submit Budget Revisions Workflow Process**—The Submit Budget Revisions Workflow Process submits budget revisions. Based on user-defined parameters, the process performs constraint validations and other budget revision operations. The Submit Budget Revisions process routes budget revisions through an approval process and updates the status and baseline values for budget revisions. The process also performs funds reservation and posts revisions to General Ledger. The process freezes and unfreezes budget revisions. Notifications are sent to users who initiate a process and to approvers.

The process is initiated at the following points:

- validating a budget revision constraint
- freezing a budget revision
- unfreezing a budget revision
- submitting a budget revision

## **Oracle 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 Management**

---

**Order and Line Runnable Processes and Functional Subprocesses**—Oracle Order Management provides seeded order and line runnable processes and functional subprocesses. Order Header workflow data is defined under the item type OM Order Header (OE OH). Seeded header runnable processes are provided to support the processing of standard Orders and Returns with approvals. Booking and Close Order



functional subprocesses are also seeded. Order Line workflow data is defined under the item type OM Order Line (OEOL). Oracle Order Management comes seeded with line-level runnable processes to support the processing of standard items, configurations, service items, drop-shipments, and other items. Functional subprocesses to schedule, ship, fulfill, invoice interface, and close order lines are also seeded. Additionally, you can configure custom processes to meet your specific business requirements using the seeded functional subprocesses and custom activities. You can use these seeded and custom runnable processes for order processing by assigning them to specific order and line transaction types.

**Change Order Notification Process**—Sends a change order notification from certain application forms. The recipient and message content are set dynamically when you select a responsibility and provide content. Uses the item type OM Change Order (OECHORD).

**COGS Process**—Generates a cost of goods sold account using the item type Generate Cost of Goods Sold Account (OECOGS).

## **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. To initiate the Open Interface Import process, submit Payables Open Interface Workflow from the Submit Requests window.

For more information about related workflows in Oracle Internet Expenses, including expense reports and procurement cards, see Oracle Internet Expenses: page B – 3.

## **Oracle Advanced Supply Chain Planning**

---

**Advanced Planning Exception Message Process**—Sends notifications to suppliers, customer contacts, or internal personnel that inform them of advanced planning exceptions and lets the recipients initiate appropriate action to correct the planning exception.

**Allocated ATP Process**—Sends notification to planners if there was any stealing between different supply sources to satisfy an Order Promising request or if the Order Scheduling process failed.

## 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.

**Step Down Allocations Process**—Automates the execution of step-down auto allocation sets to create allocation runs, generate the allocation transactions, release the allocation transactions (or require approval before the process proceeds), distribute costs, and update project summary amounts.

## Oracle Project 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 subplot 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 automatically created in Oracle Receivables. Otherwise, the process notifies the requestor with an explanation of why it was not approved.

You initiate the Credit Memo Request workflow from iReceivables. iReceivables is a web-based, self-service application that enables registered users to access their Receivables account information using a standard web browser. When an iReceivables user chooses the Dispute a Bill function, Receivables places the specified amount in dispute and initiates the Credit Memo Request process to route the request for approval.

---

## 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 Training Administration**

---

**OTA Workflow Process**—Includes workflows for Self-Service Oracle Training Administration and for Training Administration through Order Management.

- Self-Service Oracle Training Administration:
  - Checks to see if an existing event is full, then notifies student of enrollment or placement on waiting list.
  - If enrollment cancellation occurs to close to the event, cost transfer takes place, charging the customer for the enrollment.
  - When Oracle Training Administration is set to create order lines automatically, notifies event owner when it cannot find the Transfer From or Transfer To values.
  - Notifies enrollment request creator of changes to enrollment status.
  - Notifies enrollee and supervisor of enrollment cancellation.
- Training Administration through Order Management:
  - Enables invoicing from the order line only after the student has completed the course.
  - Notifies student of event or enrollment cancellation.
  - Notifies event owner of enrollment cancellation, and can enroll students from the waiting list.
  - Notifies event owner when the maximum number of event attendees has increased.
  - Notifies event owner when a customer has switched enrollments from one event to another.

---

# Oracle Workflow Business Event System Implementation in Oracle E-Business Suite

The products listed below leverage the Oracle Workflow Business Event System for business process integration. A full description of each feature is documented in its respective product's User's Guide or Configuration Guide, if one is available.

## See Also

Oracle Support Policy for Predefined Workflows, Events, and Subscriptions: page B – 20

---

### Oracle Payables

**E-mail Remittance Advice Program** – When you confirm a payment batch or create a Quick payment, the Business Event System initiates this program which automatically sends an e-mail to each supplier that has a remittance advice e-mail address defined.

---

### Oracle XML Gateway

Oracle XML Gateway leverages the Oracle Workflow Business Event System to publish and subscribe to application business events of interest to automatically trigger message creation or consumption. Seeded Workflow functions are provided for use in Workflow processes to interact directly with the XML Gateway Execution Engine to generate outbound or to consume inbound messages. The outbound messages generated by the Execution Engine are made available to the downstream Workflow activity for processing. The Execution Engine consumes the inbound messages passed to it by a Workflow process.

Two item types are delivered with the XML Gateway: the XML Gateway Standard Item Type and the XML Gateway Error Processing Item Type.

**XML Gateway Standard Item Type** – The XML Gateway Standard Item Type includes the Raise Document Delivery Event, which is used to raise a business event from an existing Workflow process. This allows you to seamlessly integrate your existing Workflow process with Oracle XML Gateway to create an outbound XML message. The functions included with the XML Gateway Standard Item Type are Consume XML Document, Generate XML Document, Generate Trading Partner XML Document, Send Document, Transform XML, and Transaction Delivery Required.

Configure the seeded events and event subscriptions delivered by the Oracle E-Business Suite for pre-built XML messages in support of Business-to-Business or Application-to-Application integration.

**XML Gateway Error Processing Item Type** – The XML Gateway Error Processing Item Type contains error handling processes to manage errors detected by the Oracle Workflow Business Event System or Oracle XML Gateway. The error processes are: Default Error Process, ECX Engine Notification Process, ECX Main Error Process, ECX Main Inbound Error Process, ECX Main Outbound Error Process, Error Handling for Inbound Messages, and Error Handling for Outbound Messages.

The XML Gateway Error Processing Item Type supports two event activities: Receive Error and Receive Send Notification Event. The Receive Error event is used by the XML Gateway to indicate that the XML Gateway execution engine has detected an error. The Receive Send Notification Event is used to indicate that the execution engine has identified a need to send a notification for errors related to an inbound process.

Oracle Workflow error handling provides active error notification to the XML Gateway System Administrator or Trading Partner with support for the Workflow retry and reprocess features. The functions provided by the XML Gateway Error Processing Item Type are: ECX Reprocess Inbound, ECX Resend Outbound Message, Get ECX In Error Details, Get ECX Out Error Details, Get System Administrator Role, and Get Trading Partner Role.

For more information, see: *Oracle XML Gateway User's Guide*.

---

## Oracle Workflow Implementation in the Oracle9i Platform

The products listed below leverage Oracle Workflow for business process definition and integration. A full description of each feature is documented in its respective product's User's Guide or Configuration Guide, if one is available.

### See Also

Oracle Support Policy for Predefined Workflows, Events, and Subscriptions: page B – 20

---

#### Oracle Warehouse Builder

Oracle Warehouse Builder includes a Workflow Deployment Wizard that lets you deploy extract, transform, and load mappings to Oracle Workflow as functions within an item type. You can then use Oracle Workflow Builder to define the sequence of these functions as a workflow process. In designing the process, you can specify job dependencies between the mappings to ensure that jobs run in the proper order. You can then run the process from Oracle Workflow or schedule the process to run using Oracle Enterprise Manager.

By defining jobs as a workflow process, you can automate the entire schedule for a set of jobs. When you run the process, Oracle Workflow manages the jobs so that they run in the sequence defined in the process. If an exception occurs, Oracle Workflow terminates the process. This approach minimizes the manual intervention required to manage warehouse load and refresh jobs.

For more information, see: Managing Dependencies Using Oracle Workflow, *Oracle Warehouse Builder User's Guide*.

---

#### Oracle9iAS InterConnect

The Oracle Workflow Business Event System enables Oracle9iAS InterConnect and Oracle Workflow to work together to provide a complete business process driven integration solution. With Oracle9iAS InterConnect and Oracle Workflow, you can define business collaborations across two or more applications to implement the business processes for an organization.

Simple business process definitions can be implicitly captured in the messaging defined through Oracle9iAS InterConnect core functionality. For more complex business processes, Oracle9iAS InterConnect leverages the robust design time and runtime Oracle Workflow



business process definition and execution support to make the processes explicit and manageable. For example, Oracle Workflow allows you to model error management for exceptions, human interaction such as approvals, message junctions including both message fan-in and fan-out, stateful routing, and composite services involving communication across several applications.

The Oracle9iAS InterConnect iStudio design tool automatically generates Oracle Workflow business event and subscription definitions corresponding to common view events and procedures. You can launch the Oracle Workflow home page from iStudio to review these definitions.

The iStudio tool also deploys process bundles as Oracle Workflow item type definitions. These item types include starter workflow processes with Oracle Workflow event activities that correspond to Publish, Subscribe, Invoke, and Implement activities defined in iStudio. You can then launch Oracle Workflow Builder from iStudio to complete the workflow process definition by specifying the sequence of the event activities and optionally adding other activities such as notifications or functions.

For example, iStudio might generate a workflow process with two event activities, one that receives a CreatePO event and another that sends an AcceptPO event. You can then use Oracle Workflow Builder to define the business process that controls the execution of these activities. For instance, add a notification activity to send an e-mail requesting approval after the CreatePO event is received and before the AcceptPO is event is sent.

At runtime, Oracle9iAS InterConnect and Oracle Workflow communicate with each other through the Oracle Workflow Business Event System, leveraging the Oracle Advanced Queuing messaging infrastructure, to execute business processes defined across multiple applications.

For more information, see: Oracle9iAS InterConnect and Oracle Workflow, *Oracle9iAS InterConnect User Guide*.

