

# Oracle Workflow

ガイド

リリース 2.6.2

2002 年 7 月

部品番号 : J06292-01

ORACLE®

---

Oracle Workflow ガイド, リリース 2.6.2

部品番号: J06292-01

原本名: Oracle Workflow Guide, Release 2.6.2

原本部品番号: A95276-03 (Vol.1)、A95277-03 (Vol.2)

原本著者: Siu Chang, Clara Jaeckel

原本協力者: 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

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

Printed in Japan.

制限付権利の説明

プログラム (ソフトウェアおよびドキュメントを含む) の使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されています。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

\* オラクル社とは、Oracle Corporation (米国オラクル) または日本オラクル株式会社 (日本オラクル) を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation (米国オラクル) およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

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.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

---

---

# 目次

<b>はじめに</b> .....	xvii
対象読者 .....	xviii
このマニュアルの構成 .....	xviii
その他の情報 .....	xix
Oracle Applications データの変更を目的としたデータベース・ツール使用の禁止 .....	xxv
オラクル社について .....	xxv
 <b>1 Oracle Workflow の概要</b>	
Oracle Workflow の概要 .....	1-2
主な機能と定義 .....	1-3
ワークフロー・プロセス .....	1-5
 <b>2 Oracle Workflow の設定</b>	
Oracle Workflow のハードウェア要件とソフトウェア要件 .....	2-2
設定の概要 .....	2-6
Oracle Workflow スタンドアロン版での設定手順の概要 .....	2-6
Oracle Applications embedded Workflow での設定手順の概要（必須） .....	2-6
オプションの設定手順 .....	2-7
その他のワークフロー機能 .....	2-8
Oracle Workflow Server のバージョンの確認 .....	2-8
設定フローチャート .....	2-9
設定チェックリスト .....	2-10
設定手順 .....	2-11
Oracle Workflow のアクセス保護の概要 .....	2-95
デフォルトのアクセス・レベルの設定 .....	2-99

ワークフロー定義ローダーの使用 .....	2-101
Workflow XML Loader の使用 .....	2-106

### 3 ワークフロー・プロセス定義

Oracle Workflow Builder の概要 .....	3-2
ナビゲータ・ツリーの構造 .....	3-4
ナビゲータ・ツリーの表示 .....	3-5
Oracle Workflow Builder によるプロセス定義の作成 .....	3-8
項目タイプのオープンと保存 .....	3-13
クイック・スタート・ウィザードの概要 .....	3-19
各種リリースのサーバーでの Oracle Workflow Builder の使用 .....	3-22
「項目タイプの定義」 Web 画面 .....	3-25

### 4 ワークフロー・プロセスのコンポーネントの定義

ワークフロー・プロセスのコンポーネント .....	4-2
項目タイプ .....	4-2
オブジェクトへのアクセス許可の設定 .....	4-16
選択肢タイプ .....	4-18
メッセージ .....	4-22
アクティビティ .....	4-40
投票アクティビティ .....	4-58
Oracle Workflow Builder のオブジェクトの削除 .....	4-64
Oracle Workflow Builder のオブジェクトの変更 .....	4-65
バージョン管理をサポートするワークフロー・オブジェクト .....	4-66
バージョン管理をサポートしないワークフロー・オブジェクト .....	4-67

### 5 ワークフロー・プロセス・ダイアグラムの定義

「プロセス」ウィンドウ .....	5-2
Oracle Workflow Builder のフォントの変更 .....	5-22
ワークフロー・プロセスのショートカット・アイコンの作成 .....	5-24
ロール .....	5-25

### 6 Oracle Workflow の事前に定義されたアクティビティ

標準アクティビティ .....	6-2
-----------------	-----

And/Or アクティビティ .....	6-2
「比較」アクティビティ .....	6-3
「実行時間の比較」アクティビティ .....	6-3
「待機」アクティビティ .....	6-4
「ブロック」アクティビティ .....	6-5
「スレッドの遅延」アクティビティ .....	6-5
「プロセスの開始」アクティビティ .....	6-5
「Noop」アクティビティ .....	6-6
「ループ・カウンタ」アクティビティ .....	6-6
「開始」アクティビティ .....	6-8
「終了」アクティビティ .....	6-8
「ロール解決」アクティビティ .....	6-8
「通知」アクティビティ .....	6-8
「はい / いいえ投票」アクティビティ .....	6-10
「マスター / ディテール連携」アクティビティ .....	6-11
「フロー待ち」アクティビティ .....	6-11
「継続フロー」アクティビティ .....	6-11
「割当」アクティビティ .....	6-13
「モニター URL」アクティビティ .....	6-13
「イベント・プロパティの取得」アクティビティ .....	6-14
「イベント・プロパティの設定」アクティビティ .....	6-14
「イベント・プロパティの比較」アクティビティ .....	6-15
「XML タグ値の取得」アクティビティ .....	6-17
「XML タグ値の比較」アクティビティ .....	6-18
「XML 変換」アクティビティ .....	6-19
<b>コンカレント・マネージャの標準アクティビティ .....</b>	<b>6-20</b>
「コンカレント・プログラムの実行」アクティビティ .....	6-20
「コンカレント・プログラムの発行」アクティビティ .....	6-21
「コンカレント・プログラム待ち」アクティビティ .....	6-22
<b>デフォルト・エラー・プロセス .....</b>	<b>6-23</b>
「システム:エラー」項目タイプと項目属性 .....	6-24
デフォルト・エラー・プロセス .....	6-25
「再試行のみ」プロセス .....	6-29
デフォルト・イベント・エラー・プロセス .....	6-32

## 7 Oracle Workflow のプロシージャおよび関数の定義

Oracle Workflow のプロシージャおよび関数の定義 .....	7-2
関数アクティビティがコールする PL/SQL プロシージャの標準 API .....	7-3
関数アクティビティがコールする Java プロシージャの標準 API .....	7-8
項目タイプのセレクト関数またはコールバック関数の標準 API .....	7-12
PL/SQL および PL/SQL CLOB 文書の標準 API .....	7-15
PL/SQL 文書 .....	7-15
PL/SQL CLOB 文書 .....	7-17
イベント・データ・ジェネレート関数の標準 API .....	7-19
キュー・ハンドラの標準 API .....	7-21
エンキュー .....	7-22
デキュー .....	7-22
イベント・サブスクリプションのルール関数の標準 API .....	7-24

## 8 Oracle Workflow の API

Oracle Workflow のプロシージャと関数 .....	8-2
ワークフロー・エンジンの概要 .....	8-3
Oracle Workflow Java インタフェース .....	8-6
ワークフロー・エンジンのその他の機能 .....	8-9
Workflow Engine API .....	8-20
CreateProcess .....	8-22
SetItemUserKey .....	8-25
GetItemUserKey .....	8-26
GetActivityLabel .....	8-27
SetItemOwner .....	8-28
StartProcess .....	8-30
LaunchProcess .....	8-33
SuspendProcess .....	8-35
ResumeProcess .....	8-37
AbortProcess .....	8-39
CreateForkProcess .....	8-41
StartForkProcess .....	8-43
Background .....	8-44
AddItemAttribute .....	8-46
AddItemAttributeArray .....	8-49
SetItemAttribute .....	8-51

SetItemAttrDocument .....	8-54
SetItemAttributeArray .....	8-56
getItemTypes .....	8-58
GetItemAttribute .....	8-59
GetItemAttrDocument .....	8-61
GetItemAttrClob .....	8-62
getItemAttributes .....	8-63
GetItemAttrInfo .....	8-64
GetActivityAttrInfo .....	8-65
GetActivityAttribute .....	8-66
GetActivityAttrClob .....	8-68
BeginActivity .....	8-69
CompleteActivity .....	8-71
CompleteActivityInternalName .....	8-74
AssignActivity .....	8-76
Event .....	8-77
HandleError .....	8-79
SetItemParent .....	8-81
ItemStatus .....	8-82
getProcessStatus .....	8-83
<b>Workflow Function API .....</b>	<b>8-84</b>
loadItemAttributes .....	8-85
loadActivityAttributes .....	8-86
getActivityAttr .....	8-87
getItemAttr .....	8-89
setItemAttrValue .....	8-90
execute .....	8-91
<b>Workflow Attribute API .....</b>	<b>8-92</b>
WFAttribute .....	8-94
value .....	8-95
getName .....	8-96
getValue .....	8-97
getType .....	8-98
getFormat .....	8-99
getValueType .....	8-100
toString .....	8-101
compareTo .....	8-102
<b>Workflow CORE API .....</b>	<b>8-104</b>

CLEAR .....	8-105
GET_ERROR .....	8-106
TOKEN .....	8-107
RAISE .....	8-108
CONTEXT .....	8-111
TRANSLATE .....	8-113
<b>Workflow PURGE API .....</b>	<b>8-114</b>
Items .....	8-116
Activities .....	8-117
Notifications .....	8-118
Total .....	8-119
TotalPERM .....	8-120
AdHocDirectory .....	8-121
「ワークフローの不要ランタイム・データのパージ」 コンカレント・プログラム .....	8-122
<b>Workflow ディレクトリ・サービス API .....</b>	<b>8-124</b>
GetRoleUsers .....	8-126
GetUserRoles .....	8-127
GetRoleInfo .....	8-128
GetRoleInfo2 .....	8-129
IsPerformer .....	8-130
UserActive .....	8-131
GetUserName .....	8-132
GetRoleName .....	8-133
GetRoleDisplayName .....	8-134
SetAdHocUserStatus .....	8-135
SetAdHocRoleStatus .....	8-136
CreateAdHocUser .....	8-136
CreateAdHocRole .....	8-139
AddUsersToAdHocRole .....	8-141
SetAdHocUserExpiration .....	8-142
SetAdHocRoleExpiration .....	8-143
SetAdHocUserAttr .....	8-144
SetAdHocRoleAttr .....	8-146
RemoveUsersFromAdHocRole .....	8-147
<b>Workflow LDAP API .....</b>	<b>8-148</b>
Synch_changes .....	8-149
Synch_all .....	8-150
Schedule_changes .....	8-151



<b>Workflow Preference API</b> .....	8-152
get_pref .....	8-152
<b>Workflow Monitor API</b> .....	8-153
GetAccessKey .....	8-154
GetDiagramURL .....	8-155
GetEnvelopeURL .....	8-157
GetAdvancedEnvelopeURL .....	8-159
<b>Oracle Workflow のビュー</b> .....	8-161
WF_ITEM_ACTIVITY_STATUSES_V .....	8-161
WF_NOTIFICATION_ATTR_RESP_V .....	8-163
WF_RUNNABLE_PROCESSES_V .....	8-164
WF_ITEMS_V .....	8-165
<b>Workflow QUEUE API</b> .....	8-167
EnqueueInbound .....	8-170
DequeueOutbound .....	8-172
DequeueEventDetail .....	8-175
PurgeEvent .....	8-177
PurgeItemType .....	8-178
ProcessInboundQueue .....	8-179
GetMessageHandle .....	8-180
DequeueException .....	8-181
DeferredQueue .....	8-182
InboundQueue .....	8-183
OutboundQueue .....	8-184
ClearMsgStack .....	8-185
CreateMsg .....	8-186
WriteMsg .....	8-187
SetMsgAttr .....	8-188
SetMsgResult .....	8-189
<b>文書管理 API</b> .....	8-191
get_launch_document_url .....	8-192
get_launch_attach_url .....	8-193
get_open_dm_display_window .....	8-194
get_open_dm_attach_window .....	8-195
set_document_id_html .....	8-196
<b>Oracle Workflow 通知システムの概要</b> .....	8-197
通知モデル .....	8-197
<b>通知 API</b> .....	8-202

Send .....	8-204
SendGroup .....	8-208
Forward .....	8-210
Transfer .....	8-212
Cancel .....	8-214
CancelGroup .....	8-215
Respond .....	8-216
Responder .....	8-218
VoteCount .....	8-219
OpenNotificationsExist .....	8-220
Close .....	8-221
AddAttr .....	8-222
SetAttribute .....	8-223
GetAttrInfo .....	8-225
GetInfo .....	8-226
GetText .....	8-227
GetShortText .....	8-228
GetAttribute .....	8-229
GetAttrDoc .....	8-231
GetSubject .....	8-232
GetBody .....	8-233
GetShortBody .....	8-234
TestContext .....	8-235
AccessCheck .....	8-236
WorkCount .....	8-237
getNotifications .....	8-238
getNotificationAttributes .....	8-239
WriteToClob .....	8-240
<b>Oracle Workflow ビジネス・イベント・システムの概要 .....</b>	<b>8-241</b>
<b>ビジネス・イベント・システムのデータ型 .....</b>	<b>8-242</b>
エージェント構造 .....	8-243
getName .....	8-243
getSystem .....	8-243
setName .....	8-244
setSystem .....	8-244
パラメータ構造 .....	8-245
getName .....	8-245
getValue .....	8-245

setName .....	8-246
setValue .....	8-246
パラメータ・リスト構造 .....	8-247
イベント・メッセージ構造 .....	8-248
Initialize .....	8-251
getPriority .....	8-251
getSendDate .....	8-252
getReceiveDate .....	8-252
getCorrelationID .....	8-252
getParameterList .....	8-252
getEventName .....	8-253
getEventKey .....	8-253
getEventData .....	8-253
getFromAgent .....	8-253
getToAgent .....	8-254
getErrorSubscription .....	8-254
getErrorMessage .....	8-254
getErrorStack .....	8-254
setPriority .....	8-255
setSendDate .....	8-255
setReceiveDate .....	8-255
setCorrelationID .....	8-256
setParameterList .....	8-256
setEventName .....	8-257
setEventKey .....	8-257
setEventData .....	8-257
setFromAgent .....	8-258
setToAgent .....	8-258
setErrorSubscription .....	8-258
setErrorMessage .....	8-259
setErrorStack .....	8-259
Content .....	8-260
Address .....	8-260
AddParameterToList .....	8-261
GetValueForParameter .....	8-261
抽象データ型の使用例 .....	8-262
WF_EVENT_T および OMBAQ_TEXT_MSG 間のマッピング .....	8-264
イベントの API .....	8-267

Raise .....	8-268
Send .....	8-272
NewAgent .....	8-274
Test .....	8-275
Enqueue .....	8-276
Listen .....	8-277
「ワークフロー・エージェント・リスナー」コンカレント・プログラム .....	8-279
SetErrorInfo .....	8-280
SetDispatchMode .....	8-281
AddParameterToList .....	8-282
AddParameterToListPos .....	8-283
GetValueForParameter .....	8-284
GetValueForParameterPos .....	8-285
<b>イベント・サブスクリプションのルール関数の API</b> .....	8-286
Default_Rule() .....	8-287
Log .....	8-289
Error .....	8-290
Warning .....	8-291
Success .....	8-292
Workflow_Protocol .....	8-293
Error_Rule .....	8-294
SetParametersIntoParameterList .....	8-295
<b>イベント関数の API</b> .....	8-296
Parameters .....	8-297
SubscriptionParameters .....	8-299
AddCorrelation .....	8-300
Generate .....	8-302
Receive .....	8-304
<b>ビジネス・イベント・システムのレプリケーションの API</b> .....	8-306
WF_EVENTS ドキュメント・タイプ定義 .....	8-308
WF_EVENTS_PKG.Generate .....	8-309
WF_EVENTS_PKG.Receive .....	8-310
WF_EVENT_GROUPS ドキュメント・タイプ定義 .....	8-311
WF_EVENT_GROUPS_PKG.Generate .....	8-312
WF_EVENT_GROUPS_PKG.Receive .....	8-313
WF_SYSTEMS ドキュメント・タイプ定義 .....	8-314
WF_SYSTEMS_PKG.Generate .....	8-315
WF_SYSTEMS_PKG.Receive .....	8-316

WF_AGENTS ドキュメント・タイプ定義 .....	8-317
WF_AGENTS_PKG.Generate .....	8-318
WF_AGENTS_PKG.Receive .....	8-319
WF_EVENT_SUBSCRIPTIONS ドキュメント・タイプ定義 .....	8-320
WF_EVENT_SUBSCRIPTIONS_PKG.Generate .....	8-321
WF_EVENT_SUBSCRIPTIONS_PKG.Receive .....	8-322

## 9 Oracle Workflow ホーム・ページ

Oracle Workflow ホーム・ページへのアクセス .....	9-2
ユーザー設定項目の設定 .....	9-6

## 10 通知の表示と応答の処理

通知処理の概要 .....	10-2
電子メールによる通知の閲覧 .....	10-2
Web ブラウザによる通知の表示 .....	10-13
電子メールによる通知要約の確認 .....	10-25
自動通知処理ルール の定義 .....	10-26

## 11 ワークフロー・プロセス監視

ワークフロー監視の概要 .....	11-2
ワークフロー・モニター .....	11-2
ワークフロー・モニターへのアクセス .....	11-8

## 12 ワークフロー定義のテスト

ワークフロー定義のテスト .....	12-2
--------------------	------

## 13 ビジネス・イベントの管理

ビジネス・イベントの管理 .....	13-2
イベント .....	13-4
システム .....	13-18
エージェント .....	13-24
イベント・サブスクリプション .....	13-37
メッセージ伝播の設定 .....	13-56
イベントの呼出し .....	13-71

システムのサインアップ .....	13-73
システムの同期 .....	13-77
ローカル・キューの確認 .....	13-79
<b>ワークフロー・エージェントの Ping/ 確認</b> .....	13-84
「ワークフロー・エージェントの Ping/ 確認」項目タイプ .....	13-85
「マスター Ping」プロセスの概要 .....	13-86
「マスター Ping」プロセス・アクティビティ .....	13-87
「詳細 Ping」プロセスの概要 .....	13-88
「詳細 Ping」プロセス・アクティビティ .....	13-89

## 14 事前定義済ワークフロー・イベント

<b>事前定義済ワークフロー・イベント</b> .....	14-2
イベント定義イベント .....	14-3
イベント・グループ定義イベント .....	14-4
システム定義イベント .....	14-5
エージェント定義イベント .....	14-6
イベント・サブスクリプション定義イベント .....	14-7
イベント・システムの同期イベント .....	14-8
シード・イベント・グループ .....	14-8
エージェントの Ping イベント .....	14-10
システムのサインアップ・イベント .....	14-12
Any イベント .....	14-13
Unexpected イベント .....	14-15
User Entry Has Changed イベント .....	14-18
<b>ワークフロー送信プロトコル</b> .....	14-20
「ワークフロー送信プロトコル」項目タイプ .....	14-21
ワークフロー・イベント・プロトコル・プロセスの概要 .....	14-22
ワークフロー・イベント・プロトコル・プロセスのアクティビティ .....	14-24
ワークフロー送信プロトコル・イベント .....	14-27

## 15 デモンストレーション用ワークフロー・プロセス

<b>サンプル・ワークフロー・プロセス</b> .....	15-2
サンプル・ワークフローのプロセス・ダイアグラムの表示 .....	15-3
<b>購買申請プロセス</b> .....	15-5

購買申請データ・モデルのインストール .....	15-6
購買申請ワークフローの開始 .....	15-8
「購買申請」項目タイプ .....	15-13
購買承認申請プロセスの概要 .....	15-14
購買申請プロセス・アクティビティ .....	15-15
「承認者に通知」サブプロセスの概要 .....	15-20
「承認者に通知」サブプロセスのアクティビティ .....	15-21
StartProcess 関数の例 .....	15-23
関数アクティビティの例 .....	15-26
例：承認者の選択 .....	15-26
例：承認権限の検証 .....	15-28
通知アクティビティの例 .....	15-30
例：購買申請承認が必要なことを通知 .....	15-31
<b>製品アンケート・プロセス</b> .....	15-33
製品アンケート・データ・モデルのインストール .....	15-34
製品アンケート・ワークフローの開始 .....	15-35
「製品アンケート」項目タイプ .....	15-38
「アンケート-シングル・プロセス」の概要 .....	15-39
「アンケート-シングル・プロセス」のアクティビティ .....	15-40
「アンケート-マスター / ディテール・プロセス」の概要 .....	15-42
「アンケート-マスター / ディテール・プロセス」のアクティビティ .....	15-43
「詳細アンケート・プロセス」の概要 .....	15-45
「詳細アンケート・プロセス」のアクティビティ .....	15-46
<b>ドキュメント・レビュー・プロセス</b> .....	15-48
「文書管理」項目タイプ .....	15-49
ドキュメント・レビュー・プロセスの概要 .....	15-50
ドキュメント・レビュー・プロセスのアクティビティ .....	15-51
<b>エラー・チェック・プロセス</b> .....	15-53
「定期警告」項目タイプ .....	15-54
エラー・チェック・プロセスの概要 .....	15-55
エラー・チェック・プロセスのアクティビティ .....	15-56
ユーザー定義の警告処理プロセスの概要 .....	15-60
ユーザー定義の警告処理プロセスのアクティビティ .....	15-61
<b>イベント・システム・デモンストレーション</b> .....	15-62
イベント・システム・デモンストレーション・データ・モデルのインストール .....	15-64

イベント・システム・デモンストレーション・ワークフローの開始 .....	15-65
「イベント・システムのデモ」項目タイプ .....	15-71
バイヤー:最上位の発注プロセスの概要 .....	15-73
バイヤー:最上位の発注プロセスのアクティビティ .....	15-74
バイヤー:サプライヤへの発注送信サブプロセスの概要 .....	15-77
バイヤー:サプライヤへの発注送信サブプロセスのアクティビティ .....	15-78
バイヤー:サプライヤの発注承認の受信サブプロセスの概要 .....	15-79
バイヤー:サプライヤの発注承認の受信サブプロセスのアクティビティ .....	15-80
バイヤー:アドバンスト出荷通知サブプロセスの概要 .....	15-82
バイヤー:アドバンスト出荷通知サブプロセスのアクティビティ .....	15-82
バイヤー:サプライヤの請求の受信サブプロセスの概要 .....	15-84
バイヤー:サプライヤの請求の受信サブプロセスのアクティビティ .....	15-84
サプライヤ:最上位のオーダー・プロセスの概要 .....	15-86
サプライヤ:最上位のオーダー・プロセスのアクティビティ .....	15-87
サプライヤ:オーダー詳細の取得サブプロセスの概要 .....	15-90
サプライヤ:オーダー詳細の取得サブプロセスのアクティビティ .....	15-91
サプライヤ:クレジット・チェック・サブプロセスの概要 .....	15-93
サプライヤ:クレジット・チェック・サブプロセスのアクティビティ .....	15-94
サプライヤ:在庫チェック・サブプロセスの概要 .....	15-95
サプライヤ:在庫チェック・サブプロセスのアクティビティ .....	15-95
サプライヤ:アドバンスト出荷通知サブプロセスの概要 .....	15-96
サプライヤ:アドバンスト出荷通知サブプロセスのアクティビティ .....	15-97
サプライヤ:サプライヤの請求書の送信サブプロセスの概要 .....	15-98
サプライヤ:サプライヤの請求書の送信サブプロセスのアクティビティ .....	15-98
発注イベント .....	15-100
発注承認イベント .....	15-102
アドバンスト出荷通知イベント .....	15-104
請求書イベント .....	15-105

## 16 Oracle Workflow の管理スクリプト

様々な SQL スクリプト .....	16-2
FNDWFLST .....	16-5
FNDWFPR .....	16-5
WFNLADD.sql .....	16-6
wfagtlst.sql .....	16-6



wfbkg.sql .....	16-7
wfbkgchk.sql .....	16-7
wfchact.sql .....	16-8
wfchacta.sql .....	16-8
wfchita.sql .....	16-8
wfchitt.sql .....	16-9
wfchluc.sql .....	16-9
wfchlut.sql .....	16-9
wfchmsg.sql .....	16-10
wfchmsga.sql .....	16-10
wfdirchk.sql .....	16-10
wfevtmq.sql .....	16-11
wfjvstop.sql .....	16-12
wfmqupd.sql .....	16-12
wfnlena.sql .....	16-12
wfntfsh.sql .....	16-13
wfprot.sql .....	16-13
wfqclean.sql .....	16-13
wfrefchk.sql .....	16-14
wfretry.sql .....	16-14
wfrmall.sql .....	16-14
wfrmita.sql .....	16-15
wfrmitms.sql .....	16-15
wfrmitt.sql .....	16-15
wfrmttype.sql .....	16-16
wfrun.sql .....	16-16
wfstat.sql .....	16-16
wfstatus.sql .....	16-16
wfstdchk.sql .....	16-16
wfver.sql .....	16-17
wfverchk.sql .....	16-17
wfverupd.sql .....	16-17

## A Oracle Workflow Builder のメニューとツールバー

Oracle Workflow Builder のメニュー .....	A-2
Oracle Workflow Builder のツールバー .....	A-8

**B 他の Oracle 製品への Oracle Workflow の実装**

Oracle E-Business Suite 埋込みの事前定義済のワークフロー .....	B-2
Oracle E-Business Suite への Oracle Workflow ビジネス・イベント・システムの実装 .....	B-14
Oracle9i プラットフォームへの Oracle Workflow の実装 .....	B-15
定義済のワークフロー、イベントおよびサブスクリプションに関するオラクル社の サポート・ポリシー .....	B-16
カスタマイズのガイドライン .....	B-16
カスタマイズに関する問題点の解決 .....	B-17
サポート対象外のカスタマイズ .....	B-17
サポート対象のカスタマイズ .....	B-18

**C Oracle Workflow のパフォーマンスの概念**

Oracle Workflow のパフォーマンスの概念 .....	C-2
パフォーマンス改善のためのワークフロー・プロセスの設計 .....	C-2
パフォーマンス改善のためのランタイム・データの管理 .....	C-7

**D 『Oracle Workflow ガイド』の改訂ノート**

通知メーラーの起動 .....	D-2
応答処理 .....	D-17

**用語集**

**索引**

---

# はじめに

# 対象読者

『Oracle Workflow ガイド』へようこそ。

このマニュアルは、次の作業上の知識を前提としています。

- 自分のビジネスエリアの原則と慣習
- Oracle Workflow

これまでに Oracle Workflow を使用したことがない場合は、Oracle University で開講されている Oracle Workflow 研修クラスを 1 つ以上受講することをお勧めします。

Oracle Applications 製品情報の詳細は、「その他の情報」を参照してください。

また、このマニュアルは、オペレーティング・システムを基本的に理解し、Oracle データベース・サーバー、PL/SQL および Oracle9i Application Server のテクノロジーに精通している読者を想定しています。これらのシステムについての知識を得ていない場合は、Oracle University で開講されている研修クラスを 1 つ以上受講されることをお勧めします。

## このマニュアルの構成

このマニュアルには、Oracle Workflow を理解および使用するために必要な情報が記載されています。

- 第 1 章では、Oracle Workflow の概要について説明します。
- 第 2 章では、Oracle Workflow をサイトに実装する方法について説明します。
- 第 3 章では、ワークフロー・プロセスを定義する方法について説明します。
- 第 4 章では、ワークフロー・プロセスの作成に必要なコンポーネントを定義する方法について説明します。
- 第 5 章では、ワークフロー・プロセスのダイアグラムを作図および定義する方法について説明します。
- 第 6 章では、Oracle Workflow に用意されている標準アクティビティについて説明します。
- 第 7 章では、Oracle Workflow がコールできる PL/SQL ファンクションと Java ファンクションの標準 API について説明します。
- 第 8 章では、Oracle Workflow の API に関する詳細情報について説明します。
- 第 9 章では、Oracle Workflow ホーム・ページについて説明します。ユーザーと管理者は、このホーム・ページから Oracle Workflow のすべての Web ベース機能に一元的にアクセスできます。
- 第 10 章では、ワークフロー通知の表示方法と処理方法について説明します。

- 第 11 章では、ワークフロー・モニターを使用してワークフロー・プロセスのステータスを管理または表示する方法について説明します。
- 第 12 章では、テストのためにワークフロー・プロセスを起動する方法について説明します。
- 第 13 章では、ビジネス・イベントの管理方法について説明します。
- 第 14 章では、Oracle Workflow に用意されている標準イベントについて説明します。
- 第 15 章では、Oracle Workflow に含まれているデモンストレーション用のワークフロー・プロセスについて説明します。
- 第 16 章では、Oracle Workflow に含まれている、その他の管理用 SQL スクリプトについて説明します。
- 付録 A では、Oracle Workflow Builder のメニューとツールバーについて説明します。
- 付録 B では、Oracle Applications embedded Workflow に含まれている事前定義済のワークフロー・プロセス、ビジネス・イベント・システムを利用した Oracle Applications 機能、および Oracle Workflow を利用した Oracle9i プラットフォーム機能について説明します。Oracle Workflow のサポート・ポリシーも記載されています。
- 付録 C では、Oracle Workflow の実行時にパフォーマンス改善のために使用できる概念と技法について説明します。
- 付録 D では、『Oracle Workflow ガイド』に関する最新の改訂情報および追加情報が記載されています。

このマニュアルの最後には、Oracle Workflow の用語集が記載されています。

## その他の情報

オンライン・マニュアル、研修およびサポート・サービスなど、複数の情報源から、Oracle Workflow に関する知識と理解を深めることができます。

このマニュアルで他の Oracle Applications マニュアルに言及している場合は、リリース 11i のマニュアルを指しています。

### オンライン・マニュアル

Oracle Applications embedded Workflow を使用している場合は、すべての Oracle Applications マニュアルをオンライン（HTML または PDF）で使用可能です。

- **オンライン・ヘルプ：** HTML ヘルプの新機能の項に、リリース 11i の新機能に関する説明があります。この情報は、Oracle Workflow のリリースごとに更新されます。新機能の項には、このマニュアルの印刷時には使用可能にならなかった機能に関する情報も記載されています。たとえば、管理者がミニ・パックまたはアップグレードからソフトウェアをインストールすると、このヘルプに新機能の説明が記載されるようになります。オンライン・ヘルプのパッチは、Oracle MetaLink からダウンロードできます。

- **11i Features Matrix:** このドキュメントには、パッチによって使用可能になる新機能と、新機能に関連する新しいドキュメントの一覧が記載されています。新機能マトリックスのドキュメントは、Oracle MetaLink からダウンロードできます。
- **README ファイル:** ダウンロード可能な新しいドキュメントまたはドキュメント・パッチについては、インストールしたパッチの README ファイルを参照してください。

このマニュアルの一部は、Windows ヘルプ形式のオンライン・ヘルプとしても使用可能です。Windows ヘルプには、Oracle Workflow Builder の「ヘルプ」メニューからアクセスできます。

スタンドアロン版の Oracle Workflow を使用している場合は、このマニュアルを HTML 形式で利用でき、その一部は Windows ヘルプ形式でも利用できます。Windows ヘルプには、Oracle Workflow Builder の「ヘルプ」メニューからアクセスできます。HTML 文書は、システム管理者が提供する URL または Oracle Workflow の Web 画面のヘルプ・アイコンから利用できます。

## 関連するユーザーズ・ガイド

Oracle Workflow は、他の Oracle Applications 製品で使用されて、埋込みのワークフローを提供します。Oracle Applications embedded Workflow を使用している場合は、Oracle Workflow を設定および使用しているときに他のユーザーズ・ガイドを参照すれば、埋込みのワークフローの詳細を理解することができます。

オンライン・マニュアルを参照するには、HTML ヘルプ・ウィンドウの拡張可能メニューから「Library」を選択する方法、メディア・パックに同梱されている Oracle Applications ドキュメント・ライブラリ CD から参照する方法、またはシステム管理者から提供される URL を使用して Web ブラウザから参照する方法があります。

## 全製品に関連するガイド

### 『Oracle Applications User's Guide』

このマニュアルには、このリリースの Oracle Workflow で利用可能なグラフィカル・ユーザ・インタフェース（GUI）を使用して、データの入力、問合せ、レポートの実行およびナビゲートを行う方法について記載されています。また、ユーザ・プロファイルの設定や、レポートと同時プロセスの実行および確認についても記載されています。

このユーザーズ・ガイドには、Oracle Applications ヘルプ・ファイルから「Getting Started with Oracle Applications」を選択してオンラインでアクセスすることもできます。

## この製品に関連するマニュアル

### 『Oracle General Ledger User's Guide』

このマニュアルには、仕訳入力、予算処理、複数会社会計および連結に関する情報が記載されています。

### 『Oracle Purchasing User's Guide』

このマニュアルには、発注と購買申請を入力して管理する方法が記載されています。

### 『Oracle Self-Service Human Resources (SSHR) User's Guide』

このマニュアルには、管理者と従業員を対象として、セルフサービス人事管理機能をセットアップする方法が記載されています。管理者と従業員は、イントラネットと Web ブラウザを使用して、個人情報およびキャリア管理機能に簡単にアクセスできます。

### 『Oracle Payables User's Guide』

このマニュアルには、仕入先、請求書および支払を入力して管理する方法が記載されています。

### 『Oracle Projects User's Guide』

このマニュアルには、プロジェクト、予算、費用、原価および請求の入力および管理方法が記載されています。

### 『Oracle Receivables User's Guide』

このマニュアルには、顧客、入金、回収および取引の入力および管理方法が記載されています。

### 『Oracle Business Intelligence System Implementation Guide』

このマニュアルには、環境に Oracle Business Intelligence (BIS) をインプリメントする方法が記載されています。

### 『BIS 11/ User Guide Online Help』

このマニュアルは、BIS アプリケーション専用のオンライン・ヘルプとして提供されるもので、インテリジェンス・レポート、Discoverer ワークブックおよび Performance Management Framework に関する情報が含まれています。

### 『Oracle Financials Open Interface Reference』

このマニュアルは、Oracle Financial Applications のすべてのユーザーズ・ガイドに記載されているオープン・インタフェースの説明を集約したものです。

### 『Oracle XML Gateway User's Guide』

このマニュアルには、適切に構成された有効な XML メッセージを Oracle Applications と取引パートナー間で作成および取り込む方法について記載されています。

## インストールとシステム管理

### 『Oracle Applications Concepts』

このマニュアルでは、Oracle Applications リリース 11i の概念、機能、テクノロジー・スタック、アーキテクチャおよび用語が紹介されています。Oracle Applications のインストール前に参照すると役立つ入門書です。また、Business Intelligence (BIS)、言語とキャラクタ・セット、Self-Service Web Applications など、Oracle Applications 全体の機能の基盤となる概念も紹介しています。

### 『Installing Oracle Applications』

このマニュアルには、Oracle Applications 製品のインストレーションを管理する手順が記載されています。リリース 11i の場合、大部分のインストール処理は Oracle Rapid Install を使用して処理されます。これにより多数の必須手順が自動化され、Oracle Applications、Oracle8 のテクノロジー・スタックおよび Oracle8i サーバーのテクノロジー・スタックを最短時間でインストールできます。このマニュアルには、Oracle Rapid Install の使用方法と、インストール完了までに必要なタスクの一覧が記載されています。このマニュアルは、個々の製品のユーザーズ・ガイドおよび実装ガイドと併用する必要があります。

### 『Upgrading Oracle Applications』

Oracle Applications リリース 10.7 またはリリース 11.0 製品からリリース 11i にアップグレードする場合は、このマニュアルを参照してください。このマニュアルには、アップグレード・プロセスの説明と、データベース固有のアップグレード・タスクおよび製品固有のアップグレード・タスクの一覧が記載されています。リリース 11i には、リリース 10.7 (NCA、SmartClient またはキャラクタ・モード) またはリリース 11.0 からアップグレードする必要があります。リリース 10.7 以前のリリースからリリース 11i に直接アップグレードすることはできません。

### 『Maintaining Oracle Applications』

このマニュアルを参考にして、AutoUpgrade、AutoPatch、AD Administration、AD Controller、AD Relink、License Manager などの各種 AD ユーティリティを実行します。このマニュアルには、操作手順、スクリーンショットおよび AD ユーティリティの実行に必要な他の情報が含まれています。また、Oracle Applications のファイル・システムおよびデータベースの管理についても記載されています。

### 『Oracle Applications System Administrator's Guide』

このマニュアルには、Oracle Applications のシステム管理者向けの計画およびリファレンス情報が記載されています。セキュリティの定義方法、メニューとオンライン・ヘルプのカスタマイズ方法および同時処理の管理方法が含まれています。

### 『Oracle Alert User's Guide』

このマニュアルには、Oracle Applications データの状態を監視するために、定期警告とイベント警告を定義する方法について記載されています。



## 『Oracle Applications Developer's Guide』

このマニュアルには、Oracle Applications 開発スタッフが従う必要のあるコーディングの標準が含まれています。『Oracle Applications User Interface Standards for Forms-Based Products』に記載されている Oracle Applications ユーザー・インタフェースの実装に必要な、Oracle Applications Object Library コンポーネントについて説明しています。また、カスタムの Oracle Forms Developer 6i フォームを作成して Oracle Applications に統合する場合に役立つ情報も記載されています。

## その他の実装のマニュアル

### 『Oracle Applications Product Update Notes』

このマニュアルは、Oracle Applications のアップグレードに関するリファレンスです。リリース 11.0 からリリース 11i への個々の Oracle Applications 製品の変更履歴が記載されています。また、この 2 つのリリース間でデータベース・オブジェクト、プロファイル・オブションおよびシード・データに加えられた新機能、機能拡張および変更も含まれています。

### 『Multiple Reporting Currencies in Oracle Applications』

Multiple Reporting Currencies 機能を使用して複数の通貨で取引を記録する場合は、Oracle Workflow を実装する前にこのマニュアルを参照してください。このマニュアルには、Oracle Workflow にこの機能を実装するための追加手順と設定に関する注意事項が詳しく記載されています。

### 『Multiple Organizations in Oracle Applications』

このマニュアルには、Oracle Workflow の単一インストールを実行するときに複数の組織構造を定義およびサポートできるように、Oracle Applications の Multiple Organization サポート機能を Oracle Workflow に設定して使用する方法が記載されています。

### 『Oracle Applications Flexfields Guide』

このマニュアルには、Oracle Workflow 導入チームおよび Oracle Applications 製品データの進行中のメンテナンスの担当者を対象として、フレックスフィールドの計画作成、設定および参照情報が記載されています。また、フレックスフィールド・データのカスタム・レポートを作成する方法も記載されています。

### 『Oracle eTechnical Reference Manuals』

各『eTechnical Reference Manual (eTRM)』には、特定の Oracle Applications 製品のデータベース・ダイアグラムと、データベース表、フォーム、レポートおよびプログラムに関する詳細な説明が記載されています。この情報を参考にして、既存アプリケーションからのデータ変換、Oracle Applications データと Oracle 以外のアプリケーションの統合および Oracle Applications 向けカスタム・レポートの記述を行うことができます。Oracle eTRM は、Oracle MetaLink からダウンロードできます。

## 『Oracle Applications User Interface Standards for Forms-Based Products』

このマニュアルには、Oracle Applications 開発スタッフが従う必要のあるユーザー・インタフェース (UI) の標準が含まれています。Oracle Applications 製品の UI と、この UI を Oracle Forms で作成するアプリケーションの設計に適用する方法を説明しています。

## 『Oracle Manufacturing APIs and Open Interfaces Manual』

このマニュアルには、他の Oracle Manufacturing アプリケーションおよび他のシステムとの統合に関する最新情報が記載されています。また、Oracle Manufacturing の API とオープン・インタフェースが記載されています。

## 『Oracle Order Management Suite APIs and Open Interfaces Manual』

このマニュアルには、他の Oracle Manufacturing アプリケーションおよび他のシステムとの統合に関する最新情報が記載されています。また、Oracle Order Management Suite の API とオープン・インタフェースに関する説明もあります。

## 『Oracle Applications Message Reference Manual』

このマニュアルには、すべての Oracle Applications メッセージが記載されています。このマニュアルは、リリース 11i のドキュメント CD-ROM から HTML 形式で利用できます。

## 研修とサポート

### 研修

オラクル社は、完全な研修コース・セットを用意しており、管理者とスタッフが Oracle Workflow をマスターし、すぐに生産性を向上できるようにお手伝いします。これらのコースでは、業務や職責分野に合ったコースのみを受講できるように、職務別に研修内容がまとめられています。

また、教育環境を選択することもできます。オラクル社の数ある研修センターで Oracle University が提供するコースに参加することもできますし、御社の施設でオラクル社の講師による研修を開催することもできます。また、Oracle Learning Network(OLN) という Oracle University のオンライン教育ユーティリティを使用することもできます。さらに、オラクル社の研修担当者がニーズにあわせて標準のコースを調整したり、カスタム・コースを開催することもできます。たとえば、御社の施設で開催されるカスタム研修コースで、御社の組織構造、用語およびデータを例として使用することができます。

### サポート

オンサイト・サポートからセンターでのサポートまで、経験豊富な専門家チームが、Oracle Workflow の継続的運用に必要な支援や情報を提供します。このチームには、サービス技術員と会計管理担当の他、顧客の業務部門、Oracle データベース・サーバー、ハードウェアおよびソフトウェア環境に精通した、オラクル社のコンサルタントとサポートの専門家からなる大勢のスタッフが含まれます。

# Oracle Applications データの変更を目的としたデータベース・ツール使用の禁止

特に指示がない限り、Oracle Applications データの変更には、SQL\*Plus、Oracle Data Browser、データベース・トリガーまたは他のツールを使用しないでください。

オラクル社は、Oracle データベース内の情報の作成、格納、変更、取出しおよび保守に使用できるように、強力なツールを提供しています。ただし、SQL\*Plus などの Oracle のツール製品を使用して Oracle Applications データを変更すると、データの整合性が破壊され、データ変更を監査できなくなる恐れがあります。

Oracle Applications 表は相関関係を持っているため、Oracle Applications を使用して変更を加えると、一度に多数の表が更新される可能性があります。ただし、Oracle Applications 以外のツールを使用して Oracle Applications を変更すると、ある表の行を変更しても、関連する表にはそれに対応する変更が加えられない場合があります。表が相互に同期しなくなると、誤った情報を取り出したり、Oracle Applications 全体で予測できない結果を生じる恐れがあります。

Oracle Applications を使用してデータを変更すると、変更が有効かどうかは Oracle Applications によって自動的にチェックされます。Oracle Applications では、情報を変更したユーザーが管理されます。データベース・ツールを使用してデータベース表に情報を入力すると、無効な情報を格納する可能性があります。また、SQL\*Plus およびその他のデータベース・ツールでは変更の記録が保持されないため、情報を変更したユーザーを追跡できなくなります。

## オラクル社について

オラクル社は、Oracle Applications のみでなく、データベース管理、アプリケーション開発、意思決定支援およびオフィス・オートメーション用の統合ソフトウェア製品ラインの開発および販売を行っています。Oracle Applications は、財務管理、サプライ・チェーン管理、製造、プロジェクト・システム、人事およびカスタマー・リレーションシップ・マネジメント用の 160 以上のソフトウェア・モジュールが完全に統合された製品です。

Oracle 製品は、メインフレーム、ミニコンピュータ、パーソナル・コンピュータ、ネットワーク・コンピュータおよびパーソナル・デジタル・アシスタントで使用できます。そのため、組織内の異なるコンピュータ、オペレーティング・システム、ネットワークおよびデータベース管理システムを単一の統一コンピューティングおよび情報リソースに統合できます。

オラクル社は情報管理ソフトウェアのリーディング・カンパニーであり、世界第二位の規模を誇るソフトウェア会社です。データベース、ツールおよびアプリケーション製品のみでなく、関連コンサルティング、研修およびサポート・サービスを世界 145ヶ国以上で提供しています。



---

# Oracle Workflow の概要

この章では、ワークフロー・プロセスの概念と、Oracle Workflow の主要な機能について説明します。ここで説明する機能は、次のとおりです。

- **Oracle Workflow Builder:** ビジネス・プロセスの定義を作成できるグラフィカル・ツールです。
- **ワークフロー・エンジン:** 実行時にプロセス定義を実装します。
- **通知システム:** ワークフロー内のユーザーに通知を送信し、ユーザーからの応答を処理します。
- **ワークフロー・モニター:** このツールにより、Web ブラウザを使用してワークフロー・プロセスを追跡できます。
- **ビジネス・イベント・システム:** システム間でビジネス・イベントを伝達できます。

## Oracle Workflow の概要

現代のビジネス・プロセスでは、常に変化するルールに応じて、複数の人に多くの種類の情報を伝達する必要があります。**Oracle Workflow** を使用すると、ビジネス・プロセスを自動化して継続的に改善していくことができます。また、簡単に変更できるビジネス・ルールに基づいて、どんな種類の情報でも企業内外の人々に発信できます。1-3 ページの「[主な機能と定義](#)」を参照してください。

### 情報のルーティング

使用できる情報が多く、その形式が多様な場合、どのようにして適切な情報を適切な人に伝達しますか？ **Oracle Workflow** を使用すると、それぞれの人に、その人が必要としているすべての情報を提供できます。また、ビジネス・プロセス上の各意思決定者に支援情報を提供することもできます。

### ビジネス・ルールの定義と変更

**Oracle Workflow** では、ドラッグ・アンド・ドロップで操作できるプロセス・デザイナーを使用してビジネス・プロセスを定義し、継続的に改善していくことができます。

単に文書のあるユーザーから他のユーザーへと、いくつかの承認手順を経てルーティングしていくワークフロー・システムとは異なり、**Oracle Workflow** では洗練されたビジネス・プロセス・モデルを作成できます。ループするプロセス、並列フローに分岐して再び合流するプロセス、サブフローやさらにそこから細かい流れに分割されるプロセスなどを定義できます。**Oracle Workflow** では、ストアード・プロシージャの結果に基づいて選択するパスを判別できるため、**Oracle** データベース・サーバーの言語である **PL/SQL** の機能をフルに活用して、ワークフロー・プロセスに影響するあらゆるビジネス・ルールを定義できます。1-5 ページの「[ワークフロー・プロセス](#)」を参照してください。

### 電子通知の配信

**Oracle Workflow** では、ビジネス・プロセスの自動化の範囲を企業全体、さらに電子メールやインターネット・ユーザーを含む企業外のユーザーにまで拡張できます。**Oracle Workflow** では、人々に彼らの対応が待たれる項目の通知を電子メールにより発信し、電子メールの応答に基づいて処理します。また、必要な支援情報が含まれた作業リストを表示し、標準の Web ブラウザを使用して処理を実行することもできます。

### システムの統合

**Oracle Workflow** では、ワークフローを開始できるビジネス・イベントへのサブスクリプションを設定したり、ビジネス・イベントが発生したときにシステム間でメッセージを伝播できます。社内のシステム間や外部システムとの間でイベントを伝達することもできます。この方法により、2 地点間のメッセージ統合を実現したり、より複雑なシステム統合のメッセージ伝達を **Oracle Workflow** を使用して集中管理できます。ルーティングおよび処理ルールが複雑なビジネス・プロセスをモデル化すれば、イベントを強力かつ柔軟に処理することができます。

## 主な機能と定義

### Oracle Workflow Builder

Oracle Workflow Builder では、簡単なドラッグ・アンド・ドロップ操作で、ビジネス・プロセスを作成、表示または変更できます。Oracle Workflow Builder を使用して、アクティビティ、項目タイプおよびメッセージなど、あらゆるワークフロー・オブジェクトを作成、変更できます。1-5 ページの「[ワークフロー・プロセス](#)」を参照してください。

いつでもワークフロー・アクティビティを追加、削除または変更でき、アクティビティ間に必要な新しい関連を設定できます。作業はワークフローの要約レベルのモデルで簡単に行うことができます。必要に応じて、ワークフロー内のアクティビティを展開して、より詳細なレベルで表示することもできます。Oracle Workflow Builder は、デスクトップ PC からでも、ネットワークに接続していないラップトップ PC からでも操作できます。

### ワークフロー・エンジン

Oracle データベース・サーバーに埋め込まれたワークフロー・エンジンでは、ワークフローの状態を監視し、プロセスのアクティビティのルーティングを調整します。たとえば、ワークフロー・アクティビティの完了など、ワークフローの状態が変化すると、PL/SQL API または Java API を介してエンジンにシグナルが送られます。エンジンは、柔軟に定義されているワークフロー・ルールに基づいて実行対象のアクティビティを判別し、そのアクティビティを実行します。ワークフロー・エンジンでは、ループ、分岐、並列フロー、サブフローなど、高度なワークフロー・ルールをサポートしています。

### ビジネス・イベント・システム

ビジネス・イベント・システムは、Oracle Advanced Queuing (AQ) インフラストラクチャを使用してシステム間でビジネス・イベントを伝達するためのアプリケーション・サービスの 1 つです。ビジネス・イベント・システムは、重要なイベントへのサブスクリプションを登録するイベント・マネージャおよびワークフロー・プロセス内のビジネス・イベントをモデル化するイベント・アクティビティから構成されます。

ローカル・イベントが発生すると、イベントを呼び出したコードと同じトランザクション内でサブスクライバ・コードが実行されます。サブスクリプション処理では、イベント情報に関するカスタム・コードの実行、ワークフロー・プロセスに対するイベント情報の送信、他のキューまたはシステムに対するイベント情報の送信などが行われます。

### ワークフロー定義ローダー

ワークフロー定義ローダーは、ワークフロー定義をデータベースとそれに対応するフラット・ファイル間で移動するユーティリティ・プログラムです。このプログラムを使用して、ワークフロー定義を開発用データベースから本番データベースに移行したり、既存の定義をアップグレードできます。ワークフロー定義ローダーはスタンドアロン・サーバー・プログラムですが、Oracle Workflow Builder にも統合されており、データベースやファイル内のワークフロー定義のオープンや保存が可能です。

## 完全なプログラム拡張性

Oracle Workflow では、独自の PL/SQL プロシージャや外部関数をアクティビティとしてワークフローに入れることができます。アプリケーション・コードを変更しなくても、ワークフロー・エンジンによりプログラムの前提条件が満たされていることが確認されれば、いつでも独自のプログラムを実行できます。

## 電子通知

Oracle Workflow では、ユーザーをワークフローに組み込んで、購買申請や受注の承認など、自動化できないアクティビティを処理できます。電子通知はロールにルーティングされます。ロールは、個々のユーザーまたはユーザー・グループです。そのロールに関連付けられているユーザーなら誰でも、その通知に基づいて作業を実行できます。

各通知には意思決定に必要なすべての情報を含むメッセージが入っています。情報は、メッセージの本文に埋め込まれているか、別の文書として添付されています。Oracle Workflow ではそれぞれの通知アクティビティの応答を理解し、次のワークフロー・アクティビティに移動する方法を判断します。

## 電子メールの統合

電子メール (E-mail) ・ユーザーは、未処理の作業項目に関する通知を受信し、選択した電子メール・アプリケーションを使用してその通知に応答できます。電子メール通知には、通知に対する応答の別手段となる添付ファイルを添付できます。

## インターネット対応ワークフロー

標準の Web ブラウザにアクセスできるユーザーは、誰でもワークフローに組み込むことができます。Web ユーザーは、「通知」Web 画面にアクセスして未処理の作業項目を表示し、別のページに移動して、より詳細な内容の表示や応答が可能です。

## 監視と管理

ワークフローの管理者とユーザーは、Java サポート機能を持つ標準 Web ブラウザを使用して、ワークフロー・モニターに接続し、ワークフロー・プロセスの作業項目の進捗を表示できます。ワークフロー・モニターでは、ワークフロー・プロセスの特定インスタンスのプロセス・ダイアグラムが説明付きのビューで表示されるため、ユーザーは作業項目のステータスをグラフィカルに見ることができます。また、作業項目、プロセスおよびプロセス内の各アクティビティについて、個別のステータス要約も表示されます。

Oracle Applications に埋め込まれた Oracle Workflow を使用し、Oracle Applications Manager を実装している場合は、Oracle Applications Manager の Oracle Workflow Manager コンポーネントを Oracle Workflow の追加管理ツールとして使用することもできます。Oracle Applications Manager は、Oracle Applications のコンカレント処理、Oracle Workflow、その他の機能の管理および診断機能を提供するツールの 1 つです。詳細は、Oracle Applications Manager のオンライン・ヘルプを参照してください。



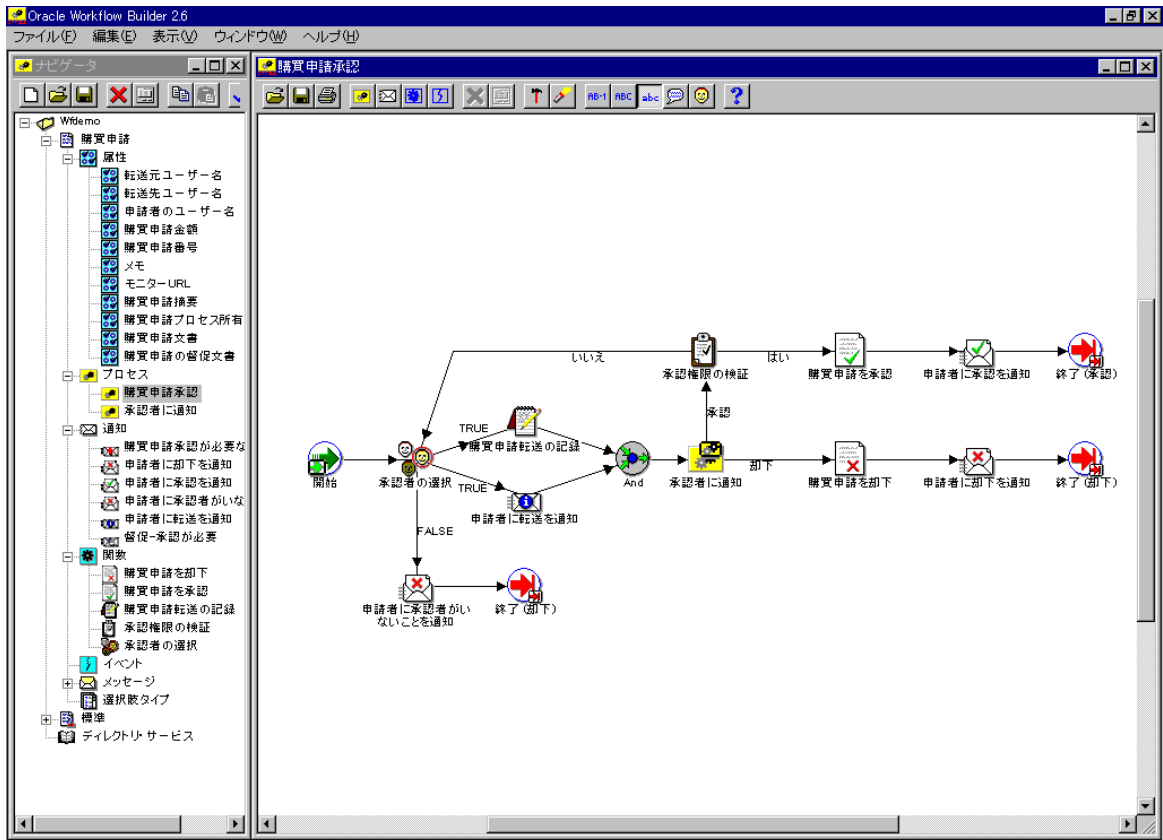
また、Oracle9i リリース 2 で利用可能なスタンドアロン版の Oracle Workflow を使用している場合は、Oracle Enterprise Manager から利用可能なスタンドアロン Oracle Workflow Manager コンポーネントを、Oracle Workflow の追加管理ツールとして使用できます。詳細は、Oracle Workflow Manager のオンライン・ヘルプを参照してください。

## ワークフロー・プロセス

Oracle Workflow は、ユーザーが定義したルールに従ってビジネス・プロセスを管理します。ルールはワークフロー・プロセス定義と呼ばれ、プロセス内で発生するアクティビティと各アクティビティ間の関連を含みます。プロセス定義に含まれるアクティビティには、PL/SQL のストアド・プロシージャや外部関数によって定義される自動化関数、任意に応答を要求する場合があるユーザーやロールへの通知、ビジネス・イベント、より細かいアクティビティの集合で構成されるサブフローなどがあります。

ワークフロー・プロセスは、アプリケーションで一連の Oracle Workflow Engine API をコールしたときに起動されます。ワークフロー・エンジンには、アプリケーションで定義されている関連作業項目を、特定のワークフロー・プロセス定義を介して実行する機能があります。ワークフロー・エンジンでは、ワークフロー・プロセス定義に従って自動化の手順を実行し、外部処理が必要な場合は該当するエージェントを起動します。

次のダイアグラムは、購買申請を管理者や複数の管理者承認をもらうためにルーティングする、簡単なワークフロー・プロセス定義を表しています。



この全体図を、プロセスまたはプロセス・ダイアグラムと呼びます。アイコンはアクティビティを表し、矢印はアクティビティ間の遷移を表しています。この表の例では、ユーザーが該当するアプリケーションで購買申請を作成して送信すると、プロセスに新規項目が作成されます。

このプロセスには、PL/SQL のストアド・プロシージャとして実装されている次のような複数のワークフロー・アクティビティが含まれています。

- 承認者の選択： ビジネス・ルールに従って、購買申請を承認するユーザーを選択します。
- 承認権限の検証： 選択された承認者に、購買申請の承認権限があるかどうかを検証します。

---

## Oracle Workflow の設定

この章では、Oracle Workflow の要件と、Oracle Workflow サイトで設定するための手順について説明します。

## Oracle Workflow のハードウェア要件とソフトウェア要件

Oracle Workflow のコンポーネントに必要なハードウェアとソフトウェアの構成は、次のとおりです。

- Oracle Workflow Builder は、Oracle Universal Installer を使用してインストールし、Oracle Net Services (Oracle8i ではリリース 8.1.6 以降、Oracle9i ではリリース 1 (9.0.1) 以降および Required Support Files (Oracle8i ではリリース 8.1.6 以降、Oracle9i ではリリース 1 (9.0.1) 以降のインストールを必要とします。Oracle Workflow Builder は、次の要件を満たす IBM、Compaq または完全に互換性のあるパーソナル・コンピュータにインストールしてください。
  - 486 以上のプロセッサ
  - 66 MHz 以上のクロック・スピード (90 MHz 以上を推奨)
  - ネットワーク・カード
  - SVGA カラー・モニター
  - ダイアルイン・アクセスが設定されているモデム (オラクル社カスタマ・サポート・センターとの通信で使用)。少なくとも 1 台の PC にモデムが構成されている必要があります。
  - リモート・アクセス / 制御ソフトウェア (カスタマ・サポート・センターがお客様の PC にモデム経由でダイアルイン・アクセスするとき使用します。推奨ソフトウェアは、Symantec 社の Norton pcANYWHERE または Microcom 社の Carbon Copy です。オラクル社のカスタマ・サポート・センターが、お客様のサイトにダイアルインして問題を診断したり、お客様のクライアント PC にパッチを直接適用するには、なんらかのリモート・アクセス / 制御ソフトウェアが必要です。)

---

**警告：** リモート・アクセスを可能にするソフトウェアをインストールする際は、ウィルスや不正アクセスを防ぐために必要なセキュリティ上の注意事項に従ってください。

---

- 論理ドライブとして使用可能な ISO 9660 フォーマットの倍速 CD-ROM
- Microsoft Windows 98、Windows 2000 または Windows NT
- Oracle Workflow Builder、Oracle Net Services および Required Support Files をインストールするための 60MB 以上の空きディスク領域
- 32MB 以上のメモリー (64MB を推奨)