---

# Oracle Support Policy for Predefined Workflows, Events, and Subscriptions

Oracle Workflow is embedded in Oracle Applications and is used by its modules to automate and streamline business processes. You can use Oracle Workflow Builder to easily modify an existing business process without changing its application's code. Oracle Workflow also allows you to extend your workflow processes as your business rules change and mature. Additionally, you can use the Event Manager to modify event and subscription definitions without changing application code.

Before you use Oracle Workflow to customize any predefined workflow process, event, or subscription, you should familiarize yourself with the following customization guidelines to ensure standard and safe design and development practices. By following these guidelines, you will be able to supply important information to Oracle Support Services in helping you resolve any issues that arise from your customizations.

---

## Customization Guidelines

1. Verify that all setup steps have been completed as documented in the Oracle Workflow Guide, and the product-specific User's Guides.
2. Test the unmodified seeded workflow, event, or subscription on a test database and ensure that it runs successfully with the setup and data specific to your environment.
3. Refer to the product-specific User's Guide and any documentation update, available on MetaLink, for the specific workflow, event, or subscription of interest. These documentation sources specifically mention what should NOT be modified. Oracle Support Services will not support modifications to any object that is specifically documented as not modifiable.
4. Gradually build in customizations step-by-step, and test the customized workflow or subscription after each step.
5. When creating PL/SQL procedures, conform to the standard PL/SQL API templates documented in the Oracle Workflow Guide. Be sure to handle exceptions in the event of an error so you can track down the procedure where the error has occurred.
6. Do not implement the customized workflow, event, or subscription in production without fully ensuring that it works successfully on a test database, which is a replica of your production setup.

---

## Resolving Customization Issues

If you encounter a problem when customizing a seeded workflow, event, or subscription, you should:

- Provide the Support analyst with the modified Workflow definition file or event or subscription definition, and where possible, identify the exact step where the problem occurred.
- Provide the Support analyst with results of running the unmodified seeded Workflow or subscription.

---

## What Is NOT Supported

The following types of customizations are not supported:

1. Modifying a workflow object that has a protection level that is less than 100.
2. Altering a workflow object's protection level if its original protection level is less than 100.
3. Modifying your access level to an unauthorized level of less than 100 for the purpose of modifying workflow objects that are protected at levels less than 100.
4. Customizations that are explicitly documented as being UNSUPPORTED in the seeded workflow's product-specific User's Guide or documentation update notes. This includes modifying processes, attributes, function activities, event activities, notifications, lookup types, messages, events, or subscriptions that are specifically documented as not to be modified.
5. Manual 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, event's, or subscription's product-specific User's Guide or documentation update notes.
2. Customization examples documented in the product-specific User's Guide or documentation update notes. Any issues that

arise are fully supported to resolution, to the extent that the customization example was followed as documented. Any deviation from what is documented amounts to a custom development issue that needs further evaluation. See number 3 below.

3. Customizations that are not explicitly stated as unsupported customizations, as required customizations, or as supported customization examples are supported to the extent that the customer must first isolate the problem following the Customization Guidelines discussed earlier. If upon evaluation, Oracle Support Services deems that the isolated problem stems from an Oracle product, Oracle will supply a solution. Otherwise, it is the responsibility of the customer to correct the custom development issue.

APPENDIX

# C



## Oracle Workflow Performance Concepts

**T**his appendix describes concepts and techniques that you can use for performance gain when running Oracle Workflow.

---

## Oracle Workflow Performance Concepts

The performance of Oracle Workflow depends on several different factors. You can enhance performance in your workflow process design through effective use of synchronous, asynchronous, and forced synchronous processes, item attributes, message attributes, subprocesses, and deferred activities. You can also use partitioning and purging techniques to address performance issues associated with large quantities of runtime data.

### See Also

*The Oracle Applications Tuning Handbook* by Andy Tremayne and Steve Mayze (Oracle Press, ISBN 0-07-212549-7)

*Oracle Database Concepts*

---

## Designing Workflow Processes for Performance

You can enhance the performance of your workflow processes through effective process design.

### Synchronous, Asynchronous, and Forced Synchronous Workflows

When designing a workflow process, you must decide whether you want it to be executed as a synchronous, asynchronous, or forced synchronous process. The process design impacts the amount of time it takes for the Workflow Engine to return control to the calling application that initiates the process.

- **Synchronous**—A synchronous process contains only activities that can be executed immediately, so that the Workflow Engine executes the process without interruption from start to finish. The Workflow Engine does not return control to the calling application until it completes the process. With a synchronous process, you can immediately check for process results that were written to item attributes or directly to the database. However, the user must wait for the process to complete. If the process takes a long time, the application may appear to hang. In this case, you should change the process to an asynchronous process.
- **Asynchronous**—An asynchronous process is a process that the Workflow Engine cannot complete immediately because it contains activities that interrupt the flow. Examples of activities that force an asynchronous process include deferred activities,

notifications with responses, blocking activities, and wait activities. Rather than waiting indefinitely when it encounters one of these activities, the Workflow Engine sets the audit tables appropriately and returns control to the calling application. The workflow process is left in an unfinished state until it is started again, usually by the Notification System, Business Event System, or the background engine. With an asynchronous process, the user does not have to wait for the process to complete to continue using the application. However, the results of the process are not available until the process is completed at a later time.

- **Forced synchronous**—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. The process results are available immediately upon completion. However, no audit trail is recorded. You may want to use a forced synchronous process if your application needs to generate a specific result quickly and recording an audit trail is not a concern. To create a forced synchronous process, you must set the item key of your process to #SYNCH and follow certain restrictions in designing your process, such as not including any notification activities.

## See Also

Overview of the Workflow Engine: page 8 – 3

Synchronous, Asynchronous, and Forced Synchronous Processes: page 8 – 14

### **Item Attributes**

---

Item attributes act as global variables that can be referenced or updated by any activity within a workflow process. The number of item attributes directly affects the startup time for work items, because the Workflow Engine creates runtime copies of all item attributes when a new work item is created. For this reason, item attributes should be kept to a minimum.

Item attributes should be used for:

- Storing working information for the work item.
- Token replacement for messages. For messages where the number of lines may vary, such as in repeating groups, do not create individual item attributes for each line. Instead, use item

and message attributes of type Document to combine the lines together.

- Storing primary key values so that functions can look up all necessary values from the database.
- Temporary placeholders to set activity attributes dynamically. For example, the performer of a notification may only be known at runtime, so you can reference an item attribute and seed the desired value just before executing the notification.

Item attributes should reference static values or values that are not in the database so that there are no concerns about keeping the values synchronized. (Primary key values, however, do not change.) Do not implement every column within a table as an item attribute.

Item attribute types that can help you reduce the number of attributes you need include the following:

- Document—The attribute value is an attached document, which enables a complex structure to be rendered inline, or attached to notifications. You can specify the following types of documents:
  - PL/SQL document—A document representing data from the database as a character string, generated from a PL/SQL procedure.
  - PL/SQL CLOB document—A document representing data from the database as a character large object (CLOB), generated from a PL/SQL procedure.
- Role—The attribute value is the internal name of a role. If a message attribute of type role is included in a notification message, the attribute automatically resolves to the role's display name, eliminating the need for you to maintain separate attributes for the role's internal and display names. Also, when you view the notification from a web browser, the role display name is a hypertext link to the e-mail address for that role.

Whenever multiple item attributes will be created, or multiple item attribute values will be set during workflow processing, use the array versions of the Add Item Attribute and Set Item Attribute workflow engine APIs (AddItemAttributeArray and SetItemAttributeArray, respectively). These APIs will significantly decrease the number of calls to Workflow Engine APIs, which can have a measurable impact on performance during batch processing. See: AddItemAttributeArray: page 8 – 46 and SetItemAttributeArray: page 8 – 53.

## See Also

Item Type Attributes: page 4 – 2



### **Message Attributes**

---

To enhance performance, message attributes should be kept to a minimum. For messages where the number of lines may vary, such as in repeating groups, do not create individual item and message attributes for each line (LINE\_INFO1, LINE\_INFO2, etc.). Instead, use item and message attributes of type Document to combine the lines together.

### **See Also**

Attribute Types: page 4 – 3

Send and Respond Message Attributes: page 4 – 24

### **Subprocesses**

---

When you design a workflow process, you can group a collection of activities together in a process activity which represents a subprocess within the main process. Using subprocesses judiciously can help make workflow diagrams clearer and easier to read and can simplify workflow monitoring and maintenance. However, subprocesses also result in additional DML operations and additional state information stored in Workflow tables. Consequently, you should avoid unnecessary use of subprocesses when there is no functional benefit.

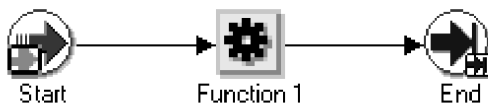
For example, the following two processes, Process 1 and Process 2, are functionally identical, both performing a function called Function 1. However, they result in different numbers of state rows being stored in Workflow tables.

Process 1 contains a Start activity, a Subprocess activity, and an End activity. The subprocess contains a Start activity, the Function 1 activity, and an End activity. This process stores 7 state rows in Workflow tables.

### Process 1

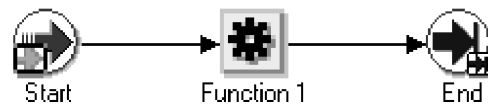


### Subprocess



Process 2 simply contains a Start activity, the Function 1 activity, and an End activity. This process stores only 4 state rows in Workflow tables.

### Process 2



Because more rows are stored in Workflow tables, the kind of design shown in the Process 1 diagram will result in slower workflow throughput and a need to purge Workflow runtime tables more frequently than what should be necessary with the Process 2 design.

**Note:** This guideline is not meant to imply that subprocesses should not be used at all. Collapsing all subprocesses can make workflow diagrams unreadable and difficult to maintain. This recommendation merely highlights that unnecessary overuse of subprocesses can have a negative performance impact.

## See Also

To Create a Process Activity: page 4 – 57

## Deferring Activities

---

The simplest and most effective way to improve the online user's response time is to defer function activities. You may want to defer activities that require a large amount of processing resource or time to complete. Oracle Workflow lets you manage the load on the Workflow Engine and the response time for the user 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.

When an activity is deferred, the main Workflow Engine can then continue to the next available activity, which may occur on some other parallel branch of the process. If no other activity is available to be executed, the Workflow Engine returns control immediately to the calling application. The user remains unaware that processing is still taking place, rendering a faster execution time.

To defer an activity, set the activity's cost above the default threshold cost at design time. The threshold cost is a PL/SQL package variable with a default value of 50 hundredths of a second. Set a cost above this threshold for all activities that you don't want the user to wait for.

At runtime, the Workflow Engine defers any thread to the background as soon as it encounters an activity with a cost higher than the threshold. Then the background engine later identifies the process as deferred and continues its execution.

In addition to deferred activities, background engines also handle timed out activities and stuck processes. You can run as many background engines as you want. You must have at least one background engine that can check for timed out activities, one that can process deferred activities, and one that can handle stuck processes. At a minimum, you need to set up one background engine that can handle both timed out and deferred activities as well as stuck processes.

Generally, you should run a separate background engine to check for stuck processes at less frequent intervals than the background engine that you run for deferred activities, normally not more often than once a day. Run the background engine to check for stuck processes when the load on the system is low.

### See Also

Deferred Processing: page 8 – 9

Setting Up Background Workflow Engines: page 2 – 43

To Set Engine Thresholds: page 2 – 47

---

## Managing Runtime Data for Performance

When the Workflow Engine executes any type of workflow other than forced synchronous processes, status information is stored in runtime tables. The amount of data stored in these tables will grow depending on the complexity and number of workflows being executed.

Performance issues associated with large quantities of runtime data can be addressed by:

- Partitioning
- Purging

---

### Partitioning for Performance

Partitioning addresses key issues in supporting very large tables and indexes by letting you decompose them into smaller and more manageable pieces called partitions. SQL queries and DML statements do not need to be modified in order to access partitioned tables. However, once partitions are defined, DDL statements can access and manipulate individual partitions rather than entire tables or indexes. In this way, partitioning can simplify the manageability of large database objects. Also, partitioning is entirely transparent to applications.

You can optionally run a script to partition certain Workflow tables that store runtime status data. This step is highly recommended for performance gain. Before running the script, you should ensure that you have backed up the tables that will be partitioned and that you have allowed sufficient free space and time for the script to run. For the version of Oracle Workflow embedded in Oracle Applications, the script is called `wfupartb.sql` and is located in the `admin/sql` subdirectory under `$FND_TOP`. For the standalone version of Oracle Workflow, the script is called `wfupart.sql` and is located in the `wf/admin/sql` subdirectory in your Oracle Home. See: [Partitioning Workflow Tables: page 2 – 12](#).

---

### Purging for Performance

You can use the Workflow purge APIs 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. The Workflow purge APIs are defined in the PL/SQL package called `WF_PURGE`.

The availability of runtime data for purging depends on the persistence type of the item type. The persistence type controls how long a status audit trail is maintained for each instance of the item type.

- If you set an item type's Persistence to Permanent, the runtime status information is maintained indefinitely until you specifically purge the information by calling the procedure `WF_PURGE.TotalPerm()`.
- If you set an item type's Persistence to Temporary, you must also specify the number of days of persistence (*'n'*). The status audit trail for each instance of a Temporary item type is maintained for at least *'n'* days of persistence after its completion date. After the *'n'* days of persistence, you can then use any of the `WF_PURGE` APIs to purge the item type's runtime status information. See: `WF_PURGE`: page 8 – 111.
- If you set an item type's Persistence to Synchronous, Oracle Workflow expects instances of that item type to be run as forced synchronous processes with an item key of `#SYNCH`. Forced synchronous processes complete in a single SQL session from start to finish and never insert into or update any database tables. Since no runtime status information is maintained, you do not normally need to perform any purging for a process with the Synchronous persistence type. However, if you run the process with a unique item key in asynchronous mode for testing or debugging purposes, Oracle Workflow does maintain runtime status information for that process instance. You can purge this information by changing the item type's Persistence to Temporary and running any of the `WF_PURGE` APIs. Then change the item type's Persistence back to Synchronous. See: `Synchronous, Asynchronous, and Forced Synchronous Processes`: page 8 – 14.

Additionally, the administration script `wfrmtype.sql` is provided to delete runtime data for a particular item type. This script prompts you for an item type to purge from a list of valid item types, then asks you to choose between deleting all runtime data associated with the specified item type or deleting only runtime data for the completed activities and items of the specified item type. See: `Wfrmtype.sql`: page 16 – 15.

## See Also

`WF_PURGE`: page 8 – 111

Persistence Type: page 4 – 4



# Glossary

**Access Level** A numeric value ranging from 0 to 1000. Every workflow user operates at a specific access level. The access level defines whether the user can modify certain workflow data. You can only modify data that is protected at a level equal to or higher than your access level.

**Activity** A unit of work performed during a business process.

**Activity Attribute** A parameter that has been externalized for a function activity that controls how the function activity operates. You define an activity attribute by displaying the activity's Attributes properties page in the Activities window. You assign a value to an activity attribute by displaying the activity node's Attribute Values properties page in the Process window.

**Agent** A named point of communication within a system.

**Attribute** See Activity Attribute, Item Type Attribute, or Message Attribute.

**Background Engines** A supplemental Workflow Engine that processes deferred or timed out activities.

**Business Event** See Event.

**Cost** A relative value that you can assign to a function or notification activity to inform the Workflow Engine how much processing is required to complete the activity. Assign a higher cost to longer running, complex activities. The Workflow Engine can be set to operate with a threshold cost. Any activity with a cost above the Workflow Engine threshold cost gets set to 'DEFERRED' and is not processed. A background engine can be set up to poll for and process deferred activities.

**Directory Services** A mapping of Oracle Workflow users and roles to a site's directory repository.

**Event** An occurrence in an internet or intranet application or program that might be significant to other objects in a system or to external agents.

**Event Activity** A business event modelled as an activity so that it can be included in a workflow process.

**Event Data** A set of additional details describing an event. The event data can be structured as an XML document. Together, the event name, event key, and event data fully communicate what occurred in the event.

**Event Key** A string that uniquely identifies an instance of an event. Together, the event name, event key, and event data fully communicate what occurred in the event.

**Event Message** A standard Workflow structure for communicating business events, defined by the datatype WF\_EVENT\_T. The event message contains the event data as well as several header properties, including the event name, event key, addressing attributes, and error information.

**Event Subscription** A registration indicating that a particular event is significant to a system and specifying the processing to perform when the triggering event occurs. Subscription processing can include calling custom code, sending the event message to a workflow process, or sending the event message to an agent.

**External Functions** Programs that are executed outside of the Oracle database server.

**External Java Functions** Java programs that are executed outside of the Oracle database server by the Java Function Activity Agent.

**Function** A PL/SQL stored procedure that can define business rules, perform automated tasks within an application, or retrieve application information. The stored procedure accepts standard arguments and returns a completion result.

**Function Activity** An automated unit of work that is defined by a PL/SQL stored procedure.

**Item** A specific process, document, or transaction that is managed by a workflow process. For example, the item managed by the Requisition Approval Process workflow is a specific requisition created by Oracle Internet Commerce's Web Requisitions page.

**Item Attribute** See Item Type Attribute.

**Item Type** A grouping of all items of a particular category that share the same set of item attributes. For example, PO Requisition is an item type used to group all requisitions created by Oracle Internet Commerce's Web Requisitions page. Item type is also used as a high level grouping for processes.

**Item Type Attribute** A feature associated with a particular item type, also known as an item attribute. An item type attribute is defined as a variable whose value can be looked up and set by the application that maintains the item. An item type attribute and its value is available to all activities in a process.

**Lookup Code** An internal name of a value defined in a lookup type.

**Lookup Type** A predefined list of values. Each value in a lookup type has an internal and a display name.

**Message** The information that is sent by a notification activity. A message must be defined before it can be associated with a notification activity. A message contains a subject, a priority, a body, and possibly one or more message attributes.



**Message Attribute** A variable that you define for a particular message to either provide information or prompt for a response when the message is sent in a notification. You can use a predefined item type attribute as a message attribute. Defined as a 'Send' source, a message attribute gets replaced with a runtime value when the message is sent. Defined as a 'Respond' source, a message attribute prompts a user for a response when the message is sent.

**Node** An instance of an activity in a process diagram as shown in the Process window.

**Notification** An instance of a message delivered to a user.

**Notification Activity** A unit of work that requires human intervention. A notification activity sends a message to a user containing the information necessary to complete the work.

**Notification Mailer** A concurrent program that sends e-mail notifications to users via a mail application, and processes e-mail responses.

**Notification Web Page** A Web page that you can view from any Web browser to query and respond to workflow notifications.

**Performer** A user or role assigned to perform a human activity (notification). Notification activities that are included in a process must be assigned to a performer.

**Process** A set of activities that need to be performed to accomplish a business goal.

**Process Definition** A workflow process as defined in Oracle Workflow Builder.

**Process Activity** A process modelled as an activity so that it can be referenced by other processes.

**Protection Level** A numeric value ranging from 0 to 1000 that represents who the data is protected from for modification. When workflow data is defined, it can either be set to customizable (1000), meaning anyone can modify it or it can be assigned a protection level that is equal to the access level of the user defining the data. In the latter case, only users operating at an access level equal to or lower than the data's protection level can modify the data.

**Result Code** The internal name of a result value, as defined by the result type.

**Result Type** The name of the lookup type that contains an activity's possible result values.

**Result Value** The value returned by a completed activity.

**Role** One or more users grouped by a common responsibility or position.

**Subscription** See Event Subscription.

**System** A logically isolated software environment such as a host machine or database instance.

**Timeout** The amount of time during which a notification activity must be performed before the Workflow Engine transitions to an error process or an alternate activity if one is defined.

**Transition** The relationship that defines the completion of one activity and the activation of another activity within a process. In a process diagram, the arrow drawn between two activities represents a transition.

**Workflow Definitions Loader** A concurrent program that lets you upload and download workflow definitions between a flat file and a database.

**Workflow Engine** The Oracle Workflow component that implements a workflow process definition. The Workflow Engine manages the state of all activities for an item, automatically executes functions and sends notifications, maintains a history of completed activities, and detects error conditions and starts error processes. The Workflow Engine is implemented in server PL/SQL and activated when a call to an engine API is made.