---

**注意：** Oracle Net Services では Microsoft の TCP/IP ドライバが必要であり、このドライバのみをサポートしています。

---

- Oracle Workflow Server は、次の要件を満たす必要があります。
  - Oracle8i Enterprise/Standard Edition データベースのリリース 8.1.6 以降または Oracle9i Enterprise/Standard Edition データベースのリリース 1 (9.0.1) 以降。Oracle Objects および JServer Options が、サポートされているサーバー・マシンにインストールされていること
  - Oracle Workflow Server を Oracle ホームにインストールした場合は、40MB 以上の空きディスク領域
  - 128MB 以上のメモリー (256MB を推奨)
  - Oracle8i では Oracle Net Services リリース 8.1.6 以降、Oracle9i では Oracle Net Services リリース 1 (9.0.1) 以降
  - Oracle8i では SQL\*Plus リリース 8.1 以降、Oracle9i では SQL\*Plus リリース 1 (9.0.1) 以降

Oracle Workflow Server を Microsoft Windows NT にインストールする場合は、次のハードウェア構成およびソフトウェア構成も必要です。

- 論理ドライブとして使用可能な ISO 9660 フォーマットの CD-ROM
- Microsoft Windows NT 4.0 以降

---

---

**注意：** Oracle8i Standard Edition のお客様と Oracle8i Enterprise Edition のお客様がまったく同じ Oracle Workflow を使用している場合でも、Oracle Workflow では使用している Oracle8i より高度な機能は使用できません。つまり、Oracle8i Standard Edition のデータベースを使用する場合は、Oracle Workflow のビジネス・イベント・システムの機能が一部制限されます。

たとえば、次のようになります。

- Oracle8i Standard Edition では、Oracle Workflow のデフォルト・キュー以外に、キューを追加できません。キューを追加する場合は、Oracle8i Enterprise Edition を選択する必要があります。
- Oracle8i Standard Edition を使用している場合、Oracle Advanced Queuing では、ローカル・データベースの外部にメッセージを伝播できません。他のシステムにメッセージを伝播する場合は、Oracle8i Enterprise Edition を選択する必要があります

ただし、Oracle9i ではこれらの制限は適用されません。Oracle9i Standard Edition のデータベースでは、Oracle Workflow を使って Oracle9i Enterprise Edition のデータベースとまったく同じ機能を利用できます。

---

---

- 電子メール通知コンポーネントには、UNIX の Sendmail または Windows NT の MAPI 互換メール・アプリケーション経由でメールを送信できる通知メーラー・プログラムが

含まれます。また、選択した電子メール・アプリケーションと通信できる適切な UNIX ゲートウェイ製品がインストールされていれば、Oracle Workflow から他の電子メール・アプリケーションにメールを送信することもできます。

---

**注意：** 2000 年 6 月 7 日にリリースされた Microsoft Outlook E-mail Security Update は、Oracle Workflow MAPI メーラーで使用される MAPI Common Messaging Calls (CMC) インタフェースをサポートしていません (「OL2000: Developer Information About the Outlook E-mail Security Update」 (<http://support.microsoft.com/support/kb/articles/Q262/7/01.ASP>) を参照)。その結果、この Microsoft Outlook E-mail Security Update 以降の更新が適用されている場合、Oracle Workflow MAPI メーラーの動作は Microsoft Windows プラットフォームでは保証されていません。Oracle Workflow MAPI メーラーの動作は、Windows XP では保証されていません。

Windows NT/2000 で Workflow を使用する場合は、UNIX 版の Oracle Workflow Notification Mailer を UNIX にインストールして、Windows NT/2000 上で実行するワークフロー・サーバーのデータベースに接続する方法が保証されています。

---

- HTML ファイルが添付された電子メール通知の送信、またはそれに対して応答するには、電子メール・アプリケーションに HTML 添付ファイルのサポート機能が必要です。また、添付ファイルを表示するには、JavaScript とフレームをサポートする Web ブラウザ・アプリケーションが必要です。
- Web 通知、ワークフロー・モニタ、およびイベント・マネージャ・コンポーネントでは、Oracle HTTP Server および mod\_plsql がサーバー・マシン上にインストールされている必要があります。Oracle HTTP Server および mod\_plsql コンポーネントは、Oracle8i Database リリース 8.1.7 以降、Oracle9i Database リリース 1 (9.0.1) 以降、または Oracle9i Application Server リリース 1.0.1 以降に含まれています。

通知を表示するには、JavaScript とフレームをサポートする Web ブラウザ・アプリケーションが必要です。ワークフロー・モニターを表示するには、Netscape Communicator バージョン 4.76 以降、Microsoft Internet Explorer バージョン 5.0x または 5.5x など、Java Development Kit (JDK) バージョン 1.1.8 以降および Abstract Windowing Toolkit (AWT) をサポートする Web ブラウザが必要です。

- 外部の Java 関数アクティビティを実行したり、Workflow XML Loader を使用するには、Java Runtime Environment (JRE) バージョン 1.1.8 以降をインストールする必要があります。
- スタンドアロン版の Oracle Workflow の HTML ヘルプを解凍するには、解凍ユーティリティが必要です。

- Oracle Internet Directory 統合を実装するには、Oracle Internet Directory リリース 2 (9.0.2) をインストールする必要があります。シングル・サインオン統合を実装するには、Oracle Internet Directory 統合を実装し、Oracle9iAS Single Sign-On Server リリース 2 (9.0.2) および Oracle9iAS Portal リリース 2 (9.0.2) をインストールし、Oracle HTTP Server と一緒に mod\_osso をインストールする必要があります。

## 設定の概要

Oracle Workflow をインストールしたら、企業サイト向けに実装するために、企業に適した作業環境とコンポーネントを設定します。

### Oracle Workflow スタンドアロン版での設定手順の概要

1. 「グローバル・ワークフロー・プリファレンス」 Web 画面を使用して、自社組織に対するデフォルトの Oracle Workflow ユーザー設定項目を設定します。「グローバル・ワークフロー・プリファレンス」 Web 画面では、ワークフロー管理者と Workflow Web Agent を定義できます。2-12 ページの「[手順 WF-2 グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。
2. 組織のディレクトリ・リポジトリで現在定義されているユーザーとロールに、Oracle Workflow のディレクトリ・サービスをマッピングします。そのためには、これらのデータベース表に基づいてビューを作成します。通知システムでは、これらのビューを使用してアクティビティに指定されている実行者に通知を送信します。ロールは、個々のユーザーまたはグループです。Oracle Workflow には、ディレクトリ・サービスのサンプル・ビューが用意されているため、それを変更して再ロードできます。2-20 ページの「[手順 WF-3 Oracle Workflow のディレクトリ・サービスの設定](#)」を参照してください。
3. Oracle データベース・サーバーのインストールで定義した言語を識別する WF\_LANGUAGES ビューを作成します。Oracle Workflow では、このビューを使用して、インストールされている言語別に、変換前のベース表に見つかった行にマッピングする行を変換表に作成します。2-36 ページの「[手順 WF-5 WF\\_LANGUAGES ビューの作成](#)」を参照してください。
4. ワークフロー・サーバーが UNIX プラットフォームにインストールされている場合は、環境変数 WF\_RESOURCES を定義します。2-40 ページの「[手順 WF-7 環境変数 WF\\_RESOURCES の設定](#)」を参照してください。
5. システム上のメインのワークフロー・エンジンの負荷とスループットを管理できるように、バックグラウンドのワークフロー・エンジンを設定します。メインのエンジンとバックグラウンド・エンジンのコストのしきい値レベルを指定して、エンジンで処理するアクティビティと延期するアクティビティを決定します。2-40 ページの「[手順 WF-8 バックグラウンドのワークフロー・エンジンの設定](#)」を参照してください。

### Oracle Applications embedded Workflow での設定手順の概要（必須）

1. 「グローバル・ワークフロー・プリファレンス」 Web 画面を使用して、自社組織に対するデフォルトの Oracle Workflow ユーザー設定項目を設定します。「グローバル・ワークフロー・プリファレンス」 Web 画面では、ワークフロー管理者と Workflow Web Agent を定義できます。2-12 ページの「[手順 WF-2 グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。



2. システム・プロファイル・オプションで「ソケット・リスナー有効化」および「Socket Listener Port」を設定します。2-38 ページの「[手順 WF-6 ソケット・リスナー・プロファイル・オプションの設定](#)」を参照してください。
3. システム上のメインのワークフロー・エンジンの負荷とスループットを管理できるように、バックグラウンドのワークフロー・エンジンを設定します。メインのエンジンとバックグラウンド・エンジンのコストのしきい値レベルを指定して、エンジンで処理するアクティビティと延期するアクティビティを決定します。2-40 ページの「[手順 WF-8 バックグラウンドのワークフロー・エンジンの設定](#)」を参照してください。

---

**注意：** Oracle Workflow のインストールでは、次の情報が自動的に設定されますが、該当するセクションを参照してバックグラウンド情報を追加することもできます。

2-20 ページ「[手順 WF-3 Oracle Workflow のディレクトリ・サービスの設定](#)」

2-36 ページ「[手順 WF-5 WF\\_LANGUAGES ビューの作成](#)」

2-40 ページ「[手順 WF-7 環境変数 WF\\_RESOURCES の設定](#)」

---

## オプションの設定手順

1. WF\_ITEM\_ACTIVITY\_STATUSES、WF\_ITEM\_ACTIVITY\_STATUSES\_H、WF\_ITEM\_ATTRIBUTE\_VALUES および WF\_ITEMS 表をパーティション化すると、パフォーマンスを改善できます。2-11 ページの「[手順 WF-1 ワークフロー表のパーティション化](#)」を参照してください。
2. Oracle9i Database Server リリース 2 または Oracle9i Application Server (Oracle9iAS) リリース 2 以降で Oracle Workflow のスタンドアロン版を使用している場合は、Workflow ディレクトリ・サービスのユーザー情報を Oracle Internet Directory (OID) に同期させることができます。また、Oracle9iAS リリース 2 以降をインストールしている場合は、OID 統合を使用してシングル・サインオン統合を実装することもできます。2-29 ページの「[手順 WF-4 Workflow ディレクトリ・サービスと Oracle Internet Directory の同期](#)」を参照してください。
3. ユーザーが電子メールで通知を受け取れるようにする場合は、通知メーラー・プログラムを設定します。2-45 ページの「[手順 WF-9 通知メーラーの導入](#)」を参照してください。
4. 電子メール通知用のテンプレートを変更できます。2-65 ページの「[手順 WF-10 メッセージ・テンプレートの変更](#)」を参照してください。
5. Oracle Workflow の Web 画面に表示される会社のロゴをカスタマイズします。2-79 ページの「[手順 WF-11 Oracle Workflow の Web 画面のロゴのカスタマイズ](#)」を参照してください。

6. Oracle Workflow のアイコン・サブディレクトリにアイコンを追加して、ワークフロー・プロセスのダイアグラム表示をカスタマイズできます。定義した各アクティビティに、カスタム・アイコンを使用できます。2-80 ページの「[手順 WF-12 Oracle Workflow へのカスタム・アイコンの追加](#)」を参照してください。
7. Oracle Workflow のスタンドアロン版を使用しているときに外部の Java 関数アクティビティを実行する場合は、Java 関数アクティビティ・エージェントを設定します。2-81 ページの「[手順 WF-13 Java 関数アクティビティ・エージェントの設定](#)」を参照してください。
8. イベント・サブスクリプション処理とワークフロー・プロセス・イベント・アクティビティを使用してシステム間でビジネス・イベントを伝達する場合は、ビジネス・イベント・システムを設定します。2-90 ページの「[手順 WF-14 ビジネス・イベント・システムの設定](#)」を参照してください。
9. Oracle8i でビジネス・イベント・システムを使用しているときに、Oracle Message Broker を使用してシステム間でイベント・メッセージを伝播する場合は、WF\_EVENT\_OMB\_QH キュー・ハンドラを設定します。2-94 ページの「[手順 WF-15 WF\\_EVENT\\_OMB\\_QH キュー・ハンドラの設定](#)」を参照してください。

## その他のワークフロー機能

Oracle Workflow とカスタム・プロセス定義を他の支社に展開する前に、ユーザーがそのデータにアクセスできるレベルを指定して、データが変更されるのを防ぐことができます。2-95 ページの「[Oracle Workflow のアクセス保護の概要](#)」を参照してください。

ワークフロー定義ローダーを使用すると、Oracle Workflow Builder を使用せずに、フラット・ファイルからデータベースにワークフロー・プロセスの定義をロードできます。2-101 ページの「[ワークフロー定義ローダーの使用](#)」を参照してください。

ビジネス・イベント・システムを使用している場合は、Workflow XML Loader を使用して、フラット・ファイルとデータベース間でビジネス・イベント・システム・オブジェクトの XML 定義をロードすることができます。2-106 ページの「[Workflow XML Loader の使用](#)」を参照してください。

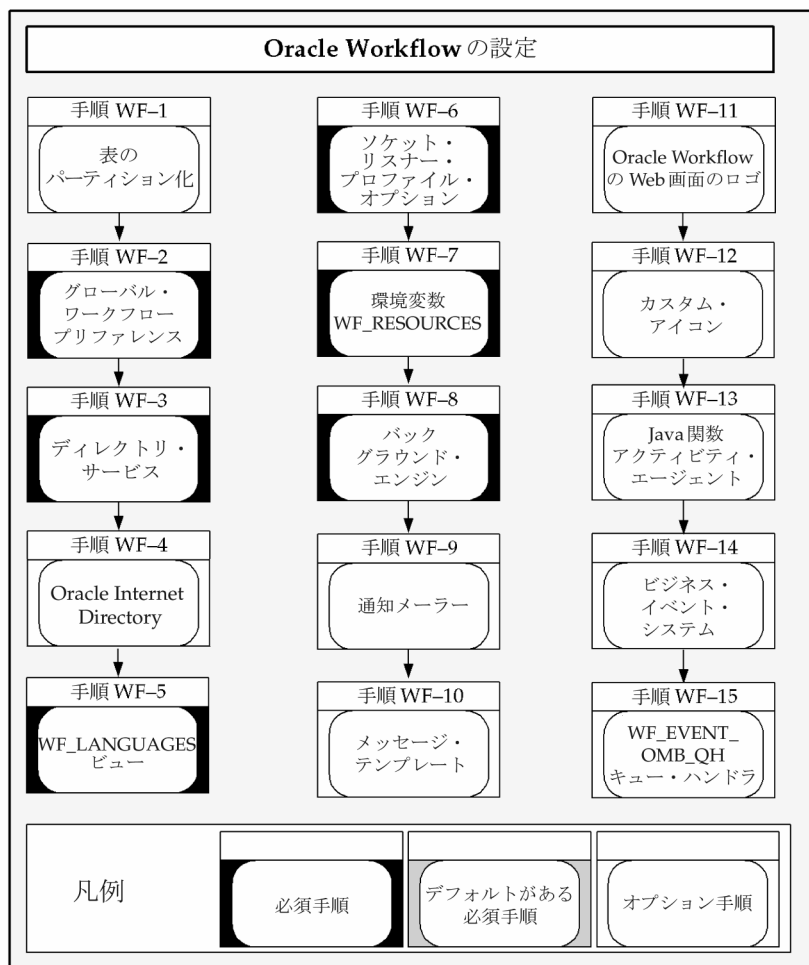
## Oracle Workflow Server のバージョンの確認

実行中の Oracle Workflow Server のバージョンを判断する場合は、SQL\*Plus を使用してスクリプト wfver.sql を実行すると、Oracle Workflow Server のアカウントに接続できます。16-17 ページの「[wfver.sql](#)」を参照してください。

また、ワークフロー定義ローダー、Oracle Workflow Builder、通知メーラー、ワークフロー・モニターなど、Oracle Workflow のすべてのモジュールでは、稼働している Oracle Workflow Server のバージョンとの互換性が自動的に検証されます。このバージョンの互換性チェックを使用すると、Oracle Workflow 2.0.3 データベース上で Oracle Workflow Builder 2.6 を実行するなどの問題を回避できます。

## 設定フローチャート

次のフローチャートは、Oracle Workflow の設定手順を示しています。これらの手順の一部は必須であり、一部はオプションです。オプションの手順を実行する必要があるのは、その関連機能や特定のビジネス機能全体を使用する計画がある場合のみです。



設定チェックリスト

次の表は、Oracle Workflow の設定手順を示しています。手順が Oracle Workflow のスタン  
ドアロン版と埋込み版のどちらに関連しているか、またオプションか必須かを明記してあり  
ます。

手順番号	必須	手順	スタンダアロン /埋込み/両方
手順 WF-1	オプション	2-11 ページ「 <a href="#">手順 WF-1 ワークフロー表のパ ーティション化</a> 」	両方
手順 WF-2	必須	2-12 ページ「 <a href="#">手順 WF-2 グローバル・ワークフ ロー・プリファレンスの設定</a> 」	両方
手順 WF-3	必須	2-20 ページ「 <a href="#">手順 WF-3 Oracle Workflow の ディレクトリ・サービスの設定</a> 」	スタンダアロン
手順 WF-4	オプション	2-29 ページ「 <a href="#">手順 WF-4 Workflow ディレク トリ・サービスと Oracle Internet Directory の同 期</a> 」	スタンダアロン
手順 WF-5	必須	2-36 ページ「 <a href="#">手順 WF-5 WF_LANGUAGES ビューの作成</a> 」	スタンダアロン
手順 WF-6	必須	2-38 ページ「 <a href="#">手順 WF-6 ソケット・リスナー・ プロファイル・オプションの設定</a> 」	埋込み
手順 WF-7	必須	2-40 ページ「 <a href="#">手順 WF-7 環境変数 WF_ RESOURCES の設定</a> 」	スタンダアロン
手順 WF-8	必須	2-40 ページ「 <a href="#">手順 WF-8 バックグラウンドの ワークフロー・エンジンの設定</a> 」	両方
手順 WF-9	オプション	2-45 ページ「 <a href="#">手順 WF-9 通知メーラーの導入</a> 」	両方
手順 WF-10	オプション	2-65 ページ「 <a href="#">手順 WF-10 メッセージ・テン プレートの変更</a> 」	両方
手順 WF-11	オプション	2-79 ページ「 <a href="#">手順 WF-11 Oracle Workflow の Web 画面のロゴのカスタマイズ</a> 」	両方
手順 WF-12	オプション	2-80 ページ「 <a href="#">手順 WF-12 Oracle Workflow への カスタム・アイコンの追加</a> 」	両方
手順 WF-13	オプション	2-81 ページ「 <a href="#">手順 WF-13 Java 関数アクティ ビティ・エージェントの設定</a> 」	スタンダアロン
手順 WF-14	オプション	2-90 ページ「 <a href="#">手順 WF-14 ビジネス・イベン ト・システムの設定</a> 」	両方
手順 WF-15	オプション	2-94 ページ「 <a href="#">手順 WF-15 WF_EVENT_OMB_QH キュー・ハンドラの設定</a> 」	両方

設定手順

手順 WF-1    ワークフロー表のパーティション化

大きな表や索引を使用するときの主要な問題を解決するには、パーティションと呼ばれる管理しやすい小さな単位に表を分割（パーティション化）します。パーティション表にアクセスするために、SQL 問合せと DML 文を変更する必要はありませんが、パーティションを定義すると、DDL 文は表または索引全体ではなく、個々のパーティションにアクセスして操作できるようになります。このように、パーティション化により、大きなデータベース・オブジェクトの管理を簡素化できます。また、パーティション化はアプリケーションに対して完全に透過的です。

スクリプトを実行して、ランタイム・ステータス・データを格納する特定のワークフロー表をパーティション化することもできます。このスクリプトは、Oracle Applications embedded Workflow の場合は wfupartb.sql、Oracle Workflow のスタンドアロン版の場合は wfupart.sql と呼ばれます。パフォーマンス改善のため、この手順を実行することをお勧めします。

このスクリプトにより、4 つのワークフロー表がパーティション化され、関連する索引が再作成されます。次の表に、このスクリプトの対象となるワークフロー表と索引を示します。

表 2-1	
表	索引
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

パーティション化スクリプトが失敗した場合にリストアできるように、スクリプトを実行する前に、前述の 4 つの表のバックアップを作成する必要があります。

このスクリプトを実行するには、表と索引の表領域に十分な空き領域が必要です。このスクリプトでパーティション表を作成中には、基礎となる表と同じ表領域内に、表より少し大きいディスク領域が必要となります。同様に、索引の表領域にも十分な空き領域が必要です。

また、スクリプト完了までに十分な時間を見込む必要があります。所要時間は、表データの量に応じて異なります。旧リリースからのアップグレード後のように、表に既存のデータが含まれている場合は、Oracle Workflow の初回インストール後のように表が空の場合に比べて、スクリプトの完了までに時間がかかります。最短時間で完了するには、このスクリプトを設定プロセスのできるだけ早い段階で実行します。

---

**注意：** Oracle Net Services を介してパーティション化スクリプトを実行する場合は、開始前に TWO\_TASK 変数を設定する必要があります。

---

Oracle Applications embedded Workflow の場合、wfupartb.sql スクリプトは \$FND\_TOP の admin/sql サブディレクトリにあります。このスクリプトの使用 방법은、次のとおりです。

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

たとえば、次のようになります。

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

Oracle Workflow スタンドアロン版の場合、wfupart.sql スクリプトは Oracle ホーム内の wf/admin/sql サブディレクトリにあります。このスクリプトの使用 방법은、次のとおりです。

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

たとえば、次のようになります。

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

パーティション化スクリプトが失敗した場合は、必要なクリーン・アップを手動で実行する必要があります。このスクリプトによる操作は nologging モードでの DDL 操作のため、ロールバックはできません。

**コンテキスト：**この手順を実行する必要があるのは、1 度のみです。

### 関連項目：

C-7 ページ「パフォーマンス改善のためのパーティション化」

## 手順 WF-2 グローバル・ワークフロー・プリファレンスの設定

「ユーザー設定項目」Web 画面から設定できるユーザー設定項目を指定し、Oracle Workflow との対話方法を制御できます。ワークフロー管理者は、「グローバル・ワークフロー・プリファレンス」Web 画面にアクセスできます。この Web 画面では、会社全体のユーザー設定項目のデフォルト値をグローバルに設定できます。各ユーザーは、「ユーザー設定項目」Web 画面でユーザー設定項目の値を変更して、いつでもデフォルトのユーザー設

定項目を上書きできます。どちらの Web 画面にも Oracle Workflow ホーム・ページからアクセスできますが、「グローバル・ワークフロー・プリファレンス」ページにアクセスできるのはワークフロー管理者のみです。

---

**注意：**「グローバル・ワークフロー・プリファレンス」および「ユーザー設定項目」Web 画面の言語、地域および通知の各設定は、ディレクトリ・サービス・ビューによって、「Language」、「Territory」および「Notification\_Preference」の各列が Oracle Workflow 設定項目表にマップされている場合のみ有効です。他の設定ソースにマップしている場合や、ハードコード化された値をこれらの列に設定している場合には、設定用の Web 画面経由で行った変更はすべて無視されます。2-20 ページの「[手順 WF-3 Oracle Workflow のディレクトリ・サービスの設定](#)」を参照してください。

---

**コンテキスト：**この手順を実行する必要があるのは、1 度のみです。

**参照：**9-6 ページの「[ユーザー設定項目の設定](#)」を参照してください。

## ► グローバル・ワークフロー・プリファレンスの設定

1. Web ブラウザを使用して Oracle Workflow ホーム・ページに接続し、次のように「グローバル・ワークフロー・プリファレンス」のリンクを選択します。

```
<webagent>/wfa_html.home
```

または、次のように「グローバル・ワークフロー・プリファレンス」Web 画面に直接接続します。

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

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。

---

**注意：**これらのページにはセキュリティが適用されるため、現行の Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。

---



2. 「グローバル・ワークフロー・プリファレンス」Web 画面に、現在のグローバル・ワークフロー・プリファレンスの要約が表示されます。「更新」を選択し、この設定項目を変更します。



グローバル・ワークフロー・プリファレンス - OWF\_MGR - Netscape

ファイル(F) 編集(E) 表示(V) ジャンプ(G) Communicator(C) ヘルプ(H)

グローバル・ワークフロー・プリファレンス

ワークフロー管理者: \*

Workflow Web Agent: [http://www.aaa\\_url.com/wf/owa](http://www.aaa_url.com/wf/owa)

Initiator Classid: ff348b6e-fd21-11d4-a3f0-00c04fa32518

Initiator Download Location: [http://www.aaa\\_url.com/plugins/jinit.exe](http://www.aaa_url.com/plugins/jinit.exe)

Initiator Version: 1.1.8.6

ローカル・システム: WF91

システムの状態: 使用可能

LDAPホスト:

LDAPポート:

LDAPユーザー名:

LDAPパスワード:

LDAP ChangeLogのベース・ディレクトリ:

LDAPユーザー・ベース・ディレクトリ:

言語: JAPANESE [JAPANESE-JAPAN]

地域: JAPAN [JAPAN-JAPANESE]

日付書式:

文書ホーム・ノード:

電子メール 通知を送信してください: HTMLメール

OK 取消

ドキュメント: 完了。

- 「ワークフロー管理者」フィールドで値リストを使用し、ワークフロー管理者権限を割り当てるロールを選択します。このロールを割り当てられたユーザーは、「Oracle Workflow プロセス検索」Web 画面を実行でき、Oracle Workflow の管理機能へのフルアクセス権が付与されます。さらに、管理ロール内のユーザーは他の全ユーザーの通知を閲覧し、「イベント・マネージャ」Web 画面にアクセスできます。

開発環境のように、すべてのユーザーとロールにワークフロー管理者権限を付与する場合は、「ワークフロー管理者」フィールドにアスタリスク (\*) を入力します。2-20 ページの「手順 WF-3 Oracle Workflow のディレクトリ・サービスの設定」を参照してください。

---

---

**注意：**「グローバル・ワークフロー・プリファレンス」画面にアクセスしないで、ワークフロー管理者権限が現在付与されているロールを確認するには、次のコマンドを使用します。

---

---

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

Oracle Workflow のインストール後に、「ワークフロー管理者」の環境設定をデフォルトの設定から管理者権限を付与するロールに変更する必要があります。

- Oracle Workflow のスタンドアロン版の場合、インストール後のデフォルトの設定はアスタリスク (\*) です。任意のユーザーでログインして、「グローバル・ワークフロー・プリファレンス」画面にアクセスし、必要な環境設定を指定することができます。
- Oracle Applications embedded Workflow の場合、インストール後のデフォルトの設定は SYSADMIN です。「グローバル・ワークフロー・プリファレンス」画面にアクセスし、必要な環境設定を指定するには、SYSADMIN でログインする必要があります。

---

---

**注意：** SYSADMIN ロールは、Oracle Applications の職責「システム管理者」に関連付けられているロールとは異なります。ワークフロー管理者権限を Oracle Applications のこの職責またはその他の職責に割り当てる場合は、「ワークフロー管理者」の環境設定をその職責に関連付けられているワークフローのロールの内部名に設定する必要があります。

---

---

WF\_ROLES ビューを問い合せて、職責のロール名を検索することができます。たとえば、Oracle Applications の様々な管理者の職責を検索するには、次のコマンドを使用します。

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

「ワークフロー管理者」の環境設定を職責のロール名に設定する場合、その職責を持つすべての Oracle Applications ユーザーにはワークフロー管理者権限が付与されます。

4. 「Workflow Web Agent」フィールドに、Oracle HTTP Server で Oracle Workflow に対して定義した Oracle Web エージェントのベース URL を入力します。

---

**注意：** このフィールドに Oracle Workflow Web Agent のベース URL を指定しないと、Oracle Workflow の多くの Web 画面で表示されている値リストの各フィールドが正しく機能しくなくなります。

---

Web サーバーとして Oracle HTTP Server を使用している場合、ベース URL は次のようになります。

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

`<server.com:portID>` は、Web リスナーが要求を受け取るサーバーと TCP/IP ポート番号を表し、`<DAD_name>` は Oracle Workflow のデータベース・スキーマ用に構成された DAD の名前を表します。

詳細は、Oracle HTTP Server のドキュメントを参照してください。

---

**注意：** Oracle Applications embedded Workflow を使用している場合は、APPS\_WEB\_AGENT のプロファイル・オプションも編集する必要があります。

---

5. Oracle Applications embedded Workflow を使用している場合は、JInitiator プラグインのクラス ID、ダウンロード先およびバージョン番号を入力します。これらの情報は、Oracle Workflow が通知にリンクされた Oracle Applications フォームを起動するとき、およびワークフロー・モニターを表示するときに必要です。2-38 ページの「[手順 WF-6 ソケット・リスナー・プロファイル・オプションの設定](#)」を参照してください。

インストールした JInitiator のクラス ID とバージョン番号は、jinit-version.txt ファイル (`<drive>:\Program Files\Oracle\jinit<version>\doc\jinit-version.txt`) に記載されています。ダウンロード先は、ユーザーのクライアント・マシンにダウンロードする JInitiator の実行ファイルを配置した場所です。詳細は、MetaLink で利用できる『Complete Guide to JInitiator for Oracle's E-Business Suite (Note 162488.1)』および『Upgrading Oracle JInitiator with Oracle Applications 11i (Note 124606.1)』を参照してください。

6. 「ローカル・システム」フィールドに、Oracle Workflow がインストールされているデータベースのシステム名が表示されます。Oracle Workflow のイベント・マネージャには、インストール時にこのデータベースのシステム定義が自動的に作成されます。ビジネス・イベント・システムでは、このシステムをローカル・システムとみなし、他のシステムをすべて外部システムとみなします。13-18 ページの「[システム](#)」を参照してください。