# Index

## Symbols

&#NID, 4-12, 4-13, 4-15, 4-32

#FROM\_ROLE attribute, 4-25

#HIDE\_REASSIGN attribute, 4-25

## A

AbortProcess(), 8-36

Access Level, 2-102  
default, 2-105

Access level indicator, 4-17

Access property page, 4-17

Access protection

*See also* Access level; Protection level  
preserving customizations, 4-18

AccessCheck(), 8-230

ACCOUNT parameter, 2-59

Acknowledge Ping event, 14-8

ACTID, 7-5, 7-14

Actions, for subscriptions, 13-37

Activities, 3-10, 4-42

accessing from different data stores, 5-7, 6-2

Concurrent Manager, 6-22

copy, 4-60

cost, 4-47

create, 4-48, 4-50, 4-54, 4-57

deferred, 4-47

effective date, 4-59

error process, 4-59

event, 4-42, 4-45

External Java functions, 2-86

for an error process, 6-26

function, 4-42, 4-44

icons, 2-85, 4-49, 4-52, 4-54, 4-58, 4-62

in a loop, 4-60

in the Buyer: Advanced Shipment Notice  
process, 15-84

in the Buyer: Receive Supplier Invoicing  
process, 15-86

in the Buyer: Receive Supplier PO  
Acknowledgement process, 15-81

in the Buyer: Send PO to Supplier process,  
15-78

in the Buyer: Top Level PO process, 15-75

in the Detail Ping process, 13-82

in the Master Ping process, 13-80

in the Notify Approver subprocess, 15-21

in the Requisition process, 15-15

in the Supplier: Advanced Shipment Notice  
process, 15-99

in the Supplier: Credit Check process, 15-95  
in the Supplier: Get Order Details process,  
15-92

in the Supplier: Send Supplier Invoice  
process, 15-101

in the Supplier: Stock Check process, 15-97

in the Supplier: Top Level Order process,  
15-88

in the Workflow Event Protocol process,  
14-21

joining branches, 5-4

notification, 4-42, 4-43

optional details, 4-59

process, 4-42, 4-46

processing cost, 8-9

result type, 4-48, 4-52, 4-58

- Standard, 4–42, 6–2
- statuses, 8–3
- System: Error, 4–42
- timing out, 5–10
- version number, 4–60
- Activities( ), 8–114
- Activity attributes
  - See also* Function activity attributes
  - setting values for, 5–12
- Activity nodes
  - in the Buyer: Advanced Shipment Notice process, 15–84
  - in the Buyer: Receive Supplier Invoicing process, 15–86
  - in the Buyer: Receive Supplier PO Acknowledgement process, 15–81
  - in the Buyer: Send PO to Supplier process, 15–78
  - in the Buyer: Top Level PO process, 15–75
  - in the Detail Ping process, 13–82
  - in the Master Ping process, 13–80
  - in the Notify Approver subprocess, 15–21
  - in the Requisition process, 15–15
  - in the Supplier: Advanced Shipment Notice process, 15–99
  - in the Supplier: Credit Check process, 15–95
  - in the Supplier: Get Order Details process, 15–92
  - in the Supplier: Send Supplier Invoice process, 15–101
  - in the Supplier: Stock Check process, 15–97
  - in the Supplier: Top Level Order process, 15–88
  - in the Workflow Event Protocol process, 14–21
- Ad hoc users and roles, 5–24
  - APIs, 8–121
- AddAttr( ), 8–216
- AddCorrelation( ), 8–294
- AddItemAttr( ), 8–43
- addItemAttrDate( ), 8–43
- AddItemAttrDateArray( ), 8–46
- addItemAttrNumber( ), 8–43
- AddItemAttrNumberArray( ), 8–46
- addItemAttrText( ), 8–43
- AddItemAttrTextArray( ), 8–46
- AddParameterToList, 8–253
- AddParameterToList( ), 8–275
- AddParameterToListPos( ), 8–276
- Address, 8–253
- AddUsersToAdHocRole( ), 8–138
- AdHocDirectory( ), 8–118
- Administrator privileges, 2–16
- Advanced Queues integration, 8–162
- Advanced Queuing, 13–2
- Agent, datatype, 8–237
- Agent Created event, 14–4
- Agent Deleted event, 14–5
- Agent Updated event, 14–4
- Agents, 13–22
  - defining, 13–29
  - deleting, 13–33
  - direction, 13–22
  - finding, 13–32
  - pinging, 13–77
  - protocol, 13–23
  - queue handlers, 13–25
  - queues, 13–24
  - scheduling listeners, 13–56
  - scheduling propagations, 13–61
  - updating, 13–33
- Agents web page, 13–29, 13–33
- ALLOW\_FORWARDED\_RESPONSE
  - parameter, 2–61
- And activity, 6–2
- Any event, 14–10
- Any transitions, 5–2
- APIs, 8–3
- AQ message payload, 8–163
- Arrows, 5–2
- Assign activity, 6–14
- AssignActivity( ), 8–74
- Asynchronous processes, 8–14, C – 2
- Attribute, token substitution, 4–41
- Attribute types
  - attribute, 4–11
  - date, 4–10, 4–35
  - document, 4–11, 4–14, 4–36

- event, 4–11, 4–36
- form, 4–10, 4–13, 4–36
- lookup, 4–10, 4–35
- number, 4–10, 4–35
- role, 4–11, 4–36
- text, 4–10, 4–35
- URL, 4–10, 4–12, 4–35
- Attribute–type attributes, 4–4
- Attributes
  - copy, 4–16
  - type, 4–3, 4–10, 4–35
- AUTOCLOSE\_FYI parameter, 2–61
- Automatic Notification Handler, 10–25
- Automatic replication, of Event Manager objects, 13–71
- Automatic responses, 10–25
- Automatic routing, 10–25

## B

- B2B Advanced Shipment Notice event, 15–106
- B2B Invoice event, 15–107
- B2B Purchase Order Acknowledgement event, 15–105
- B2B Purchase Order event, 15–102
- Background engine, scripts, 16–7
- Background Engines
  - about, 2–43
  - scripts, 16–6
  - starting, 2–44
  - submitting, 2–45
- Background( ), 8–41
- BeginActivity( ), 8–67
- Block activity, 6–5
- Business Event System, 1–3
  - checking setup, 13–53
  - managing business events, 13–2
  - overview, 8–235
  - Ping/Acknowledge example, 13–77
  - predefined events, 14–2
  - setting up, 2–96
- Business Event System Replication APIs, 8–300

- Business events, 13–4
  - in Workflow processes, 8–17
- Buyer Workbench, web page, 15–67
- Buyer: Advanced Shipment Notice process, summary, 15–83
- Buyer: Receive Supplier Invoicing process, summary, 15–85
- Buyer: Receive Supplier PO Acknowledgement process, summary, 15–80
- Buyer: Send PO to Supplier process, summary, 15–78
- Buyer: Top Level PO process, summary, 15–73

## C

- Callback functions, 7–13
  - command, 7–15
  - for item types, 4–5
- Cancel( ), 8–209
- CancelGroup( ), 8–210
- Check Setup web page, 13–53, 13–58, 13–62
- Checking
  - activity versions, 16–17
  - background engines, 16–7
  - directory service data model, 16–10
  - foreign/primary key references, 16–13
  - workflow data model, 16–16
- CLEAR( ), 8–102
- ClearMsgStack( ), 8–180
- Close( ), 8–215
- Compare Date activity, 6–3
- Compare Event Property activity, 6–16
- Compare Execution Time activity, 6–3
- Compare Number activity, 6–3
- Compare Text activity, 6–3
- compareTo( ), 8–100
- Comparison activities, 6–3
- CompleteActivity( ), 8–69
- CompleteActivityInternalName( ), 8–72
- Concurrent Manager activities, 6–22
- Concurrent Manager Functions item type, 6–22
- Concurrent program
  - FNDWFLST, 16–4

- FNDWFPR, 16–5
- Concurrent programs
  - Notification Mailer, 2–48, 2–56
  - Purge Obsolete Workflow Runtime Data, 8–119
  - Workflow Agent Listener, 8–272
  - Workflow Background Process, 2–45
  - Workflow Definitions Loader, 2–109
  - Workflow Resource Generator, 8–105
- CONNECT parameter, 2–58
- Constants, WFAttribute class, 8–90
- Content, 8–252
- Content–attached checkbox, 4–37
- CONTEXT(), 8–108
- Continue Flow activity, 6–12
- Coordinating master/detail activities, 6–11
- Cost threshold, 4–47
- CreateAdHocRole(), 8–136
- CreateAdHocUser(), 8–134
- CreateForkProcess(), 8–38
- CreateMsg(), 8–181
- CreateProcess(), 8–21
- Custom logos, in web pages, 2–84
- Customization Level, 2–105
  - for activities, 4–8, 4–11, 4–20, 4–32, 4–50, 4–53, 4–56, 4–58, 4–63

## D

- Data types, wf\_payload\_t, 8–163
- Database links
  - checking, 13–55
  - creating, 2–96
- Datatypes
  - example, 8–255
  - for the Business Event System, 8–236
  - WF\_AGENT\_T, 8–237
  - WF\_EVENT\_T, 8–242
  - WF\_PARAMETER\_LIST\_T, 8–241
  - WF\_PARAMETER\_T, 8–239
- Date–type attributes, 4–3
- DBA Studio, 2–96
- DEBUG parameter, 2–62

- Default Error Process, 6–28
- Default Event Error Process, 6–34
- Default transitions, 5–2
- DEFAULT\_ERROR, 6–28
- DEFAULT\_EVENT\_ERROR, 6–34
- Default\_Rule(), 8–281
- Defer Thread activity, 6–6
- Deferred activities, 2–43, 4–47
  - performance, C – 7
- Deferred processing
  - for event subscriptions, 13–41
  - for workflow processes, 2–43, 8–9, C – 7
- DeferredQueue function, 8–177
- Delete
  - all workflow data, 16–14
  - data for an item type, 16–15
  - item type attributes, 16–14
  - runtime data for an item type, 16–15, C – 9
  - workflow status information, 16–15
- Demonstration, directory service, 15–7
- Dequeue, queue handler, 7–24
- DequeueEventDetail(), 8–170
- DequeueException(), 8–176
- DequeueOutbound(), 8–167
- Detail Notification web page, 10–19
- Detail Ping process, summary, 13–81
- Detail process, 6–12
- Detail Survey process
  - activities, 15–47
  - summary, 15–46
- Diagram arrows, 5–2
- Direct Response e–mail, 10–3
- DIRECT\_RESPONSE parameter, 2–60
- Directory repository, 2–21
- Directory Service
  - in Navigator tree, 3–4
  - view from Builder, 5–26
- Directory services, 2–21
  - checking the data model, 2–27, 16–10
  - integrating with local workflow users, 2–28
  - integrating with native Oracle users, 2–27
  - integrating with Oracle HR, 2–27
  - synchronization, 2–30, 8–144

- Directory Services APIs, 8–121
- DISCARD parameter, 2–63
- Dispatch mode, 13–43
- Document integration, 4–3, 4–11, 4–36, 7–17
- Document Management, item type, 15–49
- Document Management APIs, 8–185
- Document management integration, 4–3, 4–6
- Document message attributes, attached vs embedded, 4–37
- Document Review process, 15–49
  - activities, 15–52
  - summary, 15–50
- Document Type Definitions
  - Business Event System, 8–300
  - WF\_AGENTS, 8–311
  - WF\_EVENT\_GROUPS, 8–305
  - WF\_EVENT\_SUBSCRIPTIONS, 8–314
  - WF\_EVENTS, 8–302
  - WF\_SYSTEMS, 8–308
- Document-type attributes, 4–3
- Documents, 4–6
- Dynamic priority, 5–11
- Dynamic timeouts, 5–10

## E

- E-mail notifications, 1–5, 2–48
  - and HTML attachments, 2–4
  - example direct response instructions, 10–7
  - modifying mail templates, 2–69
  - requirements, 2–4
  - summaries, 10–24
  - templates for, 2–48, 10–3
  - with HTML attachments, 10–2
- Edit menu, A – 3
- Effective date, 3–16
- Effective dates, 3–14, 3–16, 4–59, 8–11
- Effectivity, dates of, 3–7
- END activities, 5–4
- End Activity, 6–8
- Engine thresholds, 2–47
- Enqueue, queue handler, 7–23

- Enqueue( ), 8–269
- EnqueueInbound( ), 8–165
- Environment variables
  - WF\_ACCESS\_LEVEL, 2–102, 2–106
  - WF\_RESOURCES, 2–42
- Error activities, 6–26
- Error Check process, 15–54
  - activities, 15–57
  - summary, 15–56
- Error handling
  - for event subscriptions, 13–44
  - for process activities, 8–77
  - for workflow processes, 8–10
- Error process, 4–59, 6–26
- Error( ), 8–284
- Error\_Rule( ), 8–288
- Errored activities, retrying, 16–13
- Event activities, 4–45
  - create, 4–54
  - Workflow Engine, 8–17
- Event activity attributes, 4–56
- Event activity details, 5–12
- Event APIs, 8–260
- Event Created event, 14–2
- Event data, 7–21, 13–5, 13–36
- Event data URL, 8–50
- Event Deleted event, 14–3
- Event Function APIs, 8–290
- Event Group Creation event, 14–3
- Event Group Deleted event, 14–3
- Event Group Updated event, 14–3
- Event groups, 13–4
  - defining, 13–8
- Event Manager, 13–3
- Event messages
  - datatype, 8–242
  - enqueueing, 16–10
- Event nodes, 5–12
- Event Rule APIs, 8–279
- Event subscriptions, 13–34
  - rule functions, 7–25
- Event Subscriptions web page, 13–52
- Event System Demonstration
  - data model, 15–64

- initiating, 15–66
- overview, 15–63
- setting up, 15–66
- Event System Demonstration process,
  - installing, 15–64
- Event System Local Queues web page, 13–73
- Event Updated event, 14–2
- Event(), 8–75
- Event-type attributes, 4–4
- Events, 13–4
  - defining, 13–5
  - deleting, 13–15
  - finding, 13–14
  - predefined, 14–2
  - raising, 13–4, 13–65
  - sending to agents, 13–39
  - sending to workflow processes, 13–38
  - updating, 13–15
- Events web page, 13–5, 13–8, 13–15, 13–45
- Events: Buyer Workbench, web page, 15–67
- Events: Track Order, web page, 15–69
- Example function activity
  - Select Approver, 15–26
  - Verify Authority, 15–29
- Example process
  - Event System Demonstration, 15–63
  - Requisition, 15–5
- Execute Concurrent Program activity, 6–22
- execute(), 8–89
- External document integration, 4–6
- External Java function activities, 2–86, 8–5, 8–82

## F

- FAILCOMMAND parameter, 2–62
- File menu, A – 2
- Find Agent web page, 13–32
- Find Event web page, 13–14, 13–50
- Find Notifications web page, 10–15
- Find System web page, 13–19
- FND\_FNDWFAS, 11–8

- FND\_FNDWFNOT, 10–14
- FNDWFLST, 8–272
  - concurrent program, 16–4
- FNDWFPR, 8–119
  - concurrent program, 16–5
- Fonts
  - modifying, 5–21
  - setting, 5–21
- Forced synchronous processes, 8–14, C – 2
- Form-type attributes, 4–3
- FORWARD mode, 8–13
- Forward(), 8–194, 8–205
- Frame target, URL attributes, 4–37
- FROM parameter, 2–59
- FROM\_ROLE attribute, 4–25
- FUNCMODE, 7–5, 7–6
- Function activities, 4–44
  - create, 4–50
  - standard Java API, 7–8
  - standard PL/SQL API, 7–3
- Function activity attributes, 4–8, 4–53
- Functions, 3–10
  - See also* PL/SQL procedures
- Future-dated events, 13–41

## G

- Generate function, 13–5
- Generate()
  - WF\_AGENTS\_PKG, 8–312
  - WF\_EVENT\_FUNCTIONS\_PKG, 8–296
  - WF\_EVENT\_GROUPS\_PKG, 8–306
  - WF\_EVENT\_SUBSCRIPTIONS\_PKG, 8–315
  - WF\_EVENTS\_PKG, 8–303
  - WF\_SYSTEMS\_PKG, 8–309
- Get Event Property activity, 6–15
- Get Monitor URL activity, 6–14
- GET\_ERROR(), 8–103
- get\_launch\_attach\_url(), 8–187
- get\_launch\_document\_url(), 8–186
- get\_open\_dm\_select\_window(), 8–188, 8–189
- get\_pref(), 8–148
- GetAccessKey(), 8–150



- getActivityAttr(), 8–85
- GetActivityAttrClob(), 8–66
- GetActivityAttrDate(), 8–64
- GetActivityAttrEvent(), 8–64
- GetActivityAttrInfo(), 8–63
- GetActivityAttrNumber(), 8–64
- GetActivityAttrText(), 8–64
- GetActivityLabel(), 8–25
- GetAdvancedEnvelopeURL(), 8–155
- GetAttrDate(), 8–223
- GetAttrDoc(), 8–225
- GetAttrInfo(), 8–219
- GetAttrNumber(), 8–223
- GetAttrText(), 8–223
- GetBody(), 8–227
- getCorrelationID, 8–246
- GetDiagramURL(), 8–151
- GetEnvelopeURL(), 8–153
- getErrorMessage, 8–248
- getErrorStack, 8–248
- getErrorSubscription, 8–247
- getEventData, 8–247
- getEventKey, 8–247
- getEventName, 8–246
- getFormat(), 8–97
- getFromAgent, 8–247
- GetInfo(), 8–220
- getItemAttr(), 8–87
- GetItemAttrClob(), 8–60
- GetItemAttrDate(), 8–57
- GetItemAttrDocument(), 8–59
- GetItemAttrEvent(), 8–57
- getItemAttributes(), 8–61
- GetItemAttrInfo(), 8–62
- GetItemAttrNumber(), 8–57
- GetItemAttrText(), 8–57
- getItemTypes(), 8–56
- GetItemUserKey(), 8–24
- GetMessageHandle(), 8–175
- getName
  - WF\_AGENT\_T, 8–237

- WF\_PARAMETER\_T, 8–239
- WFAttribute, 8–94
- getNotificationAttributes(), 8–233
- getNotifications(), 8–232
- getParameterList, 8–246
- getPriority, 8–245
- getProcessStatus(), 8–81
- getReceiveDate, 8–246
- GetRoleDisplayName(), 8–131
- GetRoleInfo(), 8–125
- GetRoleInfo2(), 8–126
- GetRoleName(), 8–130
- GetRoleUsers(), 8–123
- getSendDate, 8–245
- GetShortBody(), 8–228
- GetShortText(), 8–222
- GetSubject(), 8–226
- getSystem, 8–237
- GetText(), 8–221
- getToAgent, 8–247
- getType(), 8–96
- GetUserName(), 8–129
- GetUserRoles(), 8–124
- getValue
  - WF\_PARAMETER\_T, 8–239
  - WFAttribute, 8–95
- GetValueForParameter, 8–253
- GetValueForParameter(), 8–277
- GetValueForParameterPos(), 8–278
- getValueType(), 8–98
- Global Preferences, web page, 2–14
- Global variables, 4–2

## H

- HandleError(), 8–77
- Hardware requirements, 2–2
- Help menu, A – 6
- Hidden item types, 3–4
- HIDE\_REASSIGN attribute, 4–25
- Home page, 9–2
- HTML\_MAIL\_TEMPLATE parameter, 2–65