---

---

**注意：**「ローカル専用」設定は、各 Oracle Workflow インストールに固有です。ビジネス・イベント・システムのデータを他のシステムにレプリケートした場合は、個別に設定する必要があります。

---

---

7. 「システムの状態」フィールドで、ローカル・システムに割り当てるビジネス・イベント・システムのステータスを値リストから選択します。
- 使用可能： サブスクリプションがすべてのイベントで実行されます。
  - ローカル専用： サブスクリプションが、ローカル・システムで発生したイベントでのみ実行されます。
  - 外部のみ： サブスクリプションが、外部システムから着信したイベントでのみ実行されます。
  - 使用不能： サブスクリプションがすべてのイベントで実行されません。

---

---

**注意：** Oracle Workflow では、デフォルトで「使用可能」に設定されます。「システムの状態」設定は、ビジネス・イベント・システムの設定が終了した後でも、イベント処理に必要なステータスに変更できます。

---

---

---

---

**注意：**「システムの状態」設定は、各 Oracle Workflow インストールに固有です。ビジネス・イベント・システムのデータを他のシステムにレプリケートした場合は、個別に設定する必要があります。

---

---

8. Oracle Internet Directory (OID) 同期を実装する場合は、接続先の LDAP ディレクトリの Lightweight Directory Access Protocol (LDAP) サーバー情報を指定します。
- LDAP ホスト： LDAP ディレクトリが存在するホスト。
  - LDAP ポート： ホスト上のポート。
9. OID 同期を実装する場合は、LDAP サーバーへの接続に使用する LDAP ユーザー・アカウントを指定します。この LDAP ユーザー・アカウントには書き込み権限が必要です。
- LDAP ユーザー名： LDAP ユーザー。このユーザー名は、LDAP ディレクトリにバインドする必要があります。たとえば、次のようになります。  
`cn=orcladmin`
  - LDAP パスワード： LDAP のパスワード。パスワードは、暗号化された形式で格納されます。
10. OID 同期を実装する場合は、変更ログおよびユーザー・レコードに使用するディレクトリを指定します。

- LDAP ChangeLog のベース・ディレクトリ： 変更ログが格納される LDAP ノード。たとえば、次のようになります。

cn=changelog

- LDAP ユーザー・ベース・ディレクトリ： ユーザー・レコードの検索先の LDAP ノード。たとえば、次のようになります。

cn=Base, cn=OracleSchemaVersion

11. 「言語」フィールドと「地域」フィールドでは、リストの値を使用して NLS\_LANGUAGE と NLS\_TERRITORY の組合せを選択し、通知セッションのデフォルトの言語依存動作と地域依存書式設定を定義します。
12. 「日付書式」フィールドでは、全ユーザーのワークフロー・データベース・セッションに対してデフォルトの日付書式を定義する Oracle8i 準拠の日付書式を指定します。Oracle8i 準拠の日付書式の例としては、DD-Mon-RRRR があります。日付書式を指定しなかった場合、日付書式のデフォルトは DD-MON-YYYY になります。

---

**注意：** Oracle Workflow では、日付書式に時間書式を指定しなくても、表示される日付によっては時間要素が含まれる場合があります。日付書式とともに時間書式を指定すると、Oracle Workflow で時間要素が表示される場合に、日付の後に 2 つの時間要素が表示されます。

---

13. 「文書ホーム・ノード」フィールドは空白のままにしてください。この機能は今後使用する目的で確保されています。
14. 「電子メール通知を送信してください」フィールドで、次の通知環境設定を値リストから選択します。
  - HTML メール： 通知が HTML 電子メールとして送られてきます。ユーザーがメールを読むには、HTML 形式の電子メール・ビューアを使用する必要があります。
  - HTML 添付ファイル付きのプレーン・テキスト・メール： 通知がプレーン・テキスト電子メールとして送信されますが、添付ファイルとして HTML 形式の通知が添付されます。
  - プレーン・テキスト・メール： 通知がプレーン・テキストの電子メールとして送られてきます。
  - プレーン・テキスト要約メール： すべての通知の要約がプレーン・テキストの電子メールとして送られてきます。個々の通知を処理するには、「通知」Web 画面を使用する必要があります。
  - メールを送信しないでください： 通知は電子メールとして送信されません。通知の表示と処理には、「通知」Web 画面を使用する必要があります。
15. 変更の終了後に「OK」をクリックします。

---

---

**注意：** 言語、地域、文書ホーム・ノードおよび通知のグローバルな設定項目は、`-WF_DEFAULTS-` という特殊なユーザー名で、Oracle Workflow の設定項目表に保存されます。ワークフロー管理者、Workflow Web エージェント、ローカル・システムおよび LDAP の情報は、ワークフロー・リソース表に保存されます。

---

---

### 手順 WF-3 Oracle Workflow のディレクトリ・サービスの設定

Oracle Workflow では、ワークフロー・ユーザーやロールのユーザーを柔軟に定義できます。リポジトリを構成するデータベースの表を基に 3 つのビューを作成し、Oracle Workflow がユーザーとロールの情報を参照するディレクトリ・リポジトリを決定します。この 3 つのビューは次のとおりです。

- WF\_USERS
- WF\_ROLES
- WF\_USER\_ROLES

また、Oracle Workflow には、WF\_LOCAL\_USERS、WF\_LOCAL\_ROLES および WF\_LOCAL\_USER\_ROLES という 3 つのローカル表があり、各表にはそれぞれ WF\_USERS、WF\_ROLES および WF\_USER\_ROLES ビューで定義される列と類似の列があります。これらの表を使用し、適切なディレクトリ・サービス PL/SQL API をコールして、既存のディレクトリ・リポジトリに含まれていないユーザーとロールを保存できます。8-124 ページの「[Workflow ディレクトリ・サービス API](#)」を参照してください。

---

---

**注意：** ディレクトリ・サービス・ビューにユーザーおよびロールを追加するときには、DUAL から選択しないでください。これはビューの特定の列に対する一意の制約に違反し、「select from DUAL」文の間の不要な結合によってパフォーマンスが低下するためです。

---

---

---

---

**注意：** Oracle9i リリース 2 以降または Oracle9iAS リリース 2 以降をインストールした環境で、Oracle Workflow のスタンドアロン版を使用している場合は、Workflow ディレクトリ・サービスをディレクトリ・リポジトリとして Oracle Internet Directory (OID) に同期させることができます。この場合、ディレクトリ・サービス・ビューを WF\_LOCAL 表にのみマップして、Workflow ユーザー情報と OID との同期を有効にする必要があります。2-28 ページの「[Oracle Workflow ディレクトリ・サービスとローカル・ワークフロー・ユーザーの統合](#)」および 2-29 ページの「[手順 WF-4 Workflow ディレクトリ・サービスと Oracle Internet Directory の同期](#)」を参照してください。

---

---

**コンテキスト:** この手順を実行する必要があるのは、1 度のみです。

**参照:** 2-26 ページの「事前に定義されたディレクトリ・サービス」を参照してください。

**参照:** 5-25 ページの「アド・ホックのユーザーおよびロール」を参照してください。

## WF\_USERS

WF\_USERS ビューでは、ワークフロー通知を受信する組織内のすべてのユーザーに関する情報を参照する必要があります。このビューの作成時には、次の列を必ず含めます。

- **Name:** ワークフロー・エンジンと通知システムが参照するユーザーの内部名。たとえば、ユーザーの内部名として mbeech や 009 を使用できます。この場合、009 は、ユーザーの従業員 ID を表します。

---

**注意:** 「Name」列のソースには、長さ 30 文字以内で大文字のみの列を使用する必要があります。この条件と一致する列がソース表になくても、文字列関数を使用してこの制限を回避しないでください。かわりに、「Name」列を `<orig_system>.<orig_system_id>` として定義すると、ユーザーが格納されている元の実表とその表内の特定のユーザーを Oracle Workflow で参照できます。たとえば、「PER\_PEOPLE:009」は、従業員 ID が 009 で、PER\_PEOPLE という人事表に格納されているユーザーを表します。

---

- **Display\_Name:** ユーザーの表示名。表示名の例は「Beech, Matthew」です。
- **Description:** ユーザーの説明（オプション）。
- **Notification\_Preference:** このユーザーの通知の受信方法を指定します。値に MAILTEXT、MAILHTML または MAILATTH を指定すると、ユーザーはそれぞれプレーン・テキストの電子メール、HTML 形式の電子メールまたは HTML 添付ファイル付きのプレーン・テキストの電子メールで、通知を受信して応答できます。値に QUERY を指定すると、ユーザーは「通知」Web 画面から通知を問合せできます。また、値に「SUMMARY」を指定すると、ユーザーは処理中の通知の要約を示す電子メールを定期的に受信できます。ただし、個々の通知に応答するには、その通知を「通知」Web 画面で問い合わせる必要があります。10-2 ページの「通知処理の概要」および 2-46 ページの「通知環境設定」を参照してください。

---

**注意:** 通知環境設定が MAILTEXT、MAILHTML または MAILATTH の場合、ユーザーは自分の通知を「通知」Web 画面から問い合わせることもできます。

---

---

---

**注意：** 次の文を使用して、Oracle Workflow 環境設定表に「Notification\_Preference」列をマップできます。これにより、「グローバル・ワークフロー・プリファレンス」Web 画面を使用して、社内の全ユーザーにデフォルトの通知環境設定をグローバルに設定でき、各ユーザーは「ユーザー設定項目」Web 画面の通知環境設定を変更して、このデフォルト値を上書きできます。2-12 ページの「[手順 WF-2 グローバル・ワークフロー・プリファレンスの設定](#)」、9-6 ページの「[ユーザー設定項目の設定](#)」および 8-152 ページの「[get\\_pref](#)」を参照してください。

---

---

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

- 言語： データベースの NLS\_LANGUAGE 初期化パラメータの値。この値では、ユーザーの通知セッションのデフォルトの言語依存動作を指定します。サポートされている言語の表記法一覧は、Oracle データベースのユーザーズ・ガイドまたはインストール・マニュアルを参照してください。

---

---

**注意：** 「グローバル・ワークフロー・プリファレンス」Web 画面で言語を指定すると、社内の全ユーザーに言語をグローバル設定できます。各ユーザーは、「ユーザー設定項目」Web 画面で言語を変更し、このデフォルト値を上書きできます。2-12 ページの「[手順 WF-2 グローバル・ワークフロー・プリファレンスの設定](#)」、9-6 ページの「[ユーザー設定項目の設定](#)」および 8-152 ページの「[get\\_pref](#)」を参照してください。

---

---

---

---

**注意：** 次の文を使用して、Oracle Workflow 環境設定表に「Language」列をマップできます。これにより、「グローバル・ワークフロー・プリファレンス」Web 画面を使用して、社内の全ユーザーにデフォルトの言語をグローバルに設定でき、各ユーザーは「ユーザー設定項目」Web 画面の言語を変更して、このデフォルト値を上書きできます。2-12 ページの「[手順 WF-2 グローバル・ワークフロー・プリファレンスの設定](#)」および 9-6 ページの「[ユーザー設定項目の設定](#)」を参照してください。

---

---

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



---

---

**注意：** 通知の送信時に通知メーラーによって使用される電子メール・テンプレートが、設定する言語に変換されていることを必ず確認してください。電子メール・テンプレートは、サブディレクトリ

\$ORACLE\_HOME/wf/res/<lang> の下にある wfmail.wft というファイルで送信されます。適切な言語サブディレクトリをチェックし、設定する言語にテンプレートが変換されているかどうかを確認します。2-65 ページの「[手順 WF-10 メッセージ・テンプレートの変更](#)」を参照してください。

---

---

- 地域： データベースの NLS\_TERRITORY 初期化パラメータの値。この値では、ユーザーの通知セッションで使用される地域依存の書式のデフォルト値を指定します。サポートされている地域の表記法一覧は、Oracle データベースのユーザーズ・ガイドまたはインストレーション・マニュアルを参照してください。

---

---

**注意：** 次の文を使用して、Oracle Workflow 環境設定表に「Territory」列をマップできます。これにより、「グローバル・ワークフロー・プリファレンス」Web 画面を使用して、社内の全ユーザーにデフォルトの地域をグローバルに設定でき、各ユーザーは「ユーザー設定項目」Web 画面の地域を変更して、このデフォルト値を上書きできます。2-12 ページの「[手順 WF-2 グローバル・ワークフロー・プリファレンスの設定](#)」、9-6 ページの「[ユーザー設定項目の設定](#)」および 8-152 ページの「[get\\_pref](#)」を参照してください。

---

---

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

- Email\_Address: ユーザーの有効な電子メール・アドレス、または電子メール・システムで定義されているメール配布リスト。
- Fax: ユーザーの fax 番号。
- Orig\_System: ビューの元になっているディレクトリ・リポジトリに割り当てるコード。たとえば、このビューが人事管理システムに保存されている個人データを元になっている場合は、Orig\_System を PER として定義できます。
- Orig\_System\_ID: リポジトリ・システムのユーザーを識別する主キー。たとえば、Orig\_System\_ID を、PER\_PEOPLE という人事管理データベース表の PERSON\_ID 列に格納されている値として定義できます。
- Status: ワークフロー・プロセスに参加するユーザーの使用可能ステータス。考えられるステータスには、アクティブ (ACTIVE)、長時間使用不可 (EXTLEAVE)、永久的に使用不可 (INACTIVE)、一時的に使用不可 (TMPEAVE) があります。これらのステータスは、WFSTD\_AVAILABILITY\_STATUS という選択肢タイプにも保存されています。

- **Expiration\_Date:** ディレクトリ・サービスにおいて、ユーザーが有効でなくなる日付。

**WF\_ROLES**

WF\_ROLES ビューでは、Workflow 通知を受信する組織内のすべてのロールに関する情報を参照する必要があります。このビューの作成時には、リポジトリ内のロールに関する次の列を含める必要があります。アスタリスク (\*) で始まる列は、WF\_USERS ビューに関して前述した同じ列と同様です。

---

---

**注意：** WF\_USERS で識別される各ユーザーも、ロールとして定義しておく必要があります。

---

---

---

---

**注意：** ユーザーがロールのメンバーであり、そのユーザーの情報がロール情報とは異なる場合、通知システムによってロールに通知が配信される時に、ユーザー情報がロール情報で上書きされます。たとえば、あるユーザーの通知環境設定に SUMMARY が設定されていて、同時にそのユーザーが通知環境設定に MAILHTML が設定されているマルチユーザー・ロールのメンバーを兼ねている場合を考えます。マルチユーザー・ロールに通知が割り当てられると、このユーザーはその通知が含まれる要約メッセージではなく、ロールに送信された 1 つの通知メッセージを受信します。

---

---

- **Name:** ロールの内部名。Name 列のソースには、長さ 30 文字以内で大文字のみの列を使用する必要があります。この条件と一致する列がソース表になくても、文字列関数を使用してこの制限を回避しないでください。かわりに、「Name」列を `<orig_system>:<orig_system_id>` として定義すると、ロールが格納されている元の実表とその表内の特定のロールを Oracle Workflow で参照できます。たとえば、「PER\_POSITION:009」は、ID が 009 で、PER\_POSITION という人事表に格納されている職階を表します。
- \*Display\_Name
- \*Description
- \*Notification\_Preference
- \*Language
- \*Territory
- **Email\_Address:** 指定したロールについてメール・アドレスが NULL の場合、通知メーラーはロール内の各ユーザー宛てに個別の電子メールを送信します。
- \*Fax
- \*Orig\_System

- \*Orig\_System\_ID
- \*Status
- \*Expiration\_Date

## WF\_USER\_ROLES

WF\_USER\_ROLES ビューは、WF\_USERS および WF\_ROLES 内のユーザーとロールの共通部分です。このビューの作成時には、次の列を必ず含めます。

- User\_Name: WF\_USERS ビューに表示されているユーザーの内部名
- User\_Orig\_System: WF\_USERS ビューに表示されているユーザー・ディレクトリ・リポジトリに割り当てるコード
- User\_Orig\_System\_ID: WF\_USERS ビューに表示されているユーザー・ディレクトリ・リポジトリ内のユーザーを識別する主キー
- Role\_Name: WF\_ROLES ビューに表示されているロールの内部名
- Role\_Orig\_System: WF\_ROLES ビューに表示されているロール・ディレクトリ・リポジトリに割り当てるコード
- Role\_Orig\_System\_ID: WF\_ROLES ビューに表示されているロール・ディレクトリ・リポジトリ内のロールを識別する主キー

---

---

**注意：** ユーザーの間合せ時に一意の索引を利用するには、データベースにすべて大文字でユーザー名を入力してください。ビュー定義でユーザー名を大文字に変換すると、ビューへのアクセス・パフォーマンスが低下します。

---

---

---

---

**警告：** データベース・パフォーマンスが低下するため、UNION を含むビューへの結合は作成しないでください。このような結合を作成すると、Oracle データベース・サーバーではビューの索引を保存できません。ほとんどの場合、UNION を含むワークフロー・ディレクトリ・サービスのビューを作成することになるため、これらのビューに直接結合しないでください。ディレクトリ・サービスの3つのビューからデータを取り出す必要がある場合は、該当するディレクトリ・サービス API を使用してください。8-124 ページの「[Workflow ディレクトリ・サービス API](#)」を参照してください。

---

---

## 事前に定義されたディレクトリ・サービス

Oracle Workflow には、3つのディレクトリ・サービス環境のうちの1つを実現するためのスクリプトが用意されています。Oracle Applications embedded Workflow を使用している場合は、次の処理が自動的に実行されます。

- Oracle Workflow ディレクトリ・サービスと Oracle Applications の統一環境との統合

Oracle Workflow のスタンドアロン版を使用する場合は、次の2つのディレクトリ・サービスのどちらかを選択するか、または独自に作成できます。

- ネイティブ Oracle ユーザーとディレクトリ・サービスの統合
- ローカルのワークフロー・ユーザーとディレクトリ・サービスの統合

これらのディレクトリ・サービス環境は、それぞれのスクリプトを編集してワークフロー・サーバーに対して再実行すると、さらにカスタマイズできます。

---

**注意：** 独自のディレクトリ・サービスを作成するか、前述の事前に定義されたディレクトリ・サービスを編集する場合は、wfdirchk.sql スクリプトを実行し、ディレクトリ・サービスのデータ・モデルを検証する必要があります。このスクリプトは、Oracle Workflow のスタンドアロン版の場合はサーバー上の Oracle Workflow の admin/sql サブディレクトリに、Oracle Applications embedded Workflow の場合は \$FND\_TOP の sql サブディレクトリに格納されています。16-10 ページの「[wfdirchk.sql](#)」を参照してください。

---

### ► Oracle Workflow ディレクトリ・サービスと Oracle Applications の統一環境の統合

Oracle Applications embedded Workflow を使用する場合は、Oracle Workflow ディレクトリ・サービス・ビューの基礎として Oracle Applications 統一環境が自動的に使用されます。統一環境には、Oracle Human Resources 表、Oracle Application Object Library 表、Oracle Applications の各種の表および WF\_LOCAL 表がマップされます。

Oracle Workflow には、この統一環境にマップする WF\_USERS、WF\_ROLES および WF\_USER\_ROLES ビューを定義する sql スクリプトが用意されています。このスクリプトは、Oracle Applications の導入時に自動的にインストールされ、統一環境が作成されます。ただし、このスクリプトをなんらかの理由で編集して実行し直す必要がある場合、このスクリプトは wfdirhrv.sql という名前でサーバー上の \$FND\_TOP の admin/sql サブディレクトリに格納されています。

WF\_LOCAL\_USERS と WF\_LOCAL\_ROLES に格納されているユーザーおよびロールの他に、統一環境のすべてのワークフロー・ユーザーとロールのデフォルト通知環境設定は MAILHTML に設定されます。

### ► Oracle Workflow ディレクトリ・サービスとネイティブ Oracle ユーザーの統合

Oracle Workflow のスタンドアロン版を使用する予定であれば、ディレクトリ・サービスを Oracle RDBMS 内でネイティブのユーザーとロールにマップできます。各ビューの基礎とし

て DBA\_USERS、WF\_LOCAL\_USERS、DBA\_ROLES および WF\_LOCAL\_ROLES 表を使用します。

Oracle Workflow には、ビューの設定に使用可能なスクリプトが用意されています。サーバー上の、Oracle Workflow の sql サブディレクトリにある wfdirouv.sql スクリプトを使用します。このスクリプトは、Oracle Workflow のスタンドアロン版をインストールするときに Oracle Universal Installer によって自動的に実行されます。このスクリプトは、必要に応じてカスタマイズして再実行できます。スクリプトでは、3 つのビューが作成されます。

WF\_USERS ビューでは、各 DBA ユーザーと、WF\_LOCAL\_USERS に格納されている任意のユーザーに対してワークフロー・ユーザーが作成されます。各 DBA ユーザーの場合、元のシステムは ORACLE と呼ばれ、元のシステム ID は DBA\_USERS の USERNAME 列になります。各 DBA ユーザーに対するデフォルトの通知環境設定は、MAILHTML です。

WF\_ROLES ビューには、WF\_USERS ビューの全ユーザー、WF\_LOCAL\_ROLES 表で定義される全ロールおよび DBA\_ROLES の全ロールが含まれます。role\_name は WF で始まります。各 DBA ロールの場合、元のシステムは ORACLE と呼ばれ、元のシステム ID は DBA\_ROLES の ROLE 列になります。各 DBA ロールに対するデフォルトの通知環境設定は、MAILHTML です。

WF\_USER\_ROLES ビューは、WF\_USERS と WF\_ROLES 内のユーザーおよびロールの名前と元のシステム情報で構成されます。

---

---

**注意：** wfdirouv.sql のスクリプトでは、各ネイティブ Oracle ユーザーのメール・アドレスが、ユーザー個々のユーザー名に設定されます。最小限の設定手順として、WF\_ROLES ビュー定義を使用して、ネイティブ Oracle ユーザーを既存のメール・ディレクトリ・ストアにリンクする必要があります。ただし、ユーザー名とメール・アカウントが一致する場合には、wfdirouv.sql のスクリプトを編集して、WF\_USERS ビュー定義のユーザー名に組織のドメイン（「@oracle.com」など）を追加してください。通常、変更する列は WF\_USERS の EMAIL\_ADDRESS と WF\_ROLES の EMAIL\_ADDRESS です。

---

---

➤ **Oracle Workflow ディレクトリ・サービスとローカル・ワークフロー・ユーザーの統合**

スタンドアロン版の Oracle Workflow を使用する予定で、ディレクトリ・リポジトリのユーザーおよびロールが既存のデータベース表に格納されていない場合、WF\_LOCAL 表にユーザーおよびロールの情報を入力し、ディレクトリ・サービスをその表にマップできます。

Oracle Workflow には、ビューの設定に使用可能なスクリプトが用意されています。サーバー上の、Oracle Workflow の sql サブディレクトリにある wfdircsv.sql スクリプトを使用します。このスクリプトでは、3 つのビューが作成されます。このスクリプトでビューをカスタマイズし、カスタム・ディレクトリ・リポジトリからユーザーとロールを取り込むことができます。

---

---

**注意：** Oracle Internet Directory (OID) 統合を実装して Oracle Workflow のスタンドアロン版に使用する場合、WF\_LOCAL\_USERS 表のみが OID と同期されるため、wfdircsv.sql スクリプトを実行してディレクトリ・サービス・ビューが WF\_LOCAL 表のみにマップされるようにする必要があります (OID によって管理されるのはユーザーのみです。Workflow ロールは管理されません)。この場合、スクリプトをカスタマイズして他の表を取り込まないでください。OID 統合の実装後は、OID を使用してユーザー情報を管理します。2-29 ページの「[手順 WF-4 Workflow ディレクトリ・サービスと Oracle Internet Directory の同期](#)」を参照してください。

---

---

WF\_USERS ビューの元のシステムは WF\_LOCAL\_USERS で、元のシステム ID は 0 です。

WF\_ROLES ビューには、WF\_LOCAL\_USERS の全ユーザーと、WF\_LOCAL\_ROLES で定義される全ロールが含まれます。元のシステムは WF\_LOCAL\_ROLES で、元のシステム ID は 0 です。

WF\_USER\_ROLES ビューは、WF\_USERS と WF\_ROLES 内のユーザーおよびロールの名前と元のシステム情報で構成されます。

## 手順 WF-4 Workflow ディレクトリ・サービスと Oracle Internet Directory の同期

Oracle9i リリース 2 以降または Oracle9iAS リリース 2 以降をインストールした環境で、Oracle Workflow のスタンドアロン版を使用している場合は、Lightweight Directory Access Protocol (LDAP) を使用して、Workflow ディレクトリ・サービスのユーザー情報を Oracle Internet Directory (OID) に同期させることができます。これにより、ユーザー情報を一箇所から管理および公開することによって様々なシステムを参照できるため、この統合の実装をお薦めします。

OID との同期によって、Oracle Workflow で次の処理が可能になります。

- 作業項目に所有権を割り当て、OID で定義されているユーザーに通知を送信します。
- OID と同期される他の外部ユーザー・ディレクトリと同期します。
- Oracle Portal および Oracle9iAS Single Sign-On Server による LDAP 外部認証を介して、シングル・サインオンに参加します (Oracle9iAS リリース 2 以降をインストールしている場合)。シングル・サインオンに参加している他の Oracle9iAS コンポーネントにログインしているユーザーは、自動的に認証されます。つまり、参加している他のコンポーネントにすでにアクセスしているときは、再度ログインする必要がありません。

**コンテキスト:** この手順を実行する必要があるのは、1 度のみです。

### Oracle Internet Directory

Oracle Internet Directory は、分散したユーザーおよびネットワーク・リソースに関する情報の高速検索と集中管理を可能にする汎用ディレクトリ・サービスです。Oracle Internet Directory は、Lightweight Directory Access Protocol (LDAP) Version 3 と、Oracle9i の優れたパフォーマンス、拡張性、信頼性および可用性を結合します。

LDAP は、拡張性のある標準的なディレクトリ・アクセス・プロトコルで、LDAP のクライアントとサーバーの共通の言語として使用されます。LDAP は、ディレクトリ・サービスに関する International Standardization Organization (ISO) X.500 規格の、インターネット版軽量実装として考案されました。このプロトコルは、クライアント側に必要なネットワーク・ソフトウェアを最小限に抑えることができるため、特にインターネット・ベースの Thin クライアント・アプリケーションに最適です。

OID には、次のような利点があります。

- 拡張性: Oracle Internet Directory は、Oracle9i の長所を利用して、テラバイトのディレクトリ情報のサポートを可能にします。さらに、マルチスレッド LDAP サーバーやデータベース接続プーリングなどのテクノロジーにより、短い検索応答時間で多数の同時クライアントをサポートすることができます。

Oracle Internet Directory には、Oracle Directory Manager や様々なコマンドライン・ツールなど、大量の LDAP データを操作するためのデータ管理ツールも用意されています。

- 高可用性: Oracle Internet Directory は、各種の重要なアプリケーションのニーズを満たすように設計されています。たとえば、ディレクトリ・サーバー間の完全なマルチマ

スター・レプリケーションをサポートしているため、レプリケーション・コミュニティ内の1つのサーバーが使用不可になった場合でも、別のサーバーのデータにアクセスできます。1つのサーバーのディレクトリ・データに加えられた変更に関する情報は、Oracle9i データベースの特殊な表に格納されます。これらの情報は、信頼性の高い Oracle9i レプリケーションによって、ディレクトリ環境全体にレプリケートされます。

また、Oracle Internet Directory は、Oracle9i の可用性を最大限に利用します。ディレクトリ情報は Oracle9i データベースに安全に格納され、Oracle のバックアップ機能によって保護されます。また、大きなデータストアおよび高負荷で実行される Oracle9i データベースにシステム障害が発生しても、すぐにリカバリできます。

- **セキュリティ：** Oracle Internet Directory は、広範囲に渡る柔軟なアクセス制御を提供します。管理者は特定のディレクトリ・オブジェクトまたはディレクトリ・サブツリー全体のアクセス権限を付与または制限できます。さらに、Oracle Internet Directory は、匿名、パスワード・ベース、Secure Socket Layer (SSL) Version 3 を使用した証明書ベース（認証されたアクセスおよびデータ・プライバシー用）、といった3つのレベルのユーザー認証を実装します。
- **他のディレクトリとの同期：** Oracle Internet Directory には、サード・パーティの LDAP ディレクトリを含む他のエンタープライズ・リポジトリとの同期を可能にする、Oracle Directory Integration Platform が組み込まれています。

Oracle9iAS Single Sign-On は、Oracle Internet Directory を使用してユーザー・エントリを格納します。パートナ・アプリケーションのユーザーを OID エントリのユーザー・エントリにマップし、LDAP メカニズムを使用してそれらを認証します。

### 関連項目：

『Oracle Internet Directory 管理者ガイド』

## Oracle9iAS Single Sign-On

Oracle9iAS Single Sign-On は、安全なシングル・サインオンのフレームワークを提供する Oracle9i Application Server のコンポーネントの1つです。ユーザー名とパスワードを一度入力すれば、複数の Web ベース・アプリケーションにログインすることができます。

Oracle9iAS Single Sign-On には、次の利点があります。

- ユーザー名とパスワードを特定のアプリケーションの外部で保存および管理し、企業全体で共有することができるため、管理が容易になり、管理コストを削減できます。
- アクセスするアプリケーションごとに別個のユーザー名とパスワードを管理する必要がないため、ログインが簡単になります。
- パスワードの入力が一度だけなので、単純で簡単に覚えられるパスワードを使用したり、パスワードをメモしたりする必要がなくなり、セキュリティが強化されます。

Oracle9iAS Single Sign-On の最も重要なテクノロジーは、Login Server です。Login Server はユーザーを認証し、その識別情報を Login Server に統合されたパートナ・アプリケーションに渡します。