HTMLAGENT parameter, 2–62

## I

Icons, 2–85

viewing, 4–49, 4–52, 4–54, 4–58

IDLE parameter, 2–61

InboundQueue function, 8–178

Init.ora parameters, 13–54

Initialize, 8–245

Initiating a workflow process, 15–8, 15–36,  
15–66

Internal names

updating activity, 16–7

updating activity attributes, 16–7

updating item attributes, 16–8

updating item types, 16–8

updating lookup codes, 16–8

updating lookup types, 16–9

updating message attributes, 16–9

updating messages, 16–9

IsPerformer(), 8–127

Item attributes, external document integration,  
4–6

Item type attributes, 4–2, 4–8, 8–12

arrays, 8–13

Event System Demonstration, 15–71

performance, C – 3

Requisition, 15–12

Workflow Send Protocol, 14–18

Item types, 3–9, 4–2

callback function, 4–5

Concurrent Manager Functions, 6–22

context reset, 7–13

copy, 4–15

creation, 4–7

Event System Demonstration, 15–71

loading, 3–12, 3–13

persistence type, 4–4, C – 8

Requisition, 15–12

saving, 3–12

selector functions, 4–5, 7–13

Standard, 6–2

System: Error, 6–26

System: Mailer, 2–69

Workflow Agent Ping/Acknowledge, 13–78

Workflow Send Protocol, 14–18

ITEMKEY, 7–4, 7–14

Items(), 8–113

ItemStatus(), 8–80

ITEMTYPE, 7–4, 7–14

## J

Java API, for function activities, 7–8

Java APIs, 8–5

Java Function Activity Agent, 2–86

starting, 2–86

stopping, 2–95, 16–11

Java interface, 8–5

Java monitor tool, 11–2

Java Runtime Environment, 2–5

JavaScript, support in a Web browser, 2–4

Joining activities, 5–4

## L

Launch Process activity, 6–6

LaunchProcess(), 8–30

LDAP, 2–30

LDAP APIs, 2–34, 8–144

List of values, in a web interface, 10–24, 13–22

Listen(), 8–270

Listeners

deleting, 13–61

for inbound agents, 13–56

running, 16–6

scheduling, 13–58

updating, 13–61

Load balancing, 6–9

loadActivityAttributes(), 8–84

Loader program. *See* Workflow Definitions  
Loader

Loading item types, 3–13

loadItemAttributes(), 8–83

Local system, 13–17

LOG parameter, 2–62

- Log( ), 8–283
- Login Server, 2–32
- Lookup codes, copy, 4–22
- Lookup types, 3–9, 4–19
  - copy, 4–22
  - creation, 4–20
- Lookup–type attributes, 4–3
- Loop Counter activity, 6–7
- Loop Reset, 5–3
- Loops, 4–60, 7–6, 8–10

## M

- MAPI-compliant mail application, 2–55
- Master Ping Process, summary, 13–79
- Master process, 6–12
- Master/copy systems, 13–72
- Master/Detail coordination activities, 6–11
  - notes on usage, 6–13
- Menus, Oracle Workflow Builder, A – 2
- Message attributes, 4–23, 4–24, 4–33, 4–34, 15–32
  - #FROM\_ROLE, 4–25
  - #HIDE\_REASSIGN, 4–25
  - for Workflow Cancelled Mail message, 2–78
  - for Workflow Closed Mail message, 2–81
  - for Workflow Invalid Mail message, 2–79
  - for Workflow Open FYI Mail message, 2–77
  - for Workflow Open Mail (Direct) message, 2–73
  - for Workflow Open Mail (Templated) message, 2–70
  - for Workflow Open Mail for Outlook Express message, 2–75
  - for Workflow Summary Mail message, 2–82
  - for Workflow URL Attachment message, 2–77
  - for Workflow Warning Mail message, 2–82
- formatted table, 4–26
- performance, C – 5
- Respond, 4–25, 4–35, 4–39
- Send, 4–24, 4–35
- source, 4–24, 4–35

- Message function, WF\_NOTIFICATION(), 4–26
- Message propagation, setting up, 13–53
- Message templates, for e–mail notifications, 2–69
- Messages, 3–9
  - body, 4–31, 15–31
  - copy, 4–41
  - creation, 4–29
  - overriding default priority, 5–11
  - subject, 4–30, 15–31
  - viewing, 15–32
- Messages window, 4–23
- MIME support, 2–49
- Mod\_osso, 2–32
- Monitoring
  - Workflow Monitor, 11–2
  - workitems, 1–5
- Multi-consumer queues, 13–40
- Multilingual support, 16–5, 16–12

## N

- Naming conventions, PL/SQL stored procedures, 15–15
- Navigator Toolbar, A – 7
- Navigator tree, finding objects in, 3–6
- NewAgent( ), 8–267
- NLS codeset, 2–65
- NLS support
  - in a web session, 2–39
  - in e–mail notifications, 2–39
  - in Oracle Workflow Builder, 2–38
- Node activities, dynamic priority, 5–11
- NODE parameter, 2–59
- Nodes
  - adding to a process, 5–6
  - start and end, 5–8
- NOOP activity, 6–7
- Notification, status, 16–12
- Notification access keys, 10–3
- Notification activities, 4–43
  - coupling with custom functions, 4–49, 8–13
  - create, 4–48

- Notify Requisition Approval Required, 15–31
  - Notification APIs, 8–192, 8–197
  - Notification functions, 4–49, 8–13
  - Notification history, 4–26
  - Notification ID token, 4–12, 4–13, 4–15, 4–32
  - Notification IDs, 10–3
  - Notification Mailer
    - about, 2–48
    - configuration file, 2–58
    - MIME support, 2–49
    - notification preference, 2–49
    - required folders, 2–63
    - response processing, 2–67
    - script to restart, 2–67
    - shutdown, 2–48
    - starting, 2–56
    - starting for MAPI-compliant applications, 2–57
    - starting for UNIX Sendmail, 2–55
  - Notification method, 10–2
  - Notification preference, 9–8
  - Notification preferences, 2–20, 2–49
  - Notification summaries, via e-mail, 10–24
  - Notification System, 2–48, 8–192
  - Notification templates, for e-mail notifications, 2–69
  - Notification Web page, 1–5
    - reassigning notifications, 10–22
  - Notifications, 10–2
    - dependence on directory services, 10–2
    - forwarding, 8–194
    - hiding the Reassign button, 4–25
    - HTML-formatted e-mail, 10–9
    - identifying the responder, 8–211
    - load balancing, 6–9
    - plain text e-mail using direct response, 10–6
    - plain text e-mail using templated response, 10–5
    - plain text e-mail with attachments, 10–11
    - reassign in Notification Web page, 10–22
    - reassign via e-mail, 10–12
    - responding with Notification Web page, 10–22
    - setting the From Role, 4–25
    - timed out, 8–195
    - transferring, 8–195
    - via e-mail, 2–48, 10–2
    - via Notification Web page, 10–13
  - Notifications Worklist. *See* Worklist web page
  - Notifications(), 8–115
  - Notify activity, 6–9
  - Notify Approver, example notification activities, 15–31
  - Notify Approver subprocess, summary, 15–19
  - Notify Requisition Approval Required, 15–31
  - Number-type attributes, 4–3
- ## O
- OMBAQ\_TEXT\_MSG, 8–257
  - On Revisit, 8–11
  - OpenNotificationsExist(), 8–214
  - Or activity, 6–2
  - Oracle Advanced Queues integration, 8–162
  - Oracle Advanced Queuing, 13–2
  - Oracle Applications Manager, 1–5
  - Oracle DBA Studio, 2–96
  - Oracle HTTP Server, 2–32
    - identifying the Workflow web agent, 2–17
    - Workflow server requirements, 2–4
  - Oracle Internet Directory, 2–30
  - Oracle Message Broker, 2–100
  - Oracle Net Services, 2–2
  - Oracle Workflow, implementation issues, 2–6
  - Oracle Workflow Builder, 1–3
    - Loader functionality, 3–15
    - overview, 3–2
    - requirements, 2–2
    - save modes, 3–15, 4–17
    - starting from command line, 3–17
  - Oracle Workflow home page, 9–2
  - Oracle Workflow Manager, 1–5
  - Oracle Workflow views, 8–157
  - Oracle9i Application Server, Workflow server requirements, 2–4
  - Oracle9iAS Single Sign-On, 2–32

OutboundQueue function, 8–179

## P

Parameter, datatype, 8–239

Parameter list, datatype, 8–241

Parameters( ), 8–291

Partitioning Workflow tables, 2–12, C – 8

Payload, for Advanced Queues messages,  
8–163

Performance

concepts, C – 2

deferred activities, C – 7

item attributes, C – 3

message attributes, C – 5

partitioning Workflow tables, C – 8

purging, C – 8

subprocesses, C – 5

synchronous and asynchronous workflows,  
C – 2

Periodic Alert, item type, 15–54

Persistence, 4–4, C – 8

Phase numbers, 13–36, 13–42

Ping Agent event, 14–8

Pinging agents, 13–77

PL/SQL, 1–4

document, 7–17

PL/SQL APIs

for a 'PL/SQL CLOB' document, 7–17

for a 'PL/SQL' document, 7–17

for a Queue Handler, 7–23

for a selector or callback function, 7–13

for an Event Data Generate Function, 7–21

for an Event Subscription Rule Function,  
7–25

for function activities, 7–3

PL/SQL CLOB, document, 7–17, 7–19

PL/SQL documents, 4–6

PL/SQL stored procedures

creating, 15–15

naming conventions, 15–15

scripts, 15–15

Post-notification functions, 4–43, 8–13

Predefined events, 14–2

Preferred notification method, 10–2

Preserving customizations, for an activity, 4–18

Process activities, 4–46

create, 4–57

Process definition, modifying, 3–11

Process diagram

adding nodes, 5–6

drawing, 5–2, 5–6

PROCESS parameter, 2–63

Process rollback, 8–77

Process window, 5–2

editing, 5–2

Process Window Toolbar, A – 8

Processes

activity transitions, 5–2

copying to clipboard, 5–20

creation, 3–7

editing, 3–10, 3–12

loops, 7–6, 8–10

overview, 5–19

printing, 5–20

starting, 5–4

verify, 5–21

ProcessInboundQueue( ), 8–174

Product Survey, web page, 15–36

Product Survey item type, 15–38

Product Survey process, 15–34

initiating, 15–36

installing, 15–35

Propagation, setting up, 13–53

Propagations

deleting, 13–65

for outbound agents, 13–61

scheduling, 13–62

updating, 13–65

Protection level, 2–103

reset, 16–12

Protection level locking. *See* Access protection

Protocols, 13–23

Purge

outbound notification message queue, 16–12

performance, C – 8

runtime data, 16–5

- Workflow Purge APIs, 8–111
- Purge Obsolete Workflow Runtime Data concurrent program, 8–119
- PurgeEvent(), 8–172
- PurgeItemType(), 8–173

## Q

- Queue handlers, 7–23, 13–25
  - WF\_EVENT\_OMB\_QH, 2–100
- Queue tables, 2–97
- Queues
  - assigned to agents, 13–24
  - checking, 13–56
  - reviewing, 13–72
  - setting up, 2–97

## R

- Raise Event web page, 13–66
- RAISE(), 8–105
- Raise(), 8–261
- Raising events, 13–4, 13–65
- Reassign notifications
  - hiding the Reassign button, 4–25
  - in Notification Web page, 10–22
  - via e-mail, 10–12
- Reassign web page, 10–22
- Receive date, for event messages, 8–270
- Receive()
  - WF\_AGENTS\_PKG, 8–313
  - WF\_EVENT\_FUNCTIONS\_PKG, 8–298
  - WF\_EVENT\_GROUPS\_PKG, 8–307
  - WF\_EVENT\_SUBSCRIPTIONS\_PKG, 8–316
  - WF\_EVENTS\_PKG, 8–304
  - WF\_SYSTEMS\_PKG, 8–310
- RemoveUsersFromAdHocRole, 8–143
- Replication APIs, Business Event System, 8–300
- REPLYTO parameter, 2–62
- Requirements, hardware and software, 2–2
- Requisition, data model, 15–6

- Requisition Demonstration, web page, 15–8
- Requisition process, 15–5
  - example function activities, 15–26
  - initiating, 15–8
  - installing, 15–6
  - summary, 15–13
- Reset process. *See* Rollback
- RESET\_FAILED parameter, 2–65
- RESET-NLS parameter, 2–65
- Respond attributes, 2–70, 2–72, 2–75, 2–79
- RESPOND mode, 8–13
- Respond to notification
  - HTML-formatted e-mail, 10–9
  - plain text e-mail using direct response, 10–6
  - plain text e-mail using templated response, 10–5
  - plain text e-mail with attachments, 10–11
  - via Notification Web page, 10–13
- Respond(), 8–194, 8–211
- Responder, 8–211
- Responder(), 8–212
- Response methods, direct vs. templated, 2–60
- Response processing, by Notification Mailer, 2–67
- Responses, processing, 8–194
- RESULT, 7–5, 7–15
- Result type
  - for activities, 4–48, 4–52, 4–58
  - for voting activities, 4–62
- ResumeProcess(), 8–34
- Retry Error, 6–32
- RETRY\_ONLY, 6–32
- Role
  - administrator, 2–16
  - property page, 5–26
- Role Resolution activity, 6–9
- Role-type attributes, 4–4
- Roles, 5–24
  - ad hoc, 5–24
  - loading into the Workflow Builder, 5–25
  - tab page, 5–24
  - view from Builder, 5–26
- Rollback, of process, 8–77
- Routing, automatic, 10–25

- Routing rules
  - deleting, 10–32
  - for a role, 10–27
  - listing, 10–26
  - overriding, 10–31
  - updating, 10–32

- Rule functions, 7–25
  - for event subscriptions, 13–37

- Runtime data, C – 8

## S

- Sample workflow processes, 15–2

- Savepoints, 7–3, 7–8, 8–4

- Schedule\_changes(), 8–147

- Seed event group, 14–6

- Select Approver function activity, 15–26

- Selector functions, 4–5, 7–13

- Send date, for event messages, 8–265

- Send(), 8–192, 8–199, 8–265

- SEND\_ACCESS\_KEY parameter, 2–66

- SendGroup(), 8–192, 8–203

- Set Event Property activity, 6–15

- set\_document\_id\_html(), 8–190

- SetAdHocRoleAttr(), 8–142

- SetAdHocRoleExpiration(), 8–140

- SetAdHocRoleStatus(), 8–133

- SetAdHocUserAttr(), 8–141

- SetAdHocUserExpiration(), 8–139

- SetAdHocUserStatus(), 8–132

- SetAttrDate(), 8–217

- SetAttrNumber(), 8–217

- SetAttrText(), 8–217

- setCorrelationID, 8–249

- SetDispatchMode(), 8–274

- SetErrorInfo(), 8–273

- setErrorMessage, 8–251

- setErrorStack, 8–252

- setErrorSubscription, 8–251

- setEventData, 8–250

- setEventKey, 8–250

- setEventName, 8–250

- setFromAgent, 8–251

- SetItemAttrDate(), 8–48

- SetItemAttrDateArray(), 8–53

- SetItemAttrDocument(), 8–51

- SetItemAttrEvent(), 8–48

- SetItemAttrNumber(), 8–48

- SetItemAttrNumberArray(), 8–53

- SetItemAttrText(), 8–48

- SetItemAttrTextArray(), 8–53

- setItemAttrValue(), 8–88

- SetItemOwner(), 8–26

- SetItemParent API, 6–12

- SetItemParent(), 8–79

- SetItemUserKey(), 8–23

- SetMsgAttr(), 8–183

- SetMsgResult(), 8–184

- setName

  - WF\_AGENT\_T, 8–238

  - WF\_PARAMETER\_T, 8–240

- setParameterList, 8–249

- SetParametersIntoParameterList(), 8–289

- setPriority, 8–248

- setReceiveDate, 8–249

- setSendDate, 8–248

- setSystem, 8–238

- setToAgent, 8–251

- setValue, 8–240

- Shortcuts, 5–22

- Shutdown files, 2–62

- SHUTDOWN parameter, 2–62

- Single sign-on, 2–30, 2–32

- Single-consumer queues, 13–40

- Software requirements, 2–2

- Source types, 13–35

- Standard activities, 6–2

- Standard APIs

  - for “PL/SQL CLOB” documents, 7–17, 7–19

  - for “PL/SQL” documents, 7–17

  - for a Queue Handler, 7–23

- for an Event Data Generate Function, 7–21
- for an Event Subscription Rule Function, 7–25
- for function activities, 7–3, 7–8
- for selector/callback functions, 7–13
- Standard error process, 6–26
- Standard item type, 6–2
- START activities, 5–4
- Start activity, 6–8
- StartForkProcess(), 8–40
- StartProcess function, for sample Requisition process, 15–23
- StartProcess(), 8–28
- Status report
  - developer, 16–16
  - end user, 16–16
- Stuck processes, 2–43
- Submit Concurrent Program activity, 6–23
- Subprocesses
  - performance, C – 5
  - timing out, 5–10
- Subscription Created event, 14–5
- Subscription Deleted event, 14–5
- Subscription Updated event, 14–5
- SubscriptionParameters(), 8–293
- Subscriptions, 13–34
  - deferring, 13–41
  - defining, 13–45
  - deleting, 13–52
  - finding, 13–50
  - predefined, 14–2
  - updating, 13–52
- Success(), 8–286
- SUMMARYONLY parameter, 2–59
- Supplier: Advanced Shipment Notice process, summary, 15–98
- Supplier: Credit Check process, summary, 15–94
- Supplier: Get Order Details process, summary, 15–91
- Supplier: Send Supplier Invoice process, summary, 15–100
- Supplier: Stock Check process, summary, 15–96

- Supplier: Top Level Order process, summary, 15–87
- Survey–Master/Detail process
  - activities, 15–44
  - summary, 15–42
- Survey–Single Process, activities, 15–41
- Survey–Single process, summary, 15–39
- SuspendProcess(), 8–32
- Synch\_all(), 8–146
- Synch\_changes(), 8–145
- Synchronization, with Oracle Internet Directory, 2–30, 2–34, 8–144
- Synchronize Event Systems event, 14–5
- Synchronous processes, 8–14, C – 2
- System Created event, 14–4
- System Deleted event, 14–4
- System identifier, 13–68
- System Identifier web page, 13–68
- System integration, 13–2
- System Signup event, 14–9
- System Signup web page, 13–69
- System Updated event, 14–4
- System: Error item type, 6–26
- System: Mailer item type, 2–69
- Systems, 13–17
  - defining, 13–18
  - deleting, 13–21
  - finding, 13–19
  - local, 13–17
  - master/copy, 13–72
  - signing up, 13–67, 13–69
  - synchronizing, 13–70
  - updating, 13–21
- Systems web page, 13–18, 13–21

## T

- Tag files, 2–63
- TAGFILE parameter, 2–63
- TCP/IP drivers, 2–2
- Templated Response e–mail, 10–3
- Test harness, 12–2
- Test(), 8–268



TEST\_ADDRESS parameter, 2–62  
TestContext(), 8–229  
Text-type attributes, 4–3  
Timed out processes, 2–43, C – 7  
Timeout transitions, 5–2, 5–3  
Timeouts, 5–10  
    dynamic, 5–10  
Token substitution  
    attributes, 4–41  
    of document-type message attributes, 4–14  
TOKEN(), 8–104  
Toolbars, Oracle Workflow Builder, A – 7  
toString(), 8–99  
Total(), 8–116  
TotalPERM(), 8–117  
TRANSFER mode, 8–13  
Transfer(), 8–195, 8–207  
Transitions, 5–2  
    Any, 5–2  
    creating, 5–18  
    Default, 5–2  
    editing, 5–18  
    Timeout, 5–2  
TRANSLATE(), 8–110  
Translation, 2–38

## U

Unexpected event, 14–12  
UNIX Sendmail, 2–55  
UNPROCESS parameter, 2–63  
Upgrading workflow definitions, 8–12  
URL attributes, frame target, 4–37  
URL message attributes, attached vs  
    embedded, 4–37  
URL-type attributes, 4–3  
URLs  
    for event data, 8–50  
    for Event System Demonstration web pages,  
        15–67, 15–69  
    for Find Notifications Routing Rules web  
        page, 10–27

    for Find Notifications web page, 10–13  
    for Find Processes web page, 11–8  
    for Notifications Routing Rules web page,  
        10–26  
    for Oracle Workflow home page, 9–2  
    for Product Survey web page, 15–36  
    for Requisition Demonstration web page,  
        15–10  
    for the Workflow Monitor, 11–7  
    for Worklist web page, 10–13  
User Defined Alert Action process  
    activities, 15–61  
    summary, 15–60  
User Entry Has Changed event, 14–15  
User Preferences, web page, 9–6  
User preferences, 2–14  
    document management home, 2–20, 9–8  
    language and territory, 2–19, 9–7  
    notification preference, 2–20, 9–8  
User-defined datatypes, for the Business Event  
    System, 8–236  
UserActive(), 8–128  
Users, ad hoc, 5–24