パートナ・アプリケーションはシングル・サインオン・メカニズムをサポートしているため、Login Server によって検証されたユーザー名とパスワードを受け入れることができます。パートナ・アプリケーションは、認証を Login Server に委任します。パートナ・アプリケーションが Login Server に登録されている場合、ユーザーはシングル・サインオン・メカニズムを使用してパートナ・アプリケーションにログインできます。

Oracle HTTP Server は、シングル・サインオンを使用可能にする Oracle モジュール (mod\_osso) を使用して、Login Server のパートナ・アプリケーションとして動作します。Oracle Workflow は、Oracle HTTP Server を Web サーバーとして使用します。Oracle Internet Directory/Single Sign-On 統合を実装する場合、Oracle Workflow は mod\_osso を使用して保護されている Web ページへのアクセスを認証します。

保護されている Workflow Web ページにユーザーが初めてアクセスしようすると、Workflow のセキュリティ・パッケージ WFA\_SEC は、ユーザー情報について CGI 環境変数 REMOTE\_USER をチェックします。そのユーザーが Oracle9iAS Single Sign-On に参加しているアプリケーション (Oracle Workflow など) にまだログインしていない場合は、ページが表示される前にログインを求めるプロンプトが表示されます。

---

**注意：** Oracle Internet Directory/Single Sign-On 統合を実装する場合は、インストール後の手順として WFA\_SEC パッケージをロードする必要があります。詳細は『Oracle Workflow Server インストレーション・ノート リリース 2.6.2』を参照してください。

---

変数 REMOTE\_USER を設定するため、Oracle HTTP Server は内部で mod\_osso をコールします。アプリケーション Cookie が存在しない場合、Login Server は Oracle9iAS Single Sign-On パートナ・アプリケーションとして機能します。つまり、mod\_osso はユーザーを透過的に Login Server にリダイレクトして、認証資格証明を取得します。

Login Server は次の手順を実行します。

- ユーザー名とパスワードを求めるプロンプトを表示します (ログイン Cookie が存在しない場合)。
- LDAP 準拠のディレクトリ (Oracle Internet Directory など) に依存する外部リポジトリ認証を使用して、ユーザー名とパスワードでユーザーを認証します。Login Server は OID にバインドし、ディレクトリに格納されているユーザー資格情報を検索します。
- 暗号化されたログイン Cookie を認証されたクライアント上に格納します。
- ユーザーの識別情報が含まれる暗号化パラメータを指定した URL を使用して、ユーザーをパートナ・アプリケーションに透過的にリダイレクトします。

Oracle HTTP Server は、mod\_osso を使用して次の手順を実行します。

- パラメータを復号化します。
- ユーザーを識別します。
- 独自のセッション管理を確立します (ユーザーに付与するアクセス権限の決定など)。

- パートナ・アプリケーション Cookie を設定し、後続のユーザー・アクセスを Login Server にリダイレクトしないようにします。
- 要求されたアプリケーション・ページをユーザーに提供します。

そのユーザーが同一セッション中に同じまたは別のパートナ・アプリケーションにアクセスした場合、Login Server はユーザー名とパスワードの入力を求めません。かわりに、すでにクライアントのブラウザに格納されているログイン Cookie から情報を取得します。ログイン Cookie はユーザーの識別情報を Login Server に提供し、認証がすでに実行されていることを示します。ログイン Cookie が存在しない場合、Login Server はユーザーにログイン画面を表示します。

不正なアクセスから保護するため、Login Server は暗号化 SSL チャネルを介して、ログイン Cookie をクライアントのブラウザに送信します。

ログイン Cookie は、管理者が指定した時間が経過したときまたはユーザーがブラウザを終了したときに、セッションとともに有効期限が切れます。ログイン Cookie はディスクに書き込まれません。

---

---

**注意：** パートナ・アプリケーションからログアウトし、別のユーザーでログインするには、Login Server セッションからもログアウトする必要があります。Login Server セッションからログアウトしないで別のユーザーでログインしようとすると、その認証要求はパートナ・アプリケーションを前のユーザーのログイン状態に戻します。

---

---

### 関連項目：

『Oracle9iAS Single Sign-On 管理者ガイド』

『Oracle9iAS Single Sign-On アプリケーション開発者ガイド』

## Oracle Internet Directory との同期

Oracle Workflow には、Workflow ディレクトリ・サービスのユーザー情報を OID と同期するための API が用意されています。これらの API は、WF\_LDAP という PL/SQL パッケージに定義されています。8-148 ページの「[Workflow LDAP API](#)」を参照してください。

---

---

**注意：** OID 統合には、個々のユーザーのみが含まれ、ユーザー・グループは含まれません。Workflow のルールは OID では管理されません。

---

---

### ► Workflow ディレクトリ・サービスを OID と同期するには

1. LDAP 同期に必要な次の PL/SQL パッケージがデータベースにロードされていることを確認します。

- DBMS\_LDAP: このパッケージには、LDAP サーバーのデータへのアクセスに使用可能な関数およびプロシージャが含まれています。
- WFA\_SEC: このパッケージには、Workflow セキュリティの関数およびプロシージャが含まれています。

Oracle Workflow のスタンドアロン版の場合、これらのパッケージのインストールは、Oracle Workflow Server のインストール後にインストール後の手順の一部として実行されます。

2. シングル・サインオン統合の場合は、Oracle Workflow の Database Access Descriptor が `mod_osso` 構成ファイルで保護されていることを確認します。Oracle Workflow のスタンドアロン版の場合、`mod_osso` の構成は、Oracle Workflow Server のインストール後にインストール後の手順の一部として実行されます。
3. 次のグローバル・ワークフロー・プリファレンスが、OID インストールに適した値に設定されていることを確認します。2-12 ページの「[手順 WF-2 グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。
  - LDAP ホスト
  - LDAP ポート
  - LDAP ユーザー名
  - LDAP パスワード
  - LDAP ChangeLog のベース・ディレクトリ
  - LDAP ユーザー・ベース・ディレクトリ
4. `wfdircsv.sql` スクリプトを実行して、ディレクトリ・サービス・ビューを `WF_LOCAL` 表にのみマップします。2-28 ページの「[Oracle Workflow ディレクトリ・サービスとローカル・ワークフロー・ユーザーの統合](#)」を参照してください。
5. ルール関数 `WF_SSO.user_change` を指定して、`oracle.apps.wf.public.user.change` イベントへの事前定義済みのサブスクリプションを使用可能にします。14-18 ページの「[User Entry Has Changed イベント](#)」および 13-54 ページの「[イベント・サブスクリプションの更新または削除](#)」を参照してください。
6. 同期を開始するには、`WF_LDAP.Synch_all()` API を実行します。この関数は、「グローバル・ワークフロー・プリファレンス」で指定されている LDAP ディレクトリ情報に基づいて、OID から既存のユーザー情報をすべて取得し、`oracle.apps.wf.public.user.change` イベントを呼び出します。事前定義済みのサブスクリプションをこのイベントで使用可能にすると、ユーザー情報が `WF_LOCAL_USERS` 表にロードされます。

`Synch_all()` は、OID に格納されているすべてのユーザーの情報が取得されるため、設定中はこの関数を一度だけ使用してください。ただし、リカバリまたはクリーンアップ時に必要に応じて、`Synch_all()` を実行することもできます。

次のコマンドを使用して、`Synch_all()` を実行します。

```

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. このコマンドの実行後は、変更された OID ユーザー情報のみを取得およびロードすれば、Workflow ディレクトリ・サービスと OID 間の同期を維持できます。ユーザー情報は、10 分おきに更新することをお勧めします。

変更されたユーザー情報は、WF\_LDAP.Synch\_changes() または WF\_LDAP.Schedule\_changes() を使用して OID から取得できます。WF\_LDAP.Synch\_changes() は、作成、変更、削除などの OID で発生した LDAP ユーザーの変更を識別するために、LDAP 変更ログ・レコードを問い合わせます。この関数は、「グローバル・ワークフロー・プリファレンス」で指定されている LDAP ディレクトリ情報に基づいて、OID に接続します。変更があった場合は、ユーザー情報を OID から取得し、oracle.apps.wf.public.user.change イベントを呼び出します。事前定義済みのサブスクリプションをこのイベントで使用可能にすると、変更されたユーザー情報が WF\_LOCAL\_USERS 表にロードされます。WF\_LDAP.Synch\_changes() を実行すると、更新が 1 回適用されます。

ユーザー情報の更新を定期的に続行するには、WF\_LDAP.Schedule\_changes() を使用します。このプロシージャは、DBMS\_JOB ユーティリティを使用してデータベース・ジョブを送信し、指定した間隔で定期的に WF\_LDAP.Synch\_changes() を実行します。デフォルトの間隔は 10 分です。更新のチェックも、この間隔で行うことをお勧めします。

WF\_LDAP.Schedule\_changes() を実行するスクリプトを作成できます。たとえば、10 分間隔で API を実行するには、次のコマンドを使用して SQL ファイルを作成します。

```

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

```

次に、SQL\*Plus を実行し、新しいスクリプトをデータベースにロードします。

---

---

**注意：** LDAP 設定を変更する前に、別の LDAP サーバーに移行したりして、WF\_LDAP API の実行を終了する必要があります。

---

---

---

---

**注意：** OID 統合を実装した場合、ユーザーの管理は OID を使用して行ってください。OID との統合後に OID 以外のツールを使用してユーザーを管理すると、WF\_LOCAL\_USERS 表にアドホック・ユーザーが作成され、ユーザー情報の不一致や予測できない結果が発生する可能性があるためです。同様に、OID 統合を実装した場合は、WF\_DIRECTORY パッケージの CreateAdHocUser()、SetAdHocUserStatus()、SetAdHocUserExpiration() または SetAdHocUserAttr() API を使用しないでください。

---

---

Workflow ロールは、アドホック・ロールであるため、OID では管理されません。

**関連項目：**

2-13 ページ [「グローバル・ワークフロー・プリファレンスの設定」](#)

8-148 ページ [「Workflow LDAP API」](#)

14-18 ページ [「User Entry Has Changed イベント」](#)

『Oracle9i データベース管理者ガイド』の「ジョブ・キューの管理」

8-124 ページ [「Workflow ディレクトリ・サービス API」](#)

## 手順 WF-5 WF\_LANGUAGES ビューの作成

Oracle Workflow Builder のプロパティ画面のフィールド値とユーザーに送信される Workflow 通知を、Oracle のインストール先で定義された言語に変換できます。ただし、そのためには、Oracle のインストール時に定義された言語を識別する WF\_LANGUAGES というビューを作成する必要があります。Oracle Workflow では、このビューを使用して、変換前の実表で見つかった行にマップする各言語用の行が、変換可能な表内に作成されます。

WF\_LANGUAGES ビューには次の列が必要です。

- Code: 言語コード
- Display\_Name: 言語の表示名
- NLS\_Language: セッションのデフォルトの言語依存動作を指定する Oracle NLS\_LANGUAGE 初期化パラメータの値
- NLS\_Territory: セッションの地域依存の日付書式と数値書式のデフォルト値を指定する Oracle NLS\_TERRITORY 初期化パラメータの値
- NLS\_Codeset: 言語のキャラクタ・セット
- Installed\_Flag: 言語が導入済で使用可能かどうかを示すフラグ

Oracle Workflow の事前に定義された各ディレクトリ・サービスのスクリプトには、WF\_LANGUAGES ビューの例が含まれています。

**コンテキスト:** この手順を実行する必要があるのは、1 度のみです。

**参照:** 『Oracle9i Database グローバリゼーション・サポート・ガイド』を参照してください。

### ► Oracle Workflow Builder のユーザー定義オブジェクトの他言語による表示（スタンドアロン版のみ）

1. Oracle Workflow をインストールします。
2. ワークフロー・サーバーで WF\_LANGUAGES ビューを作成します。
3. wfnlenna.sql スクリプトを実行し、該当する言語を使用可能にします。16-12 ページの「[wfnlenna.sql](#)」を参照してください。
4. WFNLADD.sql スクリプトを実行し、使用可能にした言語のための行を各ワークフロー・オブジェクト変換表に作成します。16-6 ページの「[WFNLADD.sql](#)」を参照してください。
5. 環境変数 NLS\_LANG を、新しい言語に設定します。

たとえば、UNIX では次のコマンドを使用します。

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

Windows NT の場合は、regedit32 コマンドを実行し、HKEY\_LOCAL\_MACHINE¥SOFTWARE¥ORACLE 階層の下で NLS\_LANG 設定を検索しま

す。NLS\_LANG をダブルクリックして変数を新しい言語コードに設定し、編集結果を保存します。

6. ワークフロー・プロセス定義の翻訳版を作成し、それをフラット・ファイル (.wft) として保存します。
7. 現在の NLS\_LANG 設定が正しいことを確認して、翻訳後の .wft ファイルを Workflow データベースにロードします。

---

**注意：** ワークフロー定義ローダーが .wft ファイルに指定された言語を使用してロードする言語を決定し、ワークフロー・リソース・ジェネレータが言語パラメータを受け取ります。言語パラメータを指定しなければ、ワークフロー・リソース・ジェネレータはデフォルトで現在の NLS\_LANG に設定されます。

---

► **Oracle Workflow の Web セッションの他言語による表示 (スタンドアロン版のみ)**

- ワークフロー管理者として Oracle Workflow に複数の言語をインストールしている場合は、「グローバル・ワークフロー・プリファレンス」Web 画面の「言語」パラメータを設定し、ユーザーの Web セッションで表示されるデフォルトの言語を指定できます。各ユーザーは、「ユーザー設定項目」Web 画面で「言語」パラメータを設定し、このデフォルト言語を上書きできます。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」および 9-6 ページの「[ユーザー設定項目の設定](#)」を参照してください。

► **電子メール通知の他言語による表示**

1. 適切な言語サブディレクトリ \$ORACLE\_HOME/wf/res/<lang> にあるファイル wfmail.wft をチェックして、電子メール通知のテンプレートが、設定する言語に変換されているかどうかを確認します。2-65 ページの「[手順 WF-10 メッセージ・テンプレートの変更](#)」を参照してください。
2. 表示する言語用の電子メール・テンプレートが使用可能であれば、「グローバル・ワークフロー・プリファレンス」Web 画面で、ユーザーおよびロールのデフォルト言語設定をその言語に設定できます。2-12 ページの「[手順 WF-2 グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

## 手順 WF-6 ソケット・リスナー・プロファイル・オプションの設定

「通知の詳細」Web 画面には、通知メッセージのフォーム属性をサポートする添付フォーム・アイコンを表示できます。Oracle Applications ユーザーは、Oracle Applications メニューから Oracle Workflow の通知の「ワークリスト」を起動できます。

この「ワークリスト」から、ユーザーは通知リンクを選択し、「通知の詳細」ページに通知の内容を表示できます。「通知の詳細」ページに添付フォーム・アイコンが表示される場合は、そのアイコンを選択して Oracle Applications フォームを起動できます。

Oracle Workflow で「通知の詳細」ページからフォームを起動する前に、Oracle Applications で適切なコンテキスト情報をチェックする必要があります。そのためには、フォーム側のソケット・リスナーを有効化します。

Oracle Applications のソケット・リスナーを有効化するには、「システム・プロファイル値」ウィンドウで「ソケット・リスナー有効化」プロファイル・オプションを「Yes」に設定します。

また、ワークフロー管理者は、Java プラグイン用に

\$FND\_TOP/resource/<language>/wfcfg.msg 内で次のトークン値を指定する必要があります。