## V

Vacation forwarding, 10–25  
value(), 8–93  
Verify Authority function activity, 15–29  
Version, 8–11, 16–16  
    of Oracle Workflow, 2–9  
Version compatibility, 2–9  
Version number, for activities, 4–60  
Versioning, 3–7  
View menu, A – 4  
View notifications  
    e-mail summary, 10–24  
    electronic mail, 10–2  
    Notification Web page, 10–13  
    web browser, 10–12  
Views, Oracle Workflow, 8–157  
Vote Yes/No activity, 6–10  
VoteCount(), 8–213  
Voting activities  
    processing, 8–195

result type, 4–62  
Voting activity, 4–61

## W

Wait activity, 6–4  
Wait for Concurrent Program activity, 6–24  
Wait for Flow activity, 6–12  
Warning(), 8–285  
Web agent, for Oracle Workflow, 2–17  
Web home page, 9–2  
Web notifications, requirements, 2–4  
WF\_ACCESS\_LEVEL, 2–102, 2–106  
WF\_AGENT\_T, 8–237  
WF\_AGENTS Document Type Definition, 8–311  
WF\_AGENTS\_PKG.Generate, 8–312  
WF\_AGENTS\_PKG.Receive, 8–313  
WF\_DEFERRED agent, 13–25  
WF\_DEFERRED queue, 2–97  
WF\_ENGINE.BACKGROUND, 2–44  
WF\_ERROR agent, 13–25  
WF\_ERROR queue, 2–97  
WF\_ERROR\_QH, 13–25  
WF\_EVENT\_FUNCTIONS\_PKG.Generate(), 8–296  
WF\_EVENT\_FUNCTIONS\_PKG.Receive(), 8–298  
WF\_EVENT\_GROUPS Document Type Definition, 8–305  
WF\_EVENT\_GROUPS\_PKG.Generate, 8–306  
WF\_EVENT\_GROUPS\_PKG.Receive, 8–307  
WF\_EVENT\_OMB\_QH, 13–25  
    attribute mapping, 8–257  
    setting up, 2–100  
WF\_EVENT\_QH, 13–25  
WF\_EVENT\_SUBSCRIPTIONS Document Type Definition, 8–314  
WF\_EVENT\_SUBSCRIPTIONS\_PKG.Generate, 8–315  
WF\_EVENT\_SUBSCRIPTIONS\_PKG.Receive, 8–316  
WF\_EVENT\_T, 8–242  
    mapping attributes to OMBAQ\_TEXT\_MSG, 8–257  
WF\_EVENTS Document Type Definition, 8–302  
WF\_EVENTS\_PKG.Generate, 8–303  
WF\_EVENTS\_PKG.Receive, 8–304  
WF\_IN agent, 13–25  
WF\_IN queue, 2–97  
WF\_ITEM\_ACTIVITY\_STATUSES\_V, 8–157  
WF\_ITEMS\_V, 8–161  
WF\_LANGUAGES view, 2–38  
WF\_LOCAL\_\* tables, 2–21  
WF\_NOTIFICATION() message function, 4–26  
WF\_NOTIFICATION\_ATTR\_RESP\_V, 8–159  
WF\_OUT agent, 13–25  
WF\_OUT queue, 2–97  
WF\_PARAMETER\_LIST\_T, 8–241  
WF\_PARAMETER\_T, 8–239  
wf\_payload\_t, 8–163  
WF\_PURGE, 8–111  
WF\_REQDEMO.SelectApprover, 15–26  
WF\_REQDEMO.StartProcess, 15–8  
WF\_REQDEMO.VerifyAuthority, 15–18, 15–29  
WF\_RESOURCES, environment variable, 2–42  
WF\_ROLES, view, 2–24  
WF\_RUNNABLE\_PROCESSES\_V, 8–160  
WF\_SYSTEMS Document Type Definition, 8–308  
WF\_SYSTEMS\_PKG.Generate, 8–309  
WF\_SYSTEMS\_PKG.Receive, 8–310  
WF\_USER\_ROLES, view, 2–25  
WF\_USERS, view, 2–22  
Wfagtlst.sql, 16–6  
WFAAttribute class, 8–90  
WFAAttribute(), 8–92  
Wfbkg.sql, 16–6  
Wfbkgchk.sql, 16–7  
Wfchact.sql, 16–7  
Wfchacta.sql, 16–7  
Wfchita.sql, 16–8  
Wfchitt.sql, 16–8

Wfchluc.sql, 16–8  
 Wfchlut.sql, 16–9  
 Wfchmsg.sql, 16–9  
 Wfchmsga.sql, 16–9  
 Wfdirchk.sql, 16–10  
 wfdircsv.sql, 2–28  
 wfdirhrv.sql, 2–27  
 wfdirouv.sql, 2–27  
 wfevquec.sql, 2–99  
 wfevqued.sql, 2–99  
 Wfevtenq.sql, 16–10  
 WFFunctionAPI class, 8–82  
 wfjvlsnr.bat, 2–87  
 wfjvlsnr.csh, 2–87  
 Wfjvstop.sql, 16–11  
 WFLOAD, 2–109  
 wflod, 2–108  
 wfmail.cfg, 2–58  
 Wfmqupd.sql, 16–12  
 WFNLADD.sql, 16–5  
 WFNLENA.sql, 16–12  
 Wfntfsh.sql, 16–12  
 Wfprot.sql, 16–12  
 Wfqclean.sql, 16–13  
 Wfquhndob.pls, 2–100  
 Wfquhndos.pls, 2–100  
 Wfrefchk.sql, 16–13  
 wfresgen, 8–105  
 Wfretry.sql, 16–13  
 Wfrmall.sql, 16–14  
 Wfrmita.sql, 16–14  
 Wfrmitms.sql, 16–15  
 Wfrmitt.sql, 16–15  
 Wfrmtime.sql, 16–15, C – 9  
 Wfrun.sql, 16–15  
 WFRUND.SQL, 15–8  
 Wfstat.sql, 16–16  
 Wfstatus.sql, 16–16  
 Wfststdchk.sql, 16–16  
 Wftypes.sql, 8–236  
 Wfupart.sql, 2–12  
 Wfupartb.sql, 2–12  
 Wfver.sql, 16–16  
 Wfverchk.sql, 16–17  
 Wfverupd.sql, 16–17  
 wfxload, 2–114, 2–117  
 wfxload.bat, 2–114, 2–117  
 Windows menu, A – 6  
 WorkCount( ), 8–231  
 Workflow administrator, 2–16  
 Workflow Agent Listener, 16–4  
 Workflow Agent Listener concurrent program, 8–272  
 Workflow Agent Ping/ Acknowledge, 13–77  
   item type, 13–78  
   item type attributes, 13–78  
 Workflow Builder menus, A – 2  
 Workflow Cancelled Mail message template, 2–78  
 Workflow Closed Mail message template, 2–81  
 Workflow Core APIs, 8–101  
 Workflow definitions  
   loading, 1–4  
   source control, 3–12  
   testing, 12–2  
   transferring, 2–107  
 Workflow Definitions Loader, 1–4, 2–107, 2–108  
   concurrent program, 2–109  
 Workflow Demonstrations home page, 15–2  
 Workflow Designer. *See* Oracle Workflow Builder  
 Workflow diagrams, displaying, 15–3  
 Workflow Directory Service APIs, 8–121  
 Workflow Engine, 1–3  
   calling after activity completion, 8–8  
   calling for activity initiation, 8–3  
   CANCEL mode, 8–11  
   core APIs, 8–101, 8–111  
   cost threshold, 4–47  
   deferred activities, 8–9  
   directory services, 8–121  
   error processing, 8–10  
   Java APIs, 8–5, 8–19  
   looping, 8–10

- master/detail processes, 8–79
- PL/SQL APIs, 8–19
- RUN mode, 8–11
- threshold cost, 2–47, 8–9
- Workflow Engine APIs, 8–3
- Workflow Event Protocol process, summary, 14–20
- Workflow Invalid Mail message template, 2–79
- Workflow LDAP APIs, 2–34, 8–144
- Workflow Monitor, 11–2
  - Administration buttons, 11–6
  - Detail Tab window, 11–4
  - Process Diagram window, 11–3
  - Process title, 11–3
  - setup, 2–84
- Workflow Monitor APIs, 8–149
- Workflow Notification APIs. *See* Notification APIs
- Workflow Open Mail (Direct) message template, 2–71
- Workflow Open Mail (Templated) message template, 2–69
- Workflow Open Mail for Outlook Express message template, 2–74
- Workflow Open Mail message template, 2–76
- Workflow Preferences API, 8–148
- Workflow processes
  - creating and starting, 16–15
  - monitoring, 11–2
  - samples, 15–2
- Workflow Purge APIs, 8–111
- Workflow Queue APIs, 8–162
- Workflow queues, cleaning, 16–13
- Workflow Resource Generator, 8–105
  - concurrent program, 8–106

- Workflow roles, 2–21
- Workflow Send Protocol
  - item type, 14–18
  - sample workflow process, 14–17
- Workflow Send Protocol Acknowledgement event, 14–26
- Workflow Send Protocol event, 14–24
- Workflow Server, requirements, 2–3
- Workflow Summary Mail message template, 2–82
- Workflow URL Attachment message template, 2–77
- Workflow users, 2–21
- Workflow Views, 8–157
- Workflow Warning Mail message template, 2–82
- Workflow web pages, modifying template, 2–84
- Workflow XML Loader, 2–112
- Workflow\_Protocol( ), 8–287
- Workitems. *See* Items
- Worklist web page, 10–17
- WriteMsg( ), 8–182
- WriteToClob( ), 8–234

## X

- XML Compare Tag Value (Date) activity, 6–19
- XML Compare Tag Value (Number) activity, 6–19
- XML Compare Tag Value (Text) activity, 6–19
- XML Compare Tag Value activities, 6–19
- XML Get Tag Value activity, 6–18
- XML Transform activity, 6–20

# Reader's Comment Form

## Oracle Workflow Guide Volume 2, Release 2.6.2 A95277-03

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.