```
WF_CLASSID                <Jinitiator のクラス ID>
                           (Microsoft Internet Explorer を使用している場合は必須です。)
WF_PLUGIN_DOWNLOAD        <プラグインのダウンロード先>
                           (http://<server>/OA_JAVA/ など。)
WF_PLUGIN_VERSION         <プラグインのバージョン>
                           (1.1.7.27 など)
```

ワークフロー・リソース・ジェネレータを実行し、wfcfg.msg の内容を WF\_RESOURCES 表にロードします。

これらの 3 つの値は、「グローバル・ワークフロー・プリファレンス」ページで設定することもできます。2-12 ページの「[手順 WF-2 グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

また、ソケット・リスナー・プロファイル・オプションは、Oracle Workflow が添付フォームの起動に使用するポートに設定する必要があります。このプロファイル・オプションには、ユーザーごとに異なるポートを設定できます。

ソケット・リスナー・ポートがユーザー・レベルで設定されていない場合、添付フォームはその環境に設定されているデフォルトのポートで起動されます。ユーザーごとに異なるポートを設定している場合は、指定されたポートでユーザーごとにフォームが起動されます。異なるソケット・リスナー・ポートを使用することにより、同じマシン上の Oracle Applications にログインしている 2 人のユーザーが、相互に干渉することなくそれぞれの添付フォームを同時に起動できます。



**コンテキスト：**この手順を実行する必要があるのは、1 度のみです。

**参照：**『Oracle Applications System Administrator's Guide』の「Overview of Setting User Profiles」を参照してください。

**参照：**8-108 ページの「[ワークフロー・リソース・ジェネレータの実行](#)」を参照してください。

## 手順 WF-7 環境変数 WF\_RESOURCES の設定

Oracle Workflow のスタンドアロン版を使用していて、ワークフロー・サーバーを UNIX プラットフォーム上にインストールしている場合は、環境変数 WF\_RESOURCES を、言語依存の Oracle Workflow リソース・ファイル (wf<language>.res) を指すように設定する必要があります。通常、リソース・ファイルは、Oracle Workflow Server のディレクトリ構造のうち res サブディレクトリに格納されています。

---

**注意：** 環境変数の値を二重引用符 (" ") で囲まないでください。サポートされていません。

---

ワークフロー・サーバーが Windows NT プラットフォームにインストールされていれば、この環境変数を設定する必要はありません。Windows NT 上のワークフロー・サーバーでは、wf<language>.res ファイルのパスを識別する環境変数 WF\_RESOURCES が自動的に設定されます。

また、Oracle Applications embedded Workflow を使用する場合も、この環境変数の設定は不要です。Oracle Applications の場合、言語依存の Oracle Workflow リソース・ファイルのパスは \$FND\_TOP/\$APPLRSC/wf<language>.res です。

**コンテキスト：** この手順を実行する必要があるのは、1 度のみです。

## 手順 WF-8 バックグラウンドのワークフロー・エンジンの設定

ワークフロー・エンジンは、プロセスを開始して実行すると、必要なアクティビティをすべて完了してから、次の適切なアクティビティに進みます。場合によっては、アクティビティの完了までに大量の処理リソースや時間が必要になることがあります。Oracle Workflow では、このようなコストの高いアクティビティをバックグラウンド・タスクとして実行する補助エンジンを設定して、ワークフロー・エンジンの負荷を管理できます。このような場合、コストの高いアクティビティはワークフロー・エンジンによって遅延され、後でバックグラウンド・エンジンによって実行されます。その後、メインのワークフロー・エンジンは、次に使用可能なアクティビティに進むことができますが、これによりプロセスの別の並列する分岐が発生する可能性があります。

また、バックグラウンド・エンジンは、タイムアウトになった通知アクティビティを処理するように設定する必要があります。ワークフロー・エンジンは、応答を必要とする通知アクティビティに到達すると、通知システムをコールして該当する実行者に通知を送信し、実行者が通知アクティビティを完了するまで、通知アクティビティを「NOTIFIED」ステータスに設定します。一方、タイムアウトになったアクティビティを処理するように設定されたバックグラウンド・エンジンは、「NOTIFIED」のアクティビティを定期的にチェックし、指定されたタイムアウト値になっていないかどうかをチェックします。「NOTIFIED」のアクティビティにタイムアウト値があり、現在の日付と時刻がそのタイムアウト値を超えている場合、バックグラウンド・エンジンはそのアクティビティをタイムアウトとしてマークし、ワークフロー・エンジンをコールします。次に、ワークフロー・エンジンは <Timeout> トランジション・アクティビティを実行して処理を再開します。

バックグラウンド・エンジンには、停止しているプロセスへの対応を設定する必要があります。ステータスがアクティブ (ACTIVE) でも進行できなくなったプロセスは、停止していると識別されます。たとえば、次の状況ではプロセスが停止することがあります。

- プロセス内のスレッドが次のアクティビティに移行したときに、終了アクティビティとして定義されていないのに、モデル化されたアクティビティがそれ以降に存在せず、アクティブなアクティビティも存在しない場合
- 1つのスレッドで構成されるプロセスがループバックしたときに、ループのピボット・アクティビティの「再開封時」プロパティが「無効」に設定されている場合
- アクティビティが結果を返したときに、有効なトランジションが存在しない場合。たとえば、関数アクティビティの関数が予期しない結果値を返したときに、それ以降にデフォルト・トランジションがモデル化されていない場合、そのプロセスは続行できません。

バックグラウンド・エンジンは、停止しているプロセスのステータスを `ERROR:#STUCK` に設定し、定義済みのエラー・プロセスを実行します。

必要な数だけバックグラウンド・エンジンを定義して起動し、遅延アクティビティおよびタイムアウトになったアクティビティをチェックできます。

バックグラウンド・エンジンは、特定の項目タイプに関連付けられている特定のコスト範囲内のアクティビティを処理するように制限できます。また、バックグラウンド・エンジンは、実行時に適切なアクティビティを完了するまで実行されます。

通常は、バックグラウンド・エンジンを定期的に再起動するスクリプトを使用するか (Oracle Workflow のスタンドアロン版の場合)、または定期的に再発行するようにバックグラウンド・プロセス・コンカレント・プログラムのスケジュールを設定して (Oracle Applications embedded Workflow の場合)、バックグラウンド・エンジンを定期的に行うように設定する必要があります。

バックグラウンド・エンジンは、タイムアウトになったアクティビティをチェックし、遅延アクティビティを処理し、停止しているプロセスに対応するためにそれぞれ1つ以上必要です。少なくとも、タイムアウト / 遅延アクティビティ用に1つ、停止しているプロセス用に1つは設定する必要があります。

通常、停止しているプロセスをチェックするバックグラウンド・エンジンは、遅延アクティビティを処理するバックグラウンド・エンジンと別個に設定する必要があります。ただし、実行頻度は少なくともかまいません (通常は、1日に1回)。システムの負荷が低いときにバックグラウンド・エンジンを実行して、停止しているプロセスをチェックします。

**コンテキスト:** この手順を実行する必要があるのは、1度のみです。

**参照:** 4-44 ページの「[アクティビティ・コスト](#)」を参照してください。

**参照:** 5-3 ページの「[タイムアウト・トランジション](#)」を参照してください。

**参照:** C-6 ページの「[アクティビティの遅延](#)」を参照してください。

## ▶ バックグラウンド・エンジンの起動

Oracle Workflow のスタンドアロン版を使用している場合は、WF\_ENGINE.BACKGROUND() API でバックグラウンド・エンジンを起動します。スタンドアロン版には、バックグラウンド・エンジンを繰り返し実行するサンプル・スクリプトが用意されています。DBMS\_JOB パッケージのプロシージャを使用すると、バックグラウンド・エンジンをデータベース・ジョブとしてスケジュールおよび管理できます。8-44 ページの「[Background](#)」および『Oracle9i データベース管理者ガイド』の「ジョブ・キューの管理」を参照してください。

Oracle Applications embedded Workflow を使用している場合は、「Submit Requests」フォームを使用してバックグラウンド・プロセス・コンカレント・プログラムを発行し、バックグラウンド・エンジンを起動できます。2-42 ページの「[バックグラウンド・エンジンのスケジューリング](#)」を参照してください。

---

---

**注意：** Oracle Applications Manager を実装している環境で Oracle Applications embedded Workflow を使用している場合は、Oracle Workflow Manager を使用してワークフロー・バックグラウンド・プロセス・コンカレント・プログラムを発行および管理することができます。詳細は、Oracle Applications Manager のオンライン・ヘルプを参照してください。

また、Oracle9i リリース 2 で利用可能なスタンドアロン版の Oracle Workflow を使用している場合は、Oracle Enterprise Manager から利用可能なスタンドアロン Oracle Workflow Manager コンポーネントを使用して、ワークフロー・バックグラウンド・エンジン・データベース・ジョブを発行および管理できます。詳細は、Oracle Workflow Manager のオンライン・ヘルプを参照してください。

---

---

---

---

**注意：** バックグラウンド・エンジンは、タイムアウトになったアクティビティをチェックし、遅延アクティビティを処理し、停止しているプロセスに対応するためにそれぞれ 1 つ以上必要です。少なくとも、タイムアウト / 遅延アクティビティ用に 1 つ、停止しているプロセス用に 1 つは設定する必要があります。

---

---

## バックグラウンド・エンジンのスケジューリング

Oracle Applications embedded Workflow を使用している場合は、バックグラウンド・エンジンのプロシージャをコンカレント・プログラムとして発行し、様々なバックグラウンド・エンジンを異なるタイミングで実行するようにスケジューリングできます。ワークフロー・バックグラウンド・プロセスを発行するには、Oracle Applications の「Submit Requests」ウィンドウを使用します。

---

---

**注意：** ワークフロー・バックグラウンド・プロセスのロールバック・セグメントをデフォルトより大きくする場合は、システム管理者として「コンカレント・プログラム」ウィンドウを開けば、必要な大きさのロールバック・セグメントを指定できます。このロールバック・セグメントは、デフォルトのかわりに使用されます。ただし、最初に確定したときに元に戻ります。

「コンカレント・プログラム」ウィンドウからワークフロー・バックグラウンド・プロセス・コンカレント・プログラムに問い合わせ (FNDWFBG)、「セッション管理」ボタンを選択します。次に、「セッション管理」ウィンドウで、必要な大きさのロールバック・セグメントを「ロールバック・セグメント」フィールドに入力し、作業を保存します。『Oracle Applications System Administrator's Guide』の「Concurrent Programs Window」を参照してください。

---

---

► **ワークフロー・バックグラウンド・プロセスをコンカレント・プログラムとして実行する手順は、次のとおりです。**

1. 「Submit Requests」フォームにナビゲートします。
2. 「ワークフロー・バックグラウンド・プロセス」コンカレント・プログラムを要求として発行します。『Oracle Applications User's Guide』の「Submitting a Request」を参照してください。
3. 「パラメータ」ウィンドウで次のパラメータの値を入力します。

<b>項目タイプ</b>	項目タイプを指定すると、このエンジンがその項目タイプに関連付けられたアクティビティだけに制限されます。項目タイプを指定しない場合は、エンジンは項目タイプに関係なく、すべての遅延アクティビティを処理します。
<b>下限</b>	バックグラウンド・エンジンでアクティビティを実行するために必要な最小コストを 100 分の 1 秒単位で指定します。
<b>上限</b>	バックグラウンド・エンジンでアクティビティを実行するために使用可能な最大コストを、100 分の 1 秒単位で指定します。  「下限」および「上限」を指定すると、複数のバックグラウンド・エンジンが作成され、処理するアクティビティの範囲をかなり限定することができます。これらの引数のデフォルト値は、それぞれ 0 と 100 です。つまり、バックグラウンド・エンジンは、アクティビティを効率的に実行することができます。
<b>処理遅延</b>	バックグラウンド・エンジンで、遅延アクティビティをチェックするかどうかを指定します。このパラメータを「Yes」に設定すると、遅延アクティビティをエンジンでチェックできます。

- |                 |  |
|-----------------|--|
| <b>処理タイムアウト</b> | バックグラウンド・エンジンで、タイムアウトになったアクティビティをチェックするかどうかを指定します。このパラメータを「Yes」に設定すると、タイムアウトになったアクティビティをエンジンでチェックできます。 |
| <b>処理スタック</b>   | バックグラウンド・エンジンが停止しているプロセスをチェックするかどうかを指定します。このパラメータを「Yes」に設定すると、停止しているプロセスがチェックされます。                     |

---

---

**注意：** バックグラウンド・エンジンは、タイムアウトになったアクティビティをチェックし、遅延アクティビティを処理し、停止しているプロセスに対応するためにそれぞれ 1 つ以上必要です。少なくとも、タイムアウト / 遅延アクティビティ用に 1 つ、停止しているプロセス用に 1 つは設定する必要があります。

---

---

4. 「OK」を選択して「パラメータ」ウィンドウを閉じます。
5. 実行オプションを変更してバックグラウンド・エンジンの予定を定義した後に、「発行」を選択して要求を発行します。

**関連項目：**

『Oracle Applications System Administrator's Guide』の「Concurrent Programs and Requests」

➤ **エンジンのしきい値の設定**

バックグラウンド・エンジンのしきい値を設定するには、エンジンの起動時に「下限」引数と「上限」引数を指定します。その後は、コストが指定した範囲に含まれるアクティビティのみがバックグラウンド・エンジンで処理されます。

ワークフロー・エンジンのデフォルトのしきい値は、50 です。これより高いコストのアクティビティは、バックグラウンド・エンジンにまわされ、処理が延期されます。

場合によっては、アクティビティのコストが 50 未満でも、エンジンでアクティビティを遅延させる方がよいことがあります。そのためには、関数アクティビティ用の PL/SQL ストアド・プロシージャでワークフロー・エンジンのしきい値を変更します。

ワークフロー・エンジンのしきい値は、THRESHOLD という外部定数で設定されます。このしきい値を別の値に設定するには、PL/SQL プロシージャに次の行を挿入します。

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

他のアクティビティが期待通りに処理されるように、SQL\*Plus の後か次の関数アクティビティ内でしきい値をリセットする必要があります。

## 手順 WF-9 通知メーラーの導入

通知メーラーは、電子メールの送信や Oracle Workflow 通知システムへの応答処理を実行するプログラムです。この手順を実行する必要があるのは、ワークフロー・ユーザーが、「通知ワークリスト」Web 画面のみでなく電子メール経由でも通知を受け取るようにする場合のみです。通知メーラーにより、データベース内で送信する必要のあるメッセージが検索され、これらのメッセージが SMTP アドバンスト・キューからデキューされ、メッセージごとに次の処理が実行されます。

- 受信者のロールを分析して、1 つの電子メール・アドレスを判別します。ロール自体がメール・リストになる場合もあります。
- ディレクトリ・サービスの定義に従って、データベース・セッションをロールに設定されている言語と地域に切り替えます。
- 該当するメッセージ・テンプレートを使用し、メッセージやオプションの添付ファイルを生成します。
- UNIX の Sendmail または Windows NT の MAPI 準拠のメール・アプリケーション経由でメッセージを送信します。

---

**注意：** WF SMTP\_O\_1\_QUEUE という標準エージェントが、イベント・マネージャの通知メーラー SMTP キューに定義されています。このエージェントは、イベント・マネージャの「セットアップのチェック」画面と「イベント・システム・ローカル・キュー」画面に表示されます。これらの画面を使用して、通知メーラーのキューにある通知メッセージの数をチェックできます。ただし、WF SMTP\_O\_1\_QUEUE エージェントは、ビジネス・イベント・システムでは使用されません。13-27 ページの「標準エージェント」、13-56 ページの「ビジネス・イベント・システムのセットアップのチェック」および 13-79 ページの「ローカル・キューの確認」を参照してください。

---

また、通知メーラーは、応答メール・アカウントにメールされたメッセージのテキストを解析して応答を処理し、該当する通知応答関数をコールして通知を完了します。

電子メール通知は、Oracle Workflow Builder で定義されている標準テンプレートに基づいています。テンプレートでは、返信に必要な構文が記述され、通知の確認に必要な情報が表示されます。生成された電子メール・メッセージでは、カスタム・サイト情報、締切日、応答処理に必要なすべての情報なども指定されます。2-65 ページの「[手順 WF-10 メッセージ・テンプレートの変更](#)」を参照してください。

通知メーラーを実行するように設定すると、送信するメッセージがあるかどうかデータベースで定期的に検索され、処理する応答があるかどうか応答メール・アカウントでチェックされます。シャットダウンして別のパラメータを指定して再起動する必要がない限り、他の作業は不要です。

---

---

**注意：** データベースに障害が発生したり、セッションに対するパッケージ定義が削除または置換されたために PL/SQL パッケージのステータスが無効な場合は、通知メーラー自体がシャットダウンされます。Oracle Workflow のスタンドアロン版を使用している場合に、障害が継続するときは、通知メーラーを手動で再起動するか、通知メーラーを再起動するシェルスクリプトを実行します。2-64 ページの「[通知メーラーの常駐シェルスクリプトの実行](#)」を参照してください。Oracle Applications embedded Workflow を使用している場合は、コンカレント・マネージャを使用して通知メーラー・プログラムを手動で再起動するか、定期的に再起動するようにスケジュールを設定します。

---

---

**コンテキスト：** この手順を実行する必要があるのは、1 度のみです。

**参照：** 10-2 ページの「[電子メールによる通知の閲覧](#)」を参照してください。

## MIME のフル・サポート

Oracle Workflow では、MIME (Multi-purpose Internet Mail Extensions) でコード化されたメッセージを完全にサポートします。そのため、ワークフローのユーザーは、キャラクタ・セットの異なる言語やマルチメディアでコード化されたコンテンツが含まれるメッセージを通知メーラーで変換できます。

## 通知環境設定

Oracle Workflow では、「ユーザー設定項目」Web 画面の通知環境設定によって、通知の表示方法を決定できます。9-6 ページの「[ユーザー設定項目の設定](#)」を参照してください。

通常、ユーザーの通知環境設定は、そのユーザーが使用するメール・リーダーの機能によって決定されます。メール・リーダーには、プレーン・テキストのみを表示できるもの、HTML 形式を表示できるもの、さらには添付ファイル内の HTML 形式のみを表示できるものがあります。使用できる通知環境設定は次の 5 つです。

- プレーン・テキスト・メール (MAILTEXT)： 通知メッセージはプレーン・テキストとして表示され、添付ファイルはありません。2-47 ページの「[プレーン・テキスト電子メール](#)」を参照してください。
- HTML メール (MAILHTML)： 通知メッセージは HTML 形式のテキストとして表示され、1 つ以上の添付ファイルがあります。これは「通知」Web 画面にある通知へのリンクです。通知メッセージに「内容の添付」のメッセージ属性がある場合、これらの属性もメッセージへの添付ファイルとして追加表示されます。2-48 ページの「[HTML 形式の電子メール](#)」を参照してください。



---

**注意：** 使用中のメール・リーダーではメール・メッセージ本文中の HTML 形式を解釈できない場合に、HTML 形式の通知を表示するには、通知環境設定を MAILATTH (HTML 添付ファイル付きのプレーン・テキスト・メール) に変更してください。MAILATTH の環境設定では、通知の HTML 形式版が、プレーン・テキスト通知への添付ファイルとして配信されます。

---

- HTML 添付ファイル付きのプレーン・テキスト・メール (MAILATTH) : 通知メッセージはプレーン・テキストとして表示され、少なくとも 2 つの添付ファイルがあります。1 つはメッセージの HTML 形式版、もう 1 つは「通知」Web 画面にある通知へのリンクです。通知メッセージに「内容の添付」のメッセージ属性がある場合、これらの属性もメッセージへの添付ファイルとして追加表示されます。2-50 ページの「[HTML 添付ファイル付きのプレーン・テキスト電子メール](#)」を参照してください。
- プレーン・テキスト要約メール (SUMMARY) : メッセージは、すべてのオープン通知のプレーン・テキスト要約です。要約にある個々の通知に応答するには、「通知」Web 画面から通知にアクセスする必要があります。
- メールを送信しないでください (QUERY) : 通知メーラーでは電子メール通知が送信されません。かわりに、「通知」Web 画面から、自分の通知を問い合わせて応答する必要があります。

---

**注意：** 通知環境設定がメールを送信するように設定されていても、「通知」Web 画面から、いつでも自分の通知を問い合わせて応答できます。

---

**参照：** 10-2 ページの「[電子メールによる通知の閲覧](#)」を参照してください。

**参照：** 10-13 ページの「[Web ブラウザによる通知の表示](#)」を参照してください。

**参照：** 10-25 ページの「[電子メールによる通知要約の確認](#)」を参照してください。

## プレーン・テキスト電子メール

通知の実行者が通知環境設定を MAILTEXT (プレーン・テキスト・メール) に設定している場合、通知にはワークフローの通知表でそのようにフラグが設定されます。通知メーラーが通知表を検索してそのフラグを識別すると、表の情報からプレーン・テキスト電子メールが生成され、実行者ロールに送信されます。通知メーラーがプレーン・テキスト電子メールを生成するときには、Oracle Workflow Builder のメッセージ・プロパティ画面で定義したテキスト本文を使用します。そのトークンは、メッセージ本文中で参照されているすべての属性値をプレーン・テキスト値に置き換えます。たとえば、次のようになります。

- PL/SQL および PL/SQL CLOB 文書属性は、プレーン・テキスト版の PL/SQL 文書に置換されるトークンです。

- URL 属性は、後にコロン (:) と URL を伴う URL 属性の表示名に置換されるトークンです。

<URL\_Attribute\_Display\_Name>:<URL>

---

---

**注意：**「属性」プロパティ画面で「内容の添付」がオンになっているメッセージ属性は、親通知にプレーン・テキストとして添付されます。添付ファイルに特殊な書式が含まれていると、この処理によって添付ファイルの一部が解読不能になったり、プレーン・テキストの電子メール・ソフトウェアで添付ファイルが認識されない場合があるため注意してください。このような添付ファイルを表示するには、「通知ワークリスト」Web 画面で通知を表示する必要があります。10-13 ページの「[Web ブラウザによる通知の表示](#)」を参照してください。

---

---

プレーン・テキスト電子メールの受信者が応答するときは、手動で通知に返信し、通知で提供された指示に続けて応答値を入力します。10-4 ページの「[テンプレートによる応答を使用したプレーン・テキストの電子メール通知への応答](#)」および 10-6 ページの「[直接応答を使用したプレーン・テキストの電子メール通知への応答](#)」を参照してください。

## HTML 形式の電子メール

通知の実行者が通知環境設定を MAILHTML (HTML メール) に設定している場合、通知にはワークフローの通知表でフラグが設定されます。通知メーラーが通知表を検索してそのフラグを識別すると、表内の情報から HTML 形式の電子メール通知が生成され、実行者ロールに送信されます。実行者ロールは、メッセージ本文に含まれる HTML コンテンツを解釈および表示できる電子メール・ソフトウェアを使用する必要があります。

---

---

**注意：** 使用中の電子メール・ソフトウェアでメッセージ本文中の HTML 形式を解釈できない場合は、通知環境設定を MAILATTH (HTML 添付ファイル付きのプレーン・テキスト・メール) に設定してください。

---

---

通知メーラーでは、「メッセージ本文」プロパティ画面のメッセージに定義された HTML 本文を使用して、HTML 電子メールが生成されます。HTML 本文が定義されていなければ、テキスト本文を使用して HTML 形式のメールが生成されます。通知メーラー・トークンは、メッセージ本文で参照されているメッセージ属性を HTML 形式値に置換します。たとえば、次のようになります。

- PL/SQL および PL/SQL CLOB 文書属性は、HTML タグ <pre>...</pre> で囲まれた HTML テキストまたはプレーン・テキストを置換するトークンです。
- URL 属性は、HTML アンカーに置換されるトークンです。このアンカーを選択すると、電子メール・ソフトウェアからターゲットの URL ページにリンクします。

---

---

**注意：**「属性」プロパティ画面で「内容の添付」がオンになっているメッセージ属性は、親メッセージに HTML 形式の添付ファイルとして添付されます。たとえば、次のようになります。

- メッセージ属性が URL 属性の場合は、「通知参照」という添付ファイルがメッセージに追加されます。この添付ファイルには、「内容の添付」がオンになっているメッセージの各 URL 属性へのリンクが含まれています。URL へのリンクを選択すると、その URL にナビゲートできます。通知メーラーでは、画像、ビデオまたは音声の URL コンテンツは特殊処理されません。
- メッセージ属性が PL/SQL または PL/SQL CLOB 文書属性である場合、正常に生成された PL/SQL 文書がフェッチされ、メッセージに添付されます。

---

---

HTML 形式の電子メール通知には、常に 1 つは添付文書が含まれています。添付ファイルは「通知の詳細リンク」と呼ばれます。この添付ファイルを選択すると、電子メール・ソフトウェアによってブラウザのウィンドウが開かれ、「通知の詳細」Web 画面に通知が表示されます。この Web 画面からは、通知メーラーを介して応答を処理する必要がなく、通知に直接応答できます。

---

---

**注意：** 通知メーラーの構成ファイルで SEND\_ACCESS\_KEY パラメータが N に設定されており、まだログインしていない場合は、「通知の詳細」リンクを選択すると、「通知の詳細」Web 画面にアクセスする前にログインを求めるプロンプトが表示されます。

---

---

---

**注意：**「通知の詳細リンク」添付ファイルのファイル名は、WF\_URL\_NOTIFICATION リソース・トークンのテキスト値、または、テキスト値が定義されていない場合はトークン名によって決定されます。同様に、「通知参照」添付ファイルのファイル名は、WF\_URLLIST\_ATTACHMENT リソース・トークンのテキスト値、またはトークン名（テキスト値が定義されていない場合）によって決定されます。デフォルトのファイル名は、それぞれ Notification Detail Link.html および Notification References.html です。これらの添付ファイルに異なるファイル名を指定する場合は、最初に WF\_URL\_NOTIFICATION および WF\_URLLIST\_ATTACHMENT リソース・トークンのテキスト値として新規ファイル名を指定して、.msg ソース・ファイルを作成する必要があります。次に、ワークフロー・リソース・ジェネレータのプログラムを使用して、ソース・ファイルから Oracle Workflow の新規リソース・ファイル (wf<language>.res) を生成し、ソース・ファイルからデータベース表 WF\_RESOURCES に新規シード・データをアップロードします。8-108 ページの「ワークフロー・リソース・ジェネレータの実行」および 2-40 ページの「手順 WF-7 環境変数 WF\_RESOURCES の設定」を参照してください。

---

または、HTML メッセージ本文中で応答を表すリンクをクリックして、HTML 形式の通知に応答することもできます。応答リンクによって、選択した事前定義済の応答値で変更された応答テンプレートを含むプレーン・テキスト電子メールの応答が生成されます。10-9 ページの「HTML 形式の電子メール通知への応答」を参照してください。

## HTML 添付ファイル付きのプレーン・テキスト電子メール

通知の実行者が通知環境設定を MAILATTH（HTML 添付ファイル付きのプレーン・テキスト・メール）に設定している場合、通知にはワークフローの通知表でそのようにフラグが設定されます。通知メーラーが通知表を検索してそのフラグを識別すると、表内の情報から HTML 添付ファイル付きのプレーン・テキスト電子メール通知が生成され、実行者ロールに送信されます。実行者ロールは、HTML 添付ファイルをサポートする電子メール・ソフトウェアを使用する必要があります。

通知メーラーは、「メッセージ本文」プロパティ画面でメッセージとして定義されたテキスト本文を使用して、電子メールのプレーン・テキスト本文を生成します。通知メッセージの HTML 版も生成され、プレーン・テキスト電子メールへの添付ファイルとして送信されます。この添付ファイルは、「HTML メッセージ本文」と呼ばれます。通知メーラーは、そのメッセージに定義された HTML 本文から HTML 添付ファイルの内容を生成します。HTML 本文が定義されていなければ、テキスト本文を使用して HTML 形式のメールが生成されます。通知メーラー・トークンは、プレーン・テキスト本文で参照されているすべてのメッセージ属性をプレーン・テキスト値に置換し、添付の HTML メッセージで参照されているすべてのメッセージ属性を HTML 形式値に置換します。2-47 ページの「プレーン・テキスト電子メール」および 2-48 ページの「HTML 形式の電子メール」を参照してください。

電子メール・ソフトウェアがメッセージ本文中の **HTML** 形式をサポートしている場合は、メッセージ本文中の行に **HTML** 添付ファイルも表示されます。

---

**注意：**「属性」プロパティ画面で「内容の添付」がオンになっているメッセージ属性は、**HTML** 形式の添付ファイルとして添付されます。たとえば、次のようになります。

- メッセージ属性が **URL** 属性の場合は、「通知参照」という添付ファイルがメッセージに追加されます。この添付ファイルには、「内容の添付」がオンになっているメッセージの各 **URL** 属性へのリンクが含まれています。**URL** へのリンクを選択すると、その **URL** にナビゲートできます。
  - メッセージ属性が **PL/SQL** または **PL/SQL CLOB** 文書属性である場合、正常に生成された **PL/SQL** 文書がフェッチされ、メッセージに添付されます。
- 

通知環境設定を **MAILATTH** (**HTML** 添付ファイル付きのプレーン・テキスト) に設定しているユーザーが通知を受け取ると、常に 2 つ以上の添付ファイルが含まれます。1 つの添付ファイルは「**HTML** メッセージ本文」、もう 1 つの添付ファイルは「通知の詳細リンク」です。「通知の詳細リンク」を選択すると、電子メール・ソフトウェアによってブラウザのウィンドウが開かれ、「通知の詳細」**Web** 画面に通知が表示されます。この **Web** 画面からは、通知メーラーを介して応答を処理する必要がなく、通知に直接応答できます。10-12 ページの「**HTML** 添付ファイル付きのプレーン・テキスト電子メール通知への応答」を参照してください。

---

**注意：** 通知メーラーの構成ファイルで **SEND\_ACCESS\_KEY** パラメータが **N** に設定されており、まだログインしていない場合は、「通知の詳細」リンクを選択すると、「通知の詳細」**Web** 画面にアクセスする前にログインを求めるプロンプトが表示されます。

---

または、このタイプの通知の受信者は次のいずれかの方法で応答することもできます。

- 通知に手動で返信し、通知で提供された指示に従って応答値を入力します。10-4 ページの「**テンプレートによる応答を使用したプレーン・テキストの電子メール通知への応答**」および 10-6 ページの「**直接応答を使用したプレーン・テキストの電子メール通知への応答**」を参照してください。
- 「**HTML** メッセージ本文」添付ファイルを選択して **HTML** 形式版の電子メール・メッセージを表示し、応答を表す **HTML** リンクをクリックします。応答リンクによって、選択した事前定義済の応答値で更新された応答テンプレートを含むプレーン・テキスト電子メールの応答が生成されます。

---

---

**注意：**「HTML メッセージ本文」添付ファイルのファイル名は、WF\_HTML\_MESSAGE リソース・トークンのテキスト値、またはテキスト値が定義されていない場合はトークン名によって決定されます。同様に、「通知の詳細リンク」添付ファイルのファイル名は、WF\_URL\_NOTIFICATION リソース・トークンのテキスト値、またはトークン名（テキスト値が定義されていない場合）によって決定されます。「通知参照」添付ファイルのファイル名は、WF\_URLLIST\_ATTACHMENT リソース・トークンのテキスト値、またはトークン名（テキスト値が定義されていない場合）によって決定されます。デフォルトのファイル名は、それぞれ HTML Message Body.html、Notification Detail Link.html および Notification References.html です。これらの添付ファイルに異なるファイル名を指定する場合は、最初に WF\_HTML\_MESSAGE、WF\_URL\_NOTIFICATION および WF\_URLLIST\_ATTACHMENT リソース・トークンのテキスト値として新規ファイル名を指定して、.msg ソース・ファイルを作成する必要があります。次に、ワークフロー・リソース・ジェネレータのプログラムを使用して、ソース・ファイルから Oracle Workflow の新規リソース・ファイル（wf<language>.res）を生成し、ソース・ファイルからデータベース表 WF\_RESOURCES に新規シード・データをアップロードします。8-108 ページの「ワークフロー・リソース・ジェネレータの実行」および 2-40 ページの「手順 WF-7 環境変数 WF\_RESOURCES の設定」を参照してください。

---

---

## 通知メーラーの起動

通知メーラーをインストールし、UNIX の Sendmail または Windows NT の MAPI 準拠のメール・アプリケーションで動作するように設定できます。ただしその前に、これらのメール・アプリケーションのうちの 1 つに、通知メーラー専用のメール・アカウントを 1 つ以上設定する必要があります。また、応答処理に使用するために、メール・アカウント内で 3 つのフォルダまたはファイルを定義する必要があります。

ユーザーは、Windows NT で動作する MAPI 準拠の電子メール・ソフトウェア、または UNIX Sendmail でゲートウェイを提供できる電子メール・ソフトウェアを使用して、電子メール通知を受け取ります。

---

---

**注意：** また、Oracle9i リリース 2 で利用可能なスタンドアロン版の Oracle Workflow を使用している場合は、Oracle Enterprise Manager から利用可能なスタンドアロン Oracle Workflow Manager コンポーネントを使用して、通知メーラーのスループットを監視できます。詳細は、Oracle Workflow Manager のオンライン・ヘルプを参照してください。

---

---

## ▶ UNIX Sendmail の通知メーラーの起動

### Oracle Workflow スタンドアロン版の場合

1. 通知メーラーは、サーバー上の \$ORACLE\_HOME/bin サブディレクトリにあります。通知メーラーを実行するには、次のコマンド構文を使用します。

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

UNIX Sendmail バージョンの通知メーラーを使用するときは、<xxx> を `snd` に置き換えます。<config\_file> は、通知メーラーの実行時に指定するパラメータを含む構成ファイルのフルパス名に置き換えます。

2. または、次のコマンドを入力して、通知メーラーのパラメータを構成ファイルではなくコマンドラインで引数として指定できます。

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

構成ファイルを指定している場合でも、パラメータをコマンドラインの値で指定して、構成ファイル内で特定のパラメータ値を上書きできます。

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

<arg1> <arg2> ... は、parameter=value 形式を使用して、必要な数のオプション・パラメータと値に置き換えます。

### Oracle Applications embedded Workflow の場合

1. 通知メーラー・プログラムは、コンカレント・プログラムとして登録されています。通知メーラーのコンカレント・プログラムは、「Submit Requests」フォームまたはコマンドラインから実行できます。

---

---

**注意：** Oracle Applications Manager を実装している環境で Oracle Applications embedded Workflow を使用している場合は、Oracle Workflow Manager を使用して通知メーラーを設定および実行することができます。詳細は、Oracle Applications Manager のオンライン・ヘルプを参照してください。

---

---

2. このコンカレント・プログラムを「Submit Requests」フォームから実行するには、「Submit Requests」フォームにナビゲートします。

---

---

**注意：** システム管理者は、このプログラムを実行する職責の要求セキュリティ・グループに、このコンカレント・プログラムを追加する必要があります。『Oracle Applications System Administrator's Guide』の「Overview of Concurrent Programs and Requests」を参照してください。

---

---

3. 通知メーラー・コンカレント・プログラムを要求として発行します。『Oracle Applications User's Guide』の「Submitting a Request」を参照してください。
4. 「パラメータ」ウィンドウで、構成ファイルのパスとファイル名を入力します。構成ファイルには、通知メーラーの実行時に指定するパラメータが含まれています。
5. 「OK」を選択して「パラメータ」ウィンドウを閉じます。
6. この要求の印刷オプションと実行オプションを変更してから、「発行」を選択して要求を発行します。
7. 「Submit Requests」フォームを使用しない場合は、コマンドラインから通知メーラー・コンカレント・プログラムを実行できます。次のように入力します。

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

apps/pwd を APPS スキーマのユーザー名とパスワードに置き換え、  
config\_file を通知メーラーの実行時に指定するパラメータを含む構成ファイル  
のファイル指定に置き換えます。

config\_file のファイル指定は、次のように指定できます。

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

または

```
<native path>
```

## ► MAPI 準拠メール・アプリケーションの通知メーラーの起動

1. Oracle Universal Installer を使用して、Windows NT の PC に MAPI 準拠メール・アプリケーション用の通知メーラーをインストールします。通知メーラー・プログラムは、<drive>:\¥<ORACLE\_HOME>\¥bin にあります。
2. MS-DOS プロンプト・ウィンドウに次のコマンドを入力し、通知メーラー・プログラムを起動します。

```
<drive>:\¥<ORACLE_HOME>\¥bin\wfmlr.exe -f <config_file>
```

<config\_file> は、通知メーラーの実行時に指定するパラメータを含む構成ファイル  
のフルパス名に置き換えます。



---

**注意：** このプログラムを起動するには、「Oracle - <SID NAME>」プログラム・グループにある Oracle Workflow Notification Mailer のアイコンをダブルクリックする方法もありますが、最初にこのアイコンのプロパティを編集し、ターゲットとして前述のコマンドを挿入する必要があります。

---

3. または、通知メーラーのパラメータを構成ファイルではなくコマンドラインで引数として指定する場合は、次のコマンドを入力します。

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

構成ファイルを指定している場合でも、パラメータをコマンドラインの値で指定して、構成ファイル内で特定のパラメータ値を上書きできます。

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

<arg1> <arg2> ... は、parameter=value 形式を使用して、必要なオプション・パラメータと値に置き換えます。

## ► 通知メーラーの構成ファイルの作成

1. Oracle Workflow には、構成ファイルのサンプルとして wfmail.cfg が用意されています。Oracle Workflow のスタンドアロン版を使用している場合、このファイルは Oracle Workflow Server のディレクトリ構造のうちサブディレクトリ res にあります。Oracle Applications embedded Workflow の場合は、サーバー上の \$FND\_TOP の resource サブディレクトリにあります。また、このファイルは PC 上の <drive>:\¥<ORACLE\_HOME>\¥wf¥data サブディレクトリにもあります。

---

**注意：** Oracle Applications Manager を実装している環境で Oracle Applications embedded Workflow を使用している場合は、Oracle Workflow Manager を使用して通知メーラーを設定および実行することができます。詳細は、Oracle Applications Manager のオンライン・ヘルプを参照してください。

---

2. 構成ファイルの内容は、次の形式になっています。

```
#Description
```

```
PARAMETER1=<value1>
```

```
#Description
```

```
PARAMETER2=<value2>
```

```
...
```

#で始まる行は解析されないため、コメントとして使用できます。各パラメータ名を等号(=)の左側に、各パラメータの値を右側に指定します。

3. パラメータは、次のとおりです。

- **CONNECT:** (必須) Oracle Workflow Server がインストールされているデータベース・アカウントに接続するための情報で、  
`username/password@connect_string` (または *alias*) 形式を使用します。

---

**注意:** 通知メーラーをコンカレント・プログラムとして Oracle Applications embedded Workflow から発行する場合、CONNECT パラメータは必要ありません。この場合、コンカレント・マネージャは、通知メーラーによって使用されるデータベース接続情報を渡します。

---

- **ACCOUNT:** (必須) プログラムが通知メッセージの送信に使用するメール・アカウントに接続するための情報です。MAPI 準拠メール・プログラムの場合、アカウント情報はメール・アカウントのプロファイル名とパスワードです。Sendmail の場合、アカウント情報は着信メッセージが格納されるメール・スプール・ファイルのフルパスで、`/var/mail/applmgr3` のようになります。これは、通知メーラー起動元のアカウント (この例では、`applmgr3`) に対応していることに注意してください。

---

**注意:** 通知メーラーの Sendmail バージョンを起動する場合、環境変数 `PATH` に、Sendmail 実行ファイルのディレクトリのフルパスも指定する必要があります。

---

---

**注意:** Oracle Applications embedded Workflow を使用していて、通知メーラー・コンカレント・プログラムの Sendmail 版を起動する場合は、Account パラメータをコンカレント・マネージャの起動元アカウントに設定する必要があります。

---

- **NODE:** (必須) ノードの識別子名。ノード名は送信通知 ID に含まれています。デフォルト名は `wf` です。

複数のワークフロー・データベースを 1 つのサーバー上に配置している場合は、通知メーラー構成ファイルごとに一意のノード名を定義して、各インスタンスの通知メーラーで異なる TMP ファイルが使用されるようにする必要があります。たとえば、通知メーラーをノード名 `wf` でサーバーに配置している場合、次に設定する通知メーラーのインスタンスにはノード名 `wf2` を割り当てます。

---

**注意：** 通知メーラーはインスタンスごとに1つしか実行できません。このため、各通知メーラーでは、個別の電子メール・アカウントを使用して応答を処理する必要があります。

---

- **FROM:** 通知メッセージがユーザーに配信されるときに、メッセージ・ヘッダーの「送信元」フィールドに表示される値。デフォルトはOracle Workflow です。

---

**注意：** FROM パラメータは、UNIX Sendmail バージョンの通知メーラーが「送信元」フィールドの値を設定する場合にのみ使用します。MAPI に準拠した通知メーラーでは、ACCOUNT パラメータで指定したメール・アカウントの表示値が使用されます。

---

- **SUMMARYONLY:** (必須) この通知メーラーでは、通知環境設定が SUMMARY に設定されているユーザー / ロールに割り当てられる通知のみを処理するのか、あるいは MAILTEXT、MAILATTH または MAILHTML に指定されたユーザー / ロールの通知のみを処理するのかを指定します。有効な値は Y または N で、デフォルト値は N です。通知環境設定を MAILTEXT、MAILATTH、MAILHTML または SUMMARY に指定したワークフロー・ユーザーまたはロールがある場合は、少なくとも2つの通知メーラーを設定し、一方では SUMMARYONLY=Y、他方では SUMMARYONLY=N を指定する必要があります。ディレクトリ・リポジトリからのユーザーとロールの設定は、2-20 ページを参照してください。

SUMMARYONLY=Y に設定すると、データベースを検索して該当する通知要約を配信した後に、通知メーラー自体がシャットダウンされます。したがって、通知要約を配信する頻度で通知メーラーが実行されるように、スケジュールを設定する必要があります。要約には処理中のすべての通知が含まれるため、通知メーラーによる要約を1日に一度は実行してください。スタンドアロン環境で動作する Oracle Workflow の場合は、UNIX の cron ジョブなど、オペレーティング・システム・スクリプトを作成して、通知メーラーのスケジュールを設定する必要があります。

Oracle Applications embedded Workflow のバージョンの場合は、「Submit Requests」フォームで通知メーラー・コンカレント・プログラムのスケジュールを指定するのみですみます。

- **DIRECT\_RESPONSE:** DIRECT\_RESPONSE: 通知環境設定で MAILTEXT または MAILATTH が指定されているユーザーに、テンプレートによる応答を必要とするプレーン・テキスト通知を送信するには、N または n を指定します。Y または y を指定すると、これらの通知環境設定が指定されているユーザーに、直接応答を要求するプレーン・テキスト通知が送信されます。デフォルトは N です。

テンプレートによる応答方式の場合、ユーザーは応答プロンプトのテンプレートを使用して返信し、各プロンプトの後に応答値を引用符で囲んで入力する必要があります。直接応答方式の場合、ユーザーは応答値を返信の1行目として直接入力する

必要があります。10-2 ページの「[電子メールによる通知の閲覧](#)」を参照してください。

---

**注意：** 作成した構成ファイルに `HTML_MAIL_TEMPLATE` パラメータが含まれている場合、通知メーラーは通知環境設定が `MAILATTH` のユーザーにメッセージを送信するときに、`DIRECT_RESPONSE` パラメータを無視します。かわりに、通知環境設定が `MAILATTH` のユーザーは、「Workflow Open Mail for Outlook Express」テンプレートに定義されたプレーン・テキスト・メッセージを受信します。応答は、テンプレートに基づいて行われます。「Workflow Open Mail for Outlook Express」テンプレートを使用する必要がない場合は、`HTML_MAIL_TEMPLATE` パラメータをコメントにして、`DIRECT_RESPONSE` パラメータを有効にする必要があります。

---

- `AUTOCLOSE_FYI`： この通知メーラーで、FYI（参考）通知のように応答不要な通知を、電子メールで送信後に自動的に閉じるかどうかを指定します。有効な値は Y または N で、デフォルトは Y です。

値 N を指定すると、すべての FYI 通知はユーザーが手動で閉じるまで、通知ワークリスト内で開いたままになります。

- `ALLOW_FORWARDED_RESPONSE`： 他のロールから転送された電子メール通知に対する応答を許可するかどうかを指定します。有効な値は Y または N で、デフォルトは Y です。

値が N の場合、応答通知の「送信元」欄のメール・アドレスが、記録されている宛先ロール（またはそのロール内のユーザー）のメール・アドレスと一致するかどうかチェックされます。2 つのメール・アドレスが一致する場合、その通知は有効なルーティング規則に従って転送された、または転送されなかったことを意味し、通知メーラーでは有効な応答として扱われます。2 つのメール・アドレスが一致しない場合、その通知は単に電子メール転送コマンドを使用して転送されたことを意味し、通知メーラーではこの応答が処理されず、応答なしメールとして処理されます。

値が Y の場合、応答通知の「送信元」欄のメール・アドレスはチェックされず、応答は必ず処理されます。

---

**警告：** ALLOW\_FORWARDED\_RESPONSE を N に設定する場合は、制限事項があるため注意してください。たとえば、Oracle Workflow ディレクトリ・サービスに USER/ROLE の関係を持っていない配布リストのメール・エイリアスに通知が送信されたとします。配布リスト内のユーザーがこの通知に応答した場合、「送信元」欄のメール・アドレスは各ユーザーのメール・アドレスであり、配布リストのメール・エイリアスとは一致しないため、通知メーラーでは常に応答なしメールとして扱われます。

---

- IDLE: 送信するメッセージをチェックするまでの待機時間 (秒)。この値には 0 以上の整数を指定する必要があります。デフォルトは 30 です。
- LOG: アクティビティを記録するログ・ファイルの名前。有効な値はファイル名です。このパラメータを指定できるのは、スタンドアロン版の通知メーラーのみです。コンカレント・プログラム版の通知メーラーの場合、アクティビティの出力はコンカレント・マネージャのログ・ファイルに送られます。
- SHUTDOWN: 通知メーラーにシャットダウンを指示するファイルの名前。これにより、プロセスを中断せずに通知メーラーを安全にシャットダウンできます。通知メーラーは、処理対象の通知を探す前に、常に現在の作業用ディレクトリにシャットダウン・ファイルがあるかどうか探します。ファイルがあれば、通知メーラーはシャットダウンします。通知メーラーを再起動するには、シャットダウン・ファイルを削除する必要があります。デフォルトのファイル名は、shutdown です。

Oracle Workflow のスタンドアロン版の場合、通知メーラーの現行の作業用ディレクトリは、通知メーラーを起動するディレクトリです。Oracle Applications embedded Workflow の場合、現行の作業用ディレクトリは \$APPLCSF/\$APPLLOG ディレクトリです。環境変数 \$APPLCSF を設定していない場合は、シャットダウン・ファイルを \$FND\_TOP/\$APPLLOG ディレクトリに格納してください。

- FAILCOMMAND: 通知メーラーに重大なエラーが発生した場合に実行するコマンド。
- DEBUG: デバッグ情報をログに出力するかどうかを指定します。有効な値は Y または N で、デフォルトは N です。
- TEST\_ADDRESS: すべての送信電子メール通知をダイレクトするテスト用電子メール・アドレスを指定します。テスト・アドレスによって各宛先の電子メール・アドレスが上書きされるため、各宛先の電子メール・アドレスを変更せずにテスト通知にアクセスし、ワークフロー・プロセスをテストできます。
- REPLYTO: (必須) 応答を処理する電子メール・アカウントが、発信通知を送信する電子メール・アカウントとは異なる場合に、デフォルトの返信先となる電子メール・アドレス。

---

---

**注意：** REPLYTO パラメータは、UNIX Sendmail バージョンの通知メーラーが返信先アドレスを設定する場合にのみ使用します。MAPI に準拠した通知メーラーでは、ACCOUNT パラメータで指定したメール・アカウントの表示値が使用されます。

---

---

- **HTMLAGENT:** HTML 通知応答を処理する HTML Web Agent を識別するベース URL。HTML 添付ファイル付きの電子メール通知をサポートするには、この URL が必須です。デフォルトの URL は、「グローバル・ワークフロー・プリファレンス」Web 画面で指定した Workflow Web Agent から取り出されますが、このパラメータに異なる値を入力すると、デフォルトを上書きできます。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。
- **DISCARD:** 破棄されたメッセージを入れるメール・フォルダの名前、またはメール・ファイルのフルパス名。名前の先頭に「-」が付いている場合、通知メーラーは起動時にそのフォルダまたはファイルを削除します。デフォルトは -discard です。

---

---

**注意：** UNIX Sendmail バージョンの通知メーラーの場合、DISCARD 値は常にメール・ファイルのフルパス名でなければなりません。

---

---

- **PROCESS:** 処理された通知メッセージを入れるメール・フォルダの名前、またはメール・ファイルのフルパス名。名前の先頭に「-」が付いている場合、通知メーラーは起動時にそのフォルダまたはファイルを削除します。デフォルトは processed です。

---

---

**注意：** UNIX Sendmail バージョンの通知メーラーの場合、PROCESS 値は常にメール・ファイルのフルパス名でなければなりません。

---

---

- **UNPROCESS:** 未処理の通知メッセージを入れるメール・フォルダの名前、またはメール・ファイルのフルパス名。名前の先頭に「-」が付いている場合、通知メーラーは起動時にそのフォルダまたはファイルを削除します。デフォルトは unprocessed です。

---

---

**注意：** UNIX Sendmail バージョンの通知メーラーの場合、UNPROCESS 値は常にメール・ファイルのフルパス名でなければなりません。

---

---

- **TAGFILE:** タグ・ファイルのフルパス名。タグ・ファイルには、特殊なメッセージで見つかった文字列と、メッセージの応答にそのような文字列が含まれている場合に割り当てられるステータスが表示されます。特殊なメッセージには、休暇デーモン

や一括メール・リストで送信されるものなど、返されたり、配信できなかったり、自動返信されたメッセージが含まれます。返されたメッセージ、配信不能メッセージまたは無効なメッセージの識別方法は、メール・システムによって異なります。そのため、タグ・ファイルを使用して、メール・システムによる宛先不明メッセージの識別方法と、そのメール・システムを通して送られてきたこのようなメッセージの通知メーラーによる処理方法を指定できます。

---

**注意：** タグ・ファイルによってチェックされるのは、通知 ID を持つメッセージ応答のみです。通知 ID を持たないメッセージ応答を受け取ると、通知メーラーはそのメッセージ応答を破棄フォルダに移動し、応答なしメールを受け取ったという警告メッセージを送信者に送ります。2-78 ページの「[ワークフロー警告メール](#)」メッセージ」を参照してください。

---

タグ・ファイルで使用される書式は、次のとおりです。

*Status "Matching string"*

*Status* 値は、ERROR、IGNORE または UNAVAIL で、*"Matching string"* は「送信元」行、「件名」行またはメッセージ本文で検索するテキストです。通知メーラーでは、このいずれかのステータス値が割り当てられたメッセージが次のように処理されます。

- IGNORE: メッセージを破棄フォルダに移動し、オープン通知に対する有効な返信を待ち続けます。通知のステータスは OPEN のままであり、メール・ステータスは SENT のままです。
- ERROR: メッセージを破棄フォルダに移動し、エラー処理が定義されていればそれを開始します。通知のステータスは OPEN のままですが、メールとアクティビティの各ステータスは ERROR に更新されます。原則的には、ワークフロー管理者が問題点を修正し、メール・ステータスを MAIL に更新して通知を再送信します。
- UNAVAIL (またはユーザーの定義した任意のタグ): 通知のステータスが OPEN のままであるため、メッセージを破棄フォルダに移動し、通知への返信を待ち続けます。ただし、メール・ステータスは UNAVAIL に更新されます。このステータスは情報表示のみのものです。この通知による追加処理はありません。

返信される応答値が、特定の選択肢（結果）タイプにある有効な値ではない場合、メッセージ応答に INVALID ステータスを割り当てることもできます。この場合、メッセージを破棄フォルダに移動し、無効なメッセージを送信しますが、通知のステータスやメール・ステータスは変更されないため、有効な返信を待ち続けます。2-75 ページの「[ワークフロー無効メール](#)」メッセージ」を参照してください。

---

**注意：** タグ・ファイルでは、返されたメッセージおよび自動返信メッセージを、通常の応答と一意に識別することが重要です。返されたメッセージおよび自動返信メッセージを識別しないと、通知メーラーはこれらが無効な応答と誤って判断し、無効メッセージを送って返信を待ち続けます。どちらの場合にも、通知メーラーが無効なメッセージを発信し、その無効なメッセージが返されるか自動返信されるという永久ループが発生してしまいます。

---

たとえば、件名またはメッセージ本文に文字列 "-- Unsent message follows --" を含むすべてのメッセージ応答にエラー・マークを設定する場合は、タグ・ファイルに次の行を含めます。

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

---

**注意：** メッセージ応答がタグ・ファイル内の複数の文字列と一致する場合は、一致する最初の文字列のステータスが付けられます。つまり、通知メーラーではタグ・ファイルに対して上から順に比較が実行されます。このような動作があるため、ERROR タグを先頭にして、次に UNAVAIL タグ、次に IGNORE タグという優先度順に文字列を並べる必要があります。

---

Oracle Workflow には、タグ・ファイルのサンプルとして wfmail.tag が用意されています。Oracle Workflow のスタンドアロン版を使用している場合、このファイルは Oracle Workflow Server のディレクトリ構造のうちサブディレクトリ res にあります。Oracle Applications embedded Workflow のバージョンの場合は、サーバー上の \$FND\_TOP の resource サブディレクトリにあります。

- **RESET\_FAILED:** この通知メーラーが起動したときに、FAILED ステータスの通知メールをすべて MAIL ステータスにリセットするかどうかを指定します。有効な値は Y または N で、デフォルトは N です。

この値が Y の場合、通知メーラーが起動したときに、FAILED ステータスの通知メールをすべて MAIL ステータスにリセットし、通常のメールとして処理しようとします。

- **RESET\_NLS:** 通知メーラーがメッセージを作成する前に、受信者の通知環境設定に従って通知メッセージの NLS コードセットを変換するかどうかを指定します。有効な値は Y または N で、デフォルトは N です。

この値が Y の場合、通知メーラーは受信者の Workflow ユーザー設定項目の言語および地域に基づいて、WF\_LANGUAGES 表のコードセットにメッセージを変換します。ユーザー設定項目に地域が指定されていない場合、ユーザー設定項目の言語に指定されている最初のエントリに関連付けられたコードセットを使用します。ユーザー設定項目に言語および地域が指定されていない場合は、WF\_LANGUAGES 表内で言語 AMERICAN および地域 AMERICA に割り当てられ



たコードセットを使用します。9-6 ページの「[ユーザー設定項目の設定](#)」を参照してください。

- **HTML\_MAIL\_TEMPLATE:** 通知環境設定が MAILHTML または MAILATTH のユーザーに対して、応答を必要とする電子メール通知のテンプレートとして「Workflow Open Mail for Outlook Express」メッセージを使用するには、このパラメータに OPEN\_MAIL\_OUTLOOK を指定します。電子メール・クライアントとして Microsoft Outlook Express などの電子メール・アプリケーションを使用している場合は、このメッセージ・テンプレートを選択すれば、「通知の詳細」Web 画面にリンクを追加して、ユーザーに対して通知への応答を要求できます。「ワークフロー・オープン・メール」および「ワークフロー・オープン・メール (ダイレクト)」テンプレートに応答リンクが含まれるときに、電子メール・アプリケーションが応答リンクを処理できないときは、このテンプレートを使用して対応できます。

通常の「ワークフロー・オープン・メール」および「ワークフロー・オープン・メール (ダイレクト)」テンプレートを使用する場合は、構成ファイルでこのパラメータをコメントにすれば、これらのテンプレートがデフォルトで使用されます。

---

**注意:** 作成した構成ファイルに HTML\_MAIL\_TEMPLATE パラメータが含まれている場合、通知メーラーは通知環境設定が MAILATTH のユーザーにメッセージを送信するときに、DIRECT\_RESPONSE パラメータを無視します。かわりに、通知環境設定が MAILATTH のユーザーは、「Workflow Open Mail for Outlook Express」テンプレートに定義されたプレーン・テキスト・メッセージを受信します。応答は、テンプレートに基づいて行われます。「Workflow Open Mail for Outlook Express」テンプレートを使用する必要がない場合は、HTML\_MAIL\_TEMPLATE パラメータをコメントにして、DIRECT\_RESPONSE パラメータを有効にする必要があります。

---

---

**注意:** HTML\_MAIL\_TEMPLATE パラメータは、通知環境設定が MAILTEXT のユーザーには適用されません。

---

- **SEND\_ACCESS\_KEY:** HTML 電子メール通知および HTML 添付ファイル付きの電子メール通知で送信される「通知の詳細リンク」添付ファイルにアクセス・キーを含めるには、このパラメータに Y を指定します。アクセス・キーを含めると、現在ログインしているかどうかにかかわらず、ユーザーは「通知の詳細リンク」をクリックして、「通知の詳細」Web 画面に直接アクセスできます。ただし、まだログインしていない場合は、アクセス・キーが添付されていない通知にアクセスすることはできません。「通知の詳細リンク」からアクセス・キーを除外するには、N を指定します。アクセス・キーを含まないリンクをクリックしたときに、まだログインしていない場合は、「通知の詳細」Web 画面にアクセスする前にログインを求めるプロンプトが表示されます。デフォルトは Y です。

**参照:** 10-26 ページの「[自動通知処理ルールの定義](#)」を参照してください。

## ➤ 通知メーラーの常駐シェル・スクリプトの実行

1. Oracle Workflow のスタンドアロン版を実行している場合は、障害のためにシャットダウンした場合に、その通知メーラーを再起動する常駐シェル・スクリプトを設定する必要があります。Oracle Workflow には、UNIX Sendmail の通知メーラーを再起動するためのサンプル・シェル・スクリプトが用意されています。このスクリプトは `wfmail.csh` という名前で、サーバー上の Oracle ホームの `bin` サブディレクトリにあります。

---

**注意：** Windows NT の通知メーラーを再起動する場合も、同じ技法を使用します。

---

2. オペレーティング・システムのスクリプト・プロンプトから、次のコマンドを入力してシェル・スクリプトを実行します。

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

`<config_file>` は、通知メーラーの実行時に指定するパラメータを含む構成ファイルのフルパス名に置き換えます。シェル・スクリプトにより、すべてのコマンドライン引数が通知メーラーの実行ファイルに直接渡されます。

## 応答処理

通知メーラーを起動して応答を処理する前に、応答メール・アカウントで3つのフォルダまたはファイルを作成する必要があります。この3つのフォルダまたはファイルには、破棄されたメッセージ、未処理のメッセージおよび処理済のメッセージが保持されます。

通知メーラーは次の処理を行って、応答メッセージをチェックします。

- 応答用メール・アカウントにログインします。
- メッセージをチェックします。メッセージがあればそれを読み込み、通知 ID とノード識別子をチェックします。
- メッセージが通知でない場合は、それを破棄フォルダに移動します。
- メッセージが現行のノードに対する通知であれば、それを未処理フォルダに移動します。
- メッセージが異なるノード宛の通知の場合は、後で正しいノードの通知メーラーが読み込めるようにメッセージを移動しません。

次に、通知メーラーは未処理フォルダをオープンし、各応答を処理します。メッセージごとに次の処理を行います。

- 通知 ID を取り出します。
- 指定されたタグ・ファイルがある場合はそれを参照し、そのメッセージがバウンスされたのかどうかをチェックします。メッセージがバウンスされている場合は、タグ・ファ

イルの指定に応じて、それを再ルーティングするか、または通知のステータスを更新してそれ以降の処理を中止します。

- Oracle Workflow データベース内で、この通知をチェックします。
  - － 通知が存在しない場合は、それを破棄フォルダに移動します。
  - － 通知があっても、完了または取消しされている場合は、破棄フォルダに移動します。
  - － 通知が存在していて処理中の場合は、データベースにあるメッセージの応答属性の定義を使用して応答値を検証します。応答が無効な場合は、宛先のロールに「ワークフロー無効のメール」メッセージを送信します。応答が有効な場合には、応答関数をコールして通知の応答を完了し、変更をデータベースに保存します。
- 完了した通知メッセージを処理済フォルダに移動し、未処理フォルダを閉じます。

構成ファイルで指定された `discard` パラメータと `process` パラメータが「-」で始まっている場合、通知メーラーは破棄フォルダと処理済フォルダの中身を削除し、メール・アカウントとデータベース・アカウントからログアウトします。

## 手順 WF-10 メッセージ・テンプレートの変更

Oracle Workflow Builder の「システム：メーラー」項目タイプを使用すると、Oracle Workflow が電子メール通知の送信に使用するテンプレートを構成できます。「システム：メーラー」項目タイプには、通知メッセージの各部分を表す属性があります。これらの属性のレイアウトを各テンプレート内で変更して、通知システムによって送信される電子メール・メッセージをカスタマイズできます。

「システム：メーラー」項目タイプのメッセージは、本当のメッセージではなく、通知システムによって送信される電子メール・メッセージのテンプレートとして機能します。「システム：メール・システム」メッセージによって、組み込まれるヘッダー情報、メッセージの期日と優先度などの詳細を含めるかどうか、または含める場合はどこに入れるかなど、電子メール通知の基本的な書式が決まります。

---

**警告：**「システム：メーラー」項目タイプのメッセージ・テンプレートに対して、新規の属性を追加したり、既存の属性を削除したりしないでください。

---

**コンテキスト：**この手順を実行する必要があるのは、1 度のみです。

## 「ワークフロー・オープン・メール」メッセージ

テンプレートによる応答方法を選択すると、通知システムでは、応答を必要とする電子メール通知のテンプレートとして、「ワークフロー・オープン・メール」メッセージが使用されます。この通知テンプレートには、通知の応答方法に関する一般的な指示が含まれています。また、メッセージ優先度、応答期日、通知が他のユーザーから転送されたものかどうか

か、メッセージの送信者や転送者からのコメントなど、メッセージに関する情報も含まれます。

**注意：** テンプレートによる応答方法を選択するには、通知メーラーの構成ファイル内で `DIRECT_RESPONSE=N` に設定します。2-55 ページの「[通知メーラーの構成ファイルの作成](#)」を参照してください。

プレーン・テキスト・メッセージ本文に含まれる応答指示では、テンプレートによる応答方法を使用して手動で返信する方法が記述されます。このメッセージは、通知環境設定を `MAILTEXT` または `MAILATTH` に指定している実行者に送られる通知に使用されます。HTML 形式のメッセージ本文に含まれる応答指示では、自動的に生成される応答テンプレートを使用して返信する方法が記述されます。このメッセージは、通知環境設定を `MAILHTML` に指定している実行者に送られる通知に使用されます。また、通知環境設定を `MAILATTH` に指定している実行者に送られる通知にも添付されます。

「ワークフロー・オープン・メール」メッセージには、次のメッセージ属性があります。その値は、通知アクティビティに関連付けられているメッセージ定義から取り出されます。

<b>START_DATE</b>	メッセージの送信日。
<b>TO</b>	通知の送信先のロール（実行者）。
<b>SUBJECT</b>	メッセージで定義される件名。
<b>BODY</b>	メッセージで定義される本文のテキスト。
<b>COMMENT</b>	送信者または転送者によって追加されるコメント。
<b>PRIORITY</b>	通知メッセージの優先度。
<b>DUE_DATE</b>	通知アクティビティで指定される応答期日。
<b>NOTIFICATION</b>	通知情報の識別に使用される必須の通知コード。
<b>RESPONSE</b>	実際の通知メッセージ定義の「応答」メッセージ属性で定義されるユーザー応答セクション。
<b>MAILTO</b>	受信者が通知への応答上でクリックする <code>HTML</code> タグの内容。この属性は、 <code>HTML</code> 形式の電子メール通知にのみ使用されます。
<b>CLICK_HERE_RESPONSE</b>	受信者が「通知の詳細」ページにアクセスして通知に応答するときにクリックする、 <code>HTML</code> タグの内容。この属性は現在使用されていません。

「ワークフロー・オープン・メール」メッセージの本文に表示されるボイラープレート・テキストをカスタマイズできます。このメッセージでは、アンパサンド（&）で始まる属性は、通知の送信時に実行時の値に置き換えられるトークンです。

プレーン・テキスト・メッセージ本文のボイラープレート・テキストは、次のとおりです。

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

HTML 形式のメッセージ本文のボイラプレート・テキストは、次のとおりです。

```
<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>
```

## 「ワークフロー・オープン・メール（ダイレクト）」メッセージ

直接応答方法を選択すると、通知システムでは、応答を必要とする電子メール通知のテンプレートとして、「ワークフロー・オープン・メール（ダイレクト）」メッセージが使用されます。この通知テンプレートには、通知の応答方法に関する一般的な指示が含まれています。また、メッセージ優先度、応答期日、通知が他のユーザーから転送されたものかどうか、メッセージの送信者や転送者からのコメントなど、メッセージに関する情報も含まれます。

**注意：** 直接応答方法を選択するには、通知メーラーの構成ファイル内で `DIRECT_RESPONSE=Y` に設定します。2-55 ページの「[通知メーラーの構成ファイルの作成](#)」を参照してください。

プレーン・テキスト・メッセージ本文に含まれる応答指示では、直接応答方法を使用して返信する方法が記述されます。このメッセージは、通知環境設定を `MAILTEXT` または `MAILATTH` に指定している実行者に送られる通知に使用されます。HTML 形式のメッセージ本文に含まれる応答指示では、自動的に生成される応答テンプレートを使用して返信する方法が記述されます。このメッセージは、通知環境設定を `MAILHTML` に指定している実行者に送られる通知に使用されます。また、通知環境設定を `MAILATTH` に指定している実行者に送られる通知にも添付されます。

**注意：** HTML 形式の通知または添付ファイルから自動的に生成される応答には、`DIRECT_RESPONSE` パラメータで選択した応答方法に関係なく、常に応答テンプレートが使用されます。

「ワークフロー・オープン・メール（ダイレクト）」メッセージには、次のメッセージ属性があります。その値は、通知アクティビティに関連付けられているメッセージ定義から取り出されます。

<b>START_DATE</b>	メッセージの送信日。
<b>TO</b>	通知の送信先のロール（実行者）。
<b>SUBJECT</b>	メッセージで定義される件名。
<b>BODY</b>	メッセージで定義される本文のテキスト。
<b>COMMENT</b>	送信者または転送者によって追加されるコメント。
<b>PRIORITY</b>	通知メッセージの優先度。
<b>DUE_DATE</b>	通知アクティビティで指定される応答期日。
<b>NOTIFICATION</b>	通知情報の識別に使用される必須の通知コード。
<b>RESPONSE</b>	実際の通知メッセージ定義の「応答」メッセージ属性で定義されるユーザー応答セクション。
<b>MAILTO</b>	受信者が通知への応答上でクリックする <code>HTML</code> タグの内容。この属性は、 <code>HTML</code> 形式の電子メール通知にのみ使用されます。
<b>CLICK_HERE_RESPONSE</b>	受信者が「通知の詳細」ページにアクセスして通知に応答するときにクリックする、 <code>HTML</code> タグの内容。この属性は現在使用されていません。

「ワークフロー・オープン・メール（ダイレクト）」メッセージの本文に表示されるボイラープレート・テキストをカスタマイズできます。このメッセージでは、アンパサンド（&）で始まる属性は、通知の送信時に実行時の値に置き換えられるトークンです。

プレーン・テキスト・メッセージ本文のボイラープレート・テキストは、次のとおりです。

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

HTML 形式のメッセージ本文のボイラープレート・テキストは、次のとおりです。

```
<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」メッセージ

電子メール・クライアントとして Microsoft Outlook Express などの電子メール・アプリケーションを使用する場合は、通知環境設定が MAILHTML または MAILATTH のユーザーに対して、応答を必要とする電子メール通知のテンプレートとして「Workflow Open Mail for Outlook Express」メッセージを選択する必要があります。このメッセージには、「通知の詳細」Web 画面へのリンクが含まれており、ユーザーに対して通知への応答を要求できます。「ワークフロー・オープン・メール」および「ワークフロー・オープン・メール（ダイレクト）」テンプレートに応答リンクが含まれるときに、電子メール・アプリケーションが応答リンクを処理できないときは、このテンプレートを使用して対応できます。

---

**注意：** 通知環境設定が MAILHTML または MAILATTH のユーザーのメッセージ・テンプレートとして「Workflow Open Mail for Outlook Express」メッセージを選択するには、通知メーラーの構成ファイルで HTML\_MAIL\_TEMPLATE=OPEN\_MAIL\_OUTLOOK を設定します。2-55 ページの「[通知メーラーの構成ファイルの作成](#)」を参照してください。

---

---

**注意：** 作成した構成ファイルに HTML\_MAIL\_TEMPLATE パラメータが含まれている場合、通知メーラーは通知環境設定が MAILATTH のユーザーにメッセージを送信するときに、DIRECT\_RESPONSE パラメータを無視します。かわりに、通知環境設定が MAILATTH のユーザーは、「Workflow Open Mail for Outlook Express」テンプレートに定義されたプレーン・テキスト・メッセージを受信します。応答は、テンプレートに基づいて行われます。「Workflow Open Mail for Outlook Express」テンプレートを使用する必要がない場合は、HTML\_MAIL\_TEMPLATE パラメータをコメントにして、DIRECT\_RESPONSE パラメータを有効にする必要があります。

---

プレーン・テキスト・メッセージ本文に含まれる応答指示では、テンプレートによる応答方法を使用して手動で返信する方法が記述されます。このメッセージは、通知環境設定が MAILATTH の実行者に送られる通知に使用されます。HTML 形式のメッセージ本文には、「Click here to respond」というリンクがあり、「通知の詳細」Web 画面の通知にアクセスしたユーザーに対して応答を要求できます。このメッセージは、通知環境設定を MAILHTML に指定している実行者に送られる通知に使用されます。また、通知環境設定を MAILATTH に指定している実行者に送られる通知にも添付されます。

---

**注意：** ユーザーが「Click here to respond」リンクを選択すると、Web サーバーとの Web セッションが自動的に確立されます。このリンクを使用して通知に応答するには、Web サーバーに接続する必要があります。10-2 ページの「[電子メールによる通知の閲覧](#)」を参照してください。

---

「Workflow Open Mail for Outlook Express」メッセージには、次のメッセージ属性があります。その値は、通知アクティビティに関連付けられているメッセージ定義から取り出されます。

START_DATE	メッセージの送信日。
TO	通知の送信先のロール（実行者）。
SUBJECT	メッセージで定義される件名。
BODY	メッセージで定義される本文のテキスト。
COMMENT	送信者または転送者によって追加されるコメント。



<b>PRIORITY</b>	通知メッセージの優先度。
<b>DUE_DATE</b>	通知アクティビティで指定される応答期日。
<b>NOTIFICATION</b>	通知情報の識別に使用される必須の通知コード。
<b>RESPONSE</b>	実際の通知メッセージ定義の「応答」メッセージ属性で定義されるユーザー応答セクション。
<b>MAILTO</b>	受信者が通知への応答上でクリックする HTML タグの内容。この属性は現在使用されていません。
<b>CLICK_HERE_RESPONSE</b>	受信者が「通知の詳細」ページにアクセスして通知に応答するときにクリックする、HTML タグの内容。この属性は、HTML 形式の電子メール通知にのみ使用されます。

「Workflow Open Mail for Outlook Express」メッセージの本文に表示されるボイラープレート・テキストは、カスタマイズできます。アンパサンド (&) で始まる属性は、通知の送信時に実行時の値に置き換えられるトークンです。

プレーン・テキスト・メッセージ本文のボイラープレート・テキストは、次のとおりです。

```
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
```

HTML 形式のメッセージ本文のボイラープレート・テキストは、次のとおりです。

```
<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>
```

## 「ワークフロー・オープンFYI メール」メッセージ

通知システムでは、応答の必要がないすべての電子メール通知のテンプレートとして、「ワークフロー・オープン参考メール」メッセージが使用されます。このテンプレートは、その通知が参考用（FYI: For Your Information）で、応答の必要がないことを示します。このテンプレートには、メッセージに加えて、そのメッセージの送信者や転送者からのコメントも含まれます。

「ワークフロー・オープン参考メール」メッセージには、次のメッセージ属性があります。その値は、通知アクティビティに関連付けられているメッセージ定義から取り出されます。

<b>START_DATE</b>	メッセージの送信日。
<b>TO</b>	通知の送信先のロール（実行者）。
<b>SUBJECT</b>	メッセージで定義される件名。
<b>BODY</b>	メッセージで定義される本文のテキスト。
<b>COMMENT</b>	送信者または転送者によって追加されるコメント。
<b>PRIORITY</b>	通知メッセージの優先度。
<b>DUE_DATE</b>	通知アクティビティで指定される応答期日。
<b>NOTIFICATION</b>	通知情報の識別に使用される必須の通知コード。

「ワークフロー・オープン参考メール」テンプレートの本文に表示されるテキストをカスタマイズできます。このメッセージでは、アンパサンド（&）で始まる属性は、通知の送信時に実行時の値に置き換えられるトークンです。プレーン・テキスト・メッセージ本文のボイラープレート・テキストは、次のとおりです。

```
Oracle Workflow Notification (FYI)
&COMMENT

-----
&BODY
```

HTML 形式のメッセージ本文のボイラープレート・テキストは、次のとおりです。

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

## 「Workflow URL Attachment」メッセージ

通知システムでは、「Workflow URL Attachment」メッセージをテンプレートとして使用して、URL 属性の「内容の添付」がオンになっている HTML 形式の通知メッセージに対して「通知参照」添付ファイルを作成します。このテンプレートには、各 URL へのリンクを示すリストが含まれています。

「Workflow URL Attachment」メッセージには、次のメッセージ属性があります。その値は、通知アクティビティに関連付けられているメッセージ定義から取り出されます。

**URLLIST**                      添付ファイルに含める URL のリスト。

「Workflow URL Attachment」テンプレートの本文に表示されるテキストは、カスタマイズできます。アンバサンド（&）で始まる属性は、通知の送信時に実行時の値に置き換えられるトークンです。HTML 形式のメッセージ本文のボイラープレート・テキストは、次のとおりです。

```
<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>
```

「ワークフロー取消のメール」メッセージ

「ワークフロー取消のメール」メッセージは、以前に送信された通知が取り消されたことを受領者に知らせます。このメッセージには次のメッセージ属性があり、その値は取り消された通知アクティビティに関連したメッセージ定義から取り出されます。

START_DATE	元のメッセージの送信日。
TO	通知の送信先のロール（実行者）。
SUBJECT	元のメッセージの件名。
BODY	元のメッセージのテキスト。
COMMENT	送信者または転送者によって追加されるコメント。
PRIORITY	通知メッセージの優先度。
DUE_DATE	通知アクティビティで指定される応答期日。
NOTIFICATION	通知情報の識別に使用される必須の通知コード。

プレーン・テキスト・メッセージ本文のボイラープレート・テキストは、次のとおりです。

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

HTML 形式のメッセージ本文のボイラープレート・テキストは、次のとおりです。

```
<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>
```

## 「ワークフロー無効メール」メッセージ

「ワークフロー無効メール」メッセージは、通知に対するユーザーの応答が正しくないときに送信されます。メッセージは、通知に正しく応答する方法を説明します。メッセージ属性は、次のとおりです。

<b>START_DATE</b>	元のメッセージの送信日。
<b>TO</b>	通知の送信先のロール（実行者）。
<b>SUBJECT</b>	元のメッセージの件名。
<b>BODY</b>	元のメッセージのテキスト。
<b>COMMENT</b>	送信者または転送者によって追加されるコメント。
<b>PRIORITY</b>	通知メッセージの優先度。
<b>DUE_DATE</b>	通知アクティビティで指定される応答期日。
<b>NOTIFICATION</b>	通知情報の識別に使用される必須の通知コード。
<b>RESPONSE</b>	元のメッセージ定義の「応答」メッセージ属性で定義されるユーザー応答セクション。
<b>MAIL_ERROR_MESSAGE</b>	応答処理中にエラーが発生した場合に、メール・プログラムによって生成されるエラー・メッセージ。
<b>MAIL_ERROR_STACK</b>	応答処理中にエラーが発生した場合に、メール・プログラムによって生成される引数のエラー・スタック。応答を修正しても問題を解決できない場合は、オラクル社カスタマ・サポート・センターにお問い合わせください。
<b>CLICK_HERE_RESPONSE</b>	受信者が「通知の詳細」ページにアクセスして通知に応答するときにクリックする、HTML タグの内容。この属性は現在使用されていません。

プレーン・テキスト・メッセージ本文のボイラープレート・テキストは、次のとおりです。

```
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

HTML 形式のメッセージ本文のボイラープレート・テキストは、次のとおりです。

```
<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>
```

「ワークフロー・クローズ・メール」メッセージ

「ワークフロー・クローズ・メール」メッセージは、以前に送信された通知が完了したことを受領者に知らせます。このメッセージは次のメッセージ属性を持ち、その値は完了した通知アクティビティに関連付けられているメッセージ定義から取り出されます。

START_DATE	元のメッセージの送信日。
TO	通知の送信先のロール（実行者）。
SUBJECT	元のメッセージの件名。
BODY	元のメッセージのテキスト。
COMMENT	送信者または転送者によって追加されるコメント。
PRIORITY	通知メッセージの優先度。
DUE_DATE	通知アクティビティで指定される応答期日。
NOTIFICATION	通知情報の識別に使用される必須の通知コード。

プレーン・テキスト・メッセージ本文のボイラープレート・テキストは、次のとおりです。

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

HTML 形式のメッセージ本文のボイラープレート・テキストは、次のとおりです。

```
<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>
```

## 「ワークフロー要約メール」メッセージ

通知システムでは、Oracle Workflow のディレクトリ・サービス内で通知環境設定が SUMMARY に設定されているユーザーとロールに Workflow 通知の要約を送信するためのテンプレートとして、「ワークフロー要約メール」メッセージが使用されます。「ワークフロー要約メール」メッセージは、特定のユーザーまたはロールに関して現在処理中の全通知の要約です。このメッセージは次のメッセージ属性を持ち、その値は処理中の通知アクティビティに関連付けられているメッセージ定義から取り出されます。

<b>SUMMARY</b>	要約レポート。
<b>USER_NAME</b>	通知要約の発信先となるユーザー / ロール（実行者）。
<b>SYSDATE</b>	現在の日付。

プレーン・テキスト・メッセージ本文のボイラープレート・テキストは、次のとおりです。

Summary of Notifications for '&USER\_NAME'  
(Please use the Notifications web page to see details or respond.)

-----  
&SUMMARY

HTML 形式のメッセージ本文のボイラープレート・テキストは、次のとおりです。

```
<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>
```

## 「ワークフロー警告メール」メッセージ

通知システムでは、ユーザーから不要なメールを受け取った場合に、そのユーザーにメッセージを送信するためのテンプレートとして、「ワークフロー警告メール」メッセージが使用されます。このメッセージには、不要なメールから取り出した値と次のようなメッセージ属性があります。

<b>UBODY</b>	不要なメール・メッセージのテキスト。
<b>USUBJECT</b>	不要なメールの件名のテキスト。
<b>UFROM</b>	不要なメールを送信したユーザーのアドレス。

プレーン・テキスト・メッセージ本文のボイラープレート・テキストは、次のとおりです。

```
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
```

HTML 形式のメッセージ本文のボイラープレート・テキストは、次のとおりです。

```
<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>
```



## 手順 WF-11 Oracle Workflow の Web 画面のロゴのカスタマイズ

Oracle Workflow の Web 画面とワークフロー・モニターをお使いのサイトで使用するには、Oracle HTTP Server をインストールしておく必要があります。詳細は、使用している Web サーバーのドキュメントを参照してください。

Web サーバーのインストールおよび設定後に、Oracle Workflow の Web ページに表示される会社のロゴをカスタマイズできます。

JavaScript がサポートされる Web ブラウザを使用して「通知」Web 画面に接続するか、バージョン 1.1.8 以上の Java Development Kit (JDK) または Abstract Windowing Toolkit (AWT) がサポートされる Web ブラウザを使用して、ワークフロー・モニターに接続します。

### ► Oracle Workflow の Web 画面のカスタマイズ

Oracle Workflow の Web 画面右上隅に表示される会社のロゴをカスタマイズします。

1. 会社のロゴ・ファイル (.gif 形式) をコピーするか、ファイル名を変更して、FNDLOGOS.gif (Oracle Applications embedded Workflow を使用している場合) または WFLOGO.gif (Oracle Workflow のスタンドアロン版を使用している場合) を作成します。
2. このファイルを、Web サーバーの /OA\_MEDIA/ 仮想ディレクトリが指す物理ディレクトリに移動します。

---

---

**注意：** Oracle Applications embedded Workflow を使用している場合、/OA\_MEDIA/ のマッピングは Oracle Applications のインストールおよび設定手順の一部として実行されます。

---

---

---

---

**注意：** スタンドアロン版の Oracle Workflow を使用している場合、/OA\_MEDIA/ のマッピングは Oracle Workflow Server のインストールとワークフロー・モニターの設定後に実行されます。

---

---

**コンテキスト：** この手順を実行する必要があるのは、1 度のみです。

## 手順 WF-12 Oracle Workflow へのカスタム・アイコンの追加

Oracle Workflow Builder では、PC 上の Oracle Workflow 領域の Icon サブディレクトリ内でアイコンが検索されます。Icon サブディレクトリは、Oracle Workflow Builder のレジストリ内で定義されています。通常、Oracle Workflow 領域は、ORACLE ホーム・ディレクトリの wf サブディレクトリです。

Oracle Workflow には、アクティビティやプロセスに使用できるように、様々なアイコンが用意されています。この領域には、.ico 拡張子が付いた Windows のアイコン・ファイルであればどのファイルでも追加できます。

カスタム・アイコンを Oracle Workflow Builder のプロセス定義に組み込んで、プロセスを表示するときにワークフロー・モニターに表示する場合は、次の操作を実行する必要があります。

- カスタム・アイコン・ファイル (.ico) を gif 形式 (.gif) に変換します。
- Web サーバーの /OA\_MEDIA/ 仮想ディレクトリが指す物理ディレクトリに .gif ファイルをコピーして、ワークフロー・モニターでそれらのファイルにアクセスできるようにします。

---

---

**注意：** Oracle Applications embedded Workflow を使用している場合、/OA\_MEDIA/ のマッピングは Oracle Applications のインストールおよび設定手順の一部として実行されます。

---

---

---

---

**注意：** スタンドアロン版の Oracle Workflow を使用している場合、/OA\_MEDIA/ のマッピングは Oracle Workflow Server のインストールとワークフロー・モニターの設定後に実行されます。

---

---

**コンテキスト：** この手順を実行する必要があるのは、1 度のみです。

## 手順 WF-13 Java 関数アクティビティ・エージェントの設定

外部 Java 関数アクティビティを実行するには、Java 関数アクティビティ・エージェントを設定する必要があります。現在、この機能を使用できるのは Oracle Workflow のスタンドアロン版のみです。Java 関数アクティビティ・エージェントは、外部 Java アクティビティに関連するメッセージを外部関数処理用の送信キューからデキューし、該当する Java 関数をコールして、結果を外部関数処理用の受信キューに入れます。

---

---

**注意：** これらの送信キューおよび受信キューは、ビジネス・イベント・システムに使用するキューとは異なります。8-167 ページの「[Workflow QUEUE API](#)」を参照してください。

---

---

Java 関数が完了したら、バックグラウンド・エンジンを実行して受信キューを処理し、関数アクティビティを実行する必要があります。2-40 ページの「[手順 WF-8 バックグラウンドのワークフロー・エンジンの設定](#)」を参照してください。

標準 Workflow アクティビティの一部は外部の Java 関数アクティビティで、Java 関数アクティビティ・エージェントを必要とします。また、独自の外部 Java 関数アクティビティを定義することもできます。6-2 ページの「[標準アクティビティ](#)」、4-47 ページの「[関数アクティビティの作成](#)」、7-8 ページの「[関数アクティビティがコールする Java プロシージャの標準 API](#)」を参照してください。

Java 関数アクティビティ・エージェントを実行するには、Java Runtime Environment (JRE) バージョン 1.1.8 以上をインストールしておく必要があります。

---

---

**注意：** Java Runtime Environment は、<http://www.javasoft.com> からダウンロードして利用できます。

---

---

**コンテキスト：** この手順を実行する必要があるのは、1 度のみです。

## Java 関数アクティビティ・エージェントの起動

Oracle Workflow のスタンドアロン版を使用している場合、Oracle Workflow が提供するスクリプトを実行して Java 関数アクティビティ・エージェントを起動できます。このエージェントは手動で起動することもできます。

Java 関数アクティビティ・エージェントを起動するときは、データベース接続の詳細を指定する必要があります。使用するキャラクタ・セットおよび JDBC ドライバのタイプを指定することもできます。

実行方法（スクリプトからまたは手動で）、実行するプラットフォーム、および指定するオプションに応じて、エージェントの起動に使用するコマンドが異なります。

## スクリプトからの Java 関数アクティビティ・エージェントの起動

Oracle Workflow のスタンドアロン版を使用している場合、wfjvlsnr.csh (UNIX の場合) または wfjvlsnr.bat (Windows NT の場合) というスクリプトを実行して、Java 関数アクティビティ・エージェントを起動できます。これらのスクリプトは、サーバーの Oracle Workflow の admin サブディレクトリに格納されています。

独自の外部 Java 関数アクティビティを定義する場合は、スクリプトを編集してカスタム Java クラスが入っている JAR ファイルへのパスを追加する必要があります。カスタム・クラス・ファイルは、Java 関数アクティビティ・エージェントが実行されるのと同じプラットフォームに常駐させる必要があります。ただし、Java 関数アクティビティ・エージェントをデータベースと同じ層に常駐させる必要はありません。

プラットフォームおよび指定するオプションに応じて、異なる構文のコマンドを使用してスクリプトを実行できます。

### UNIX での wfjvlsnr.csh スクリプトの実行

たとえば、UNIX では、次のコマンドを使用して wfjvlsnr.csh スクリプトを実行できます (デフォルトの JDBC OCI8 ドライバを使用する場合)。

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

コマンドのパラメータを次のように置き換えます。

- **<user\_name>**: Oracle Workflow データベース・アカウントのユーザー名を指定します。
- **<password>**: Oracle Workflow データベース・アカウントのパスワードを指定します。
- **<connect\_string>**: データベースの接続文字列。このコマンドはデフォルトの JDBC OCI8 ドライバを使用するため、接続文字列には、TNSNAMES エントリに指定されたデータベース名を指定する必要があります。
- **<character\_set>**: データベース・セッションで使用するキャラクタ・セット。キャラクタ・セットを指定しない場合は、デフォルトで UTF8 が使用されます。

UNIX では、次のコマンドを使用して wfjvlsnr.csh スクリプトを実行することもできます (使用する JDBC ドライバのタイプを指定する場合)。

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

コマンドのパラメータを次のように置き換えます。

- **<user\_name>**: Oracle Workflow データベース・アカウントのユーザー名を指定します。

- **<password>**: Oracle Workflow データベース・アカウントのパスワードを指定します。
- **<connect\_string>**: データベースの接続文字列。接続文字列の形式は、JDBC ドライバのタイプに応じて変わります。
  - JDBC OCI8 ドライバの場合は、TNSNAMES エントリに指定されたデータベース名を次の形式で指定する必要があります。  
   <database\_name>
  - JDBC Thin ドライバの場合は、2 種類の接続文字列を使用できます。一方の接続文字列では、ホスト名、ポート番号およびデータベース・システム識別子 (SID) を次の形式で指定する必要があります。  
   <host\_name>:<port\_number>:<database\_SID>

もう一方の接続文字列では、ホスト名、プロトコル、ポート番号および SID で構成される Oracle Net の名前 - 値ペアを、次の形式で指定する必要があります。  
 (description=(address=(host=<host\_name>) (protocol=<protocol>) (port=<port\_number>)) (connect\_data=(sid=<database\_SID>)))
- **<JDBC\_driver>**: データベース接続に使用する JDBC ドライバのタイプ。JDBC ドライバのタイプには、oci8 または thin を指定できます。キャラクタ・セットを指定しない場合は、デフォルトで JDBC OCI8 ドライバが使用されます。

---

**注意:** ユーザー名、パスワード、接続文字列および JDBC ドライバのタイプを含む接続詳細は、二重引用符で囲んでキャラクタ・セット・パラメータと切り離す必要があります。

---

- **<character\_set>**: データベース・セッションで使用するキャラクタ・セット。キャラクタ・セットを指定しない場合は、デフォルトで UTF8 が使用されます。

### Windows NT での wfjvlsnr.bat スクリプトの実行

Windows NT では、次のコマンドを使用して wfjvlsnr.bat スクリプトを実行できます (デフォルトの JDBC OCI8 ドライバを使用する場合)。

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

コマンドのパラメータを次のように置き換えます。

- **<user\_name>**: Oracle Workflow データベース・アカウントのユーザー名を指定します。
- **<password>**: Oracle Workflow データベース・アカウントのパスワードを指定します。

- `<connect_string>`: データベースの接続文字列。このコマンドはデフォルトの JDBC OCI8 ドライバを使用するため、接続文字列には、TNSNAMES エントリに指定されたデータベース名を指定する必要があります。
- `<character_set>`: データベース・セッションで使用するキャラクタ・セット。キャラクタ・セットを指定しない場合は、デフォルトで UTF8 が使用されます。

Windows NT では、次のコマンドを使用して wfjvlsnr.bat スクリプトを実行することもできます (使用する JDBC ドライバのタイプを指定する場合)。

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

コマンドのパラメータを次のように置き換えます。

- `<user_name>`: Oracle Workflow データベース・アカウントのユーザー名を指定します。
- `<password>`: Oracle Workflow データベース・アカウントのパスワードを指定します。
- `<connect_string>`: データベースの接続文字列。接続文字列の形式は、JDBC ドライバのタイプに応じて変わります。
  - JDBC OCI8 ドライバの場合は、TNSNAMES エントリに指定されたデータベース名を次の形式で指定する必要があります。  
`<database_name>`
  - JDBC Thin ドライバの場合は、2 種類の接続文字列を使用できます。一方の接続文字列では、ホスト名、ポート番号およびデータベース・システム識別子 (SID) を次の形式で指定する必要があります。  
`<host_name>:<port_number>:<database_SID>`
  - もう一方の接続文字列では、ホスト名、プロトコル、ポート番号および SID で構成される Oracle Net の名前 - 値ペアを、次の形式で指定する必要があります。  
`(description=(address=(host=<host_name>)(protocol=  
<protocol>)(port=<port_number>))(connect_data=(sid=  
<database_SID>)))`
- `<JDBC_driver>`: データベース接続に使用する JDBC ドライバのタイプ。JDBC ドライバのタイプには、oci8 または thin を指定できます。キャラクタ・セットを指定しない場合は、デフォルトで JDBC OCI8 ドライバが使用されます。

---

**注意：** ユーザー名、パスワード、接続文字列および JDBC ドライバのタイプを含む接続詳細は、二重引用符で囲んでキャラクタ・セット・パラメータと切り離す必要があります。

---

- `<character_set>`: データベース・セッションで使用するキャラクタ・セット。キャラクタ・セットを指定しない場合は、デフォルトで UTF8 が使用されます。

## 手動による Java 関数アクティビティ・エージェントの起動

Java 関数アクティビティ・エージェントを手動で起動するには、CLASSPATH、Oracle Workflow データベース・アカウントのユーザー名とパスワード、およびデータベース接続文字列を指定して、`oracle.apps.fnd.wf.WFFALsnr` に対して JRE を実行します。使用するキャラクタ・セットおよび JDBC ドライバのタイプを指定することもできます。

この CLASSPATH は、Java Runtime Environment、Workflow JAR ファイルが入っているディレクトリ、Oracle XML Parser、Oracle JDBC 実装および次の Workflow JAR ファイルを指している必要があります。

- `wfjava.jar`: Java 関数アクティビティ・エージェント
- `wfapi.jar`: Workflow Java API
- Share JAR ファイル: Workflow Java API によって参照されるユーティリティ。Oracle9i で Oracle Workflow のスタンドアロン版を使用している場合、このファイルの名前は `share-<version>.jar` (`share-1_1_9.jar` など、または現行バージョン) となります。Oracle Applications embedded Workflow では、このファイルの名前は `fndbalishare.jar` となります。
- Ewt JAR ファイル: Workflow Java API によって参照されるユーティリティ。Oracle9i で Oracle Workflow のスタンドアロン版を使用している場合、このファイルの名前は `ewt-<version>.jar` (`ewt-3_3_18.jar` など、または現行バージョン) となります。Oracle Applications embedded Workflow では、このファイルの名前は `fndewt.jar` となります。
- Swing JAR ファイル: オプションのユーティリティ。Oracle9i で Oracle Workflow のスタンドアロン版を使用している場合、このファイルの名前は `swingall-<version>.jar` (`swingall-1_1_1.jar` など、または現行バージョン) となります。Oracle Applications embedded Workflow では、このファイルの名前は `fndswing.jar` となります。

---

**注意：** Oracle9i で Oracle Workflow スタンドアロン版を使用している場合、Workflow JAR ファイルは <ORACLE\_HOME>/jlib ディレクトリに格納されています。Oracle Applications embedded Workflow では、Workflow JAR ファイルは <ORACLE\_HOME>/wf/java/oracle/apps/fnd/wf/jar/ ディレクトリに格納されています。

---

独自の外部 Java 関数アクティビティを定義する場合は、カスタム Java クラスが入っている JAR ファイルも CLASSPATH に追加する必要があります。カスタム・クラス・ファイルは、Java 関数アクティビティ・エージェントが実行されるのと同じプラットフォームに常駐させる必要があります。ただし、Java 関数アクティビティ・エージェントをデータベースと同じ層に常駐させる必要はありません。

プラットフォームおよび指定するオプションに応じて、異なる構文のコマンドを使用して Java 関数アクティビティ・エージェントを手動で起動できます。

### UNIX での Java 関数アクティビティ・エージェントの起動

たとえば、UNIX では、次のコマンドを使用して Java 関数アクティビティ・エージェントを起動できます（デフォルトの JDBC OCI8 ドライバを使用する場合）。

```
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>
```

このコマンドでは、-DCHARSET プシオンを使用して、使用するキャラクタ・セットを指定することもできます。キャラクタ・セットを指定しない場合は、デフォルトで UTF8 が使用されます。

コマンドのパラメータを次のように置き換えます。

- <character\_set>: データベース・セッションで使用するキャラクタ・セット。
- <user\_name>: Oracle Workflow データベース・アカウントのユーザー名を指定します。
- <password>: Oracle Workflow データベース・アカウントのパスワードを指定します。



- **<connect\_string>**: データベースの接続文字列。このコマンドはデフォルトの JDBC OCI8 ドライバを使用するため、接続文字列には、TNSNAMES エントリに指定されたデータベース名を指定する必要があります。

UNIX では、次のコマンドを使用して Java 関数アクティビティ・エージェントを起動することもできます（使用する JDBC ドライバのタイプを指定する場合）。

```
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>]
```

このコマンドでは、-DCHARSET オプションを使用して、使用するキャラクタ・セットを指定することもできます。キャラクタ・セットを指定しない場合は、デフォルトで UTF8 が使用されます。

コマンドのパラメータを次のように置き換えます。

- **<character\_set>**: データベース・セッションで使用するキャラクタ・セット。
- **<user\_name>**: Oracle Workflow データベース・アカウントのユーザー名を指定します。
- **<password>**: Oracle Workflow データベース・アカウントのパスワードを指定します。
- **<connect\_string>**: データベースの接続文字列。接続文字列の形式は、JDBC ドライバのタイプに応じて変わります。
  - JDBC OCI8 ドライバの場合は、TNSNAMES エントリに指定されたデータベース名を次の形式で指定する必要があります。  
**<database\_name>**
  - JDBC Thin ドライバの場合は、2 種類の接続文字列を使用できます。一方の接続文字列では、ホスト名、ポート番号およびデータベース・システム識別子 (SID) を次の形式で指定する必要があります。  
**<host\_name>:<port\_number>:<database\_SID>**

もう一方の接続文字列では、ホスト名、プロトコル、ポート番号および SID で構成される Oracle Net の名前 - 値ペアを、次の形式で指定する必要があります。  
 (description=(address=(host=<host\_name>) (protocol=

```
<protocol>)) (port=<port_number>)) (connect_data=(sid=
<database_SID>)))
```

- **<JDBC\_driver>**: データベース接続に使用する JDBC ドライバのタイプ。JDBC ドライバのタイプには、oci8 または thin を指定できます。キャラクタ・セットを指定しない場合は、デフォルトで JDBC OCI8 ドライバが使用されます。

### Windows NT での Java 関数アクティビティ・エージェントの起動

Windows NT では、次のコマンドを使用して Java 関数アクティビティ・エージェントを起動できます（デフォルトの JDBC OCI8 ドライバを使用する場合）。

```
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>
```

このコマンドでは、-DCHARSET オプションを使用して、使用するキャラクタ・セットを指定することもできます。キャラクタ・セットを指定しない場合は、デフォルトで UTF8 が使用されます。

コマンドのパラメータを次のように置き換えます。

- **<character\_set>**: データベース・セッションで使用するキャラクタ・セット。
- **<user\_name>**: Oracle Workflow データベース・アカウントのユーザー名を指定します。
- **<password>**: Oracle Workflow データベース・アカウントのパスワードを指定します。
- **<connect\_string>**: データベースの接続文字列。このコマンドはデフォルトの JDBC OCI8 ドライバを使用するため、接続文字列には、TNSNAMES エントリに指定されたデータベース名を指定する必要があります。

Windows NT では、次のコマンドを使用して Java 関数アクティビティ・エージェントを起動することもできます（使用する JDBC ドライバのタイプを指定する場合）。

```
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>]

```

このコマンドでは、-DCHARSET オプションを使用して、使用するキャラクタ・セットを指定することもできます。キャラクタ・セットを指定しない場合は、デフォルトで UTF8 が使用されます。

コマンドのパラメータを次のように置き換えます。

- <character\_set>: データベース・セッションで使用するキャラクタ・セット。
- <user\_name>: Oracle Workflow データベース・アカウントのユーザー名を指定します。
- <password>: Oracle Workflow データベース・アカウントのパスワードを指定します。
- <connect\_string>: データベースの接続文字列。接続文字列の形式は、JDBC ドライバのタイプに応じて変わります。
  - JDBC OCI8 ドライバの場合は、TNSNAMES エントリに指定されたデータベース名を次の形式で指定する必要があります。  
   <database\_name>
  - JDBC Thin ドライバの場合は、2 種類の接続文字列を使用できます。一方の接続文字列では、ホスト名、ポート番号およびデータベース・システム識別子 (SID) を次の形式で指定する必要があります。  
   <host\_name>:<port\_number>:<database\_SID>  
   もう一方の接続文字列では、ホスト名、プロトコル、ポート番号および SID で構成される Oracle Net の名前 - 値ペアを、次の形式で指定する必要があります。  
   (description=(address=(host=<host\_name>)(protocol=<protocol>)(port=<port\_number>))(connect\_data=(sid=<database\_SID>)))
- <JDBC\_driver>: データベース接続に使用する JDBC ドライバのタイプ。JDBC ドライバのタイプには、oci8 または thin を指定できます。キャラクタ・セットを指定しない場合は、デフォルトで JDBC OCI8 ドライバが使用されます。

## Java 関数アクティビティ・エージェントの停止

通常、Java 関数アクティビティ・エージェントは、永続的に動作します。ただし、wfjvstop.sql というスクリプトを実行すれば、このエージェントを停止することができます。このスクリプトは、Oracle Workflow Server の admin/sql サブディレクトリに入っ

ています。このスクリプトを実行すると、停止メッセージが送信キューに入ります。16-12 ページの「[wfjvstop.sql](#)」を参照してください。

---

**注意：** 複数の Java 関数アクティビティ・エージェントを同時に実行している場合は、Java 関数アクティビティ・エージェントごとに [wfjvstop.sql](#) スクリプトを実行する必要があります。

---

## 手順 WF-14 ビジネス・イベント・システムの設定

ビジネス・イベント・システムは、Oracle Workflow とともに提供されるアプリケーション・サービスの 1 つで、Oracle Advanced Queuing (AQ) を使用してシステム間でビジネス・イベントを伝達します。イベント処理を行う場合は、この手順を実行する必要があります。8-241 ページの「[Oracle Workflow ビジネス・イベント・システムの概要](#)」および 13-2 ページの「[ビジネス・イベントの管理](#)」を参照してください。

ビジネス・イベント・システムを設定するには、次の手順で行います。

1. ローカル・システムと外部システムとの間でビジネス・イベントを伝達する場合は、外部システムへのデータベース・リンクを作成します。
2. イベントを伝播するときにカスタム・キューを使用する場合は、専用のキューを設定します。
3. リスナーが標準の WF\_ERROR キューを監視するようにスケジューリングして、ビジネス・イベント・システムのエラー処理を有効にします。2-91 ページの「[キューの設定](#)」および 13-62 ページの「[ローカル・インバウンド・エージェントのリスナーのスケジューリング](#)」を参照してください。

データベース・リンクの作成とキューの設定は、手動で行う以外に、Oracle Enterprise Manager の Oracle DBA Studio を使用して行うこともできます。ワークフロー管理者は、Oracle DBA Studio を使用すれば、SQL DDL コマンドの知識がなくても、データベース・リンク、キュー表、キューおよびキューの伝播をすばやく簡単に作成および管理できます。『Oracle Enterprise Manager 管理者ガイド』を参照してください。

**コンテキスト：** この手順を実行する必要があるのは、1 度のみです。

### データベース・リンクの作成

システム間でイベント・メッセージを伝播するには、ローカル・システムからリモート・システムへのデータベース・リンクを作成する必要があります。ドメイン名を使用して、完全修飾されたデータベース・リンク名を指定する必要があります。

次の構文を使用して、データベース・リンクを作成できます。

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

たとえば、次のようになります。

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

ローカル・データベースおよびリモート・データベースに複数の Oracle Workflow をインストールしている場合、同じユーザー名とパスワードを使用して 2 つのシステムにアクセスするときは、`<user> IDENTIFIED BY <password>` 句を省略できます。この場合、データベース・リンクには、データベースに接続したユーザーのユーザー名とパスワードが使用されます。

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

すべてのユーザーが使用できるパブリック・データベース・リンクを作成する場合は、パラメータ PUBLIC を指定します。

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

データベース・リンクの名前を確認するには、次の構文を使用します。

```
SELECT db_link FROM all_db_links
```

「セットアップのチェック」Web 画面を使用して、作成したデータベース・リンクが設定されているかどうかを確認することもできます。13-57 ページの「[ビジネス・イベント・システムのセットアップのチェック](#)」を参照してください。

#### 関連項目：

『Oracle9i SQL リファレンス』の第 12 章「CREATE DATABASE LINK」

## キューの設定

ビジネス・イベント・システムでは、Oracle Advanced Queuing (AQ) を使用してシステム間でイベント・メッセージを伝達します。このため、イベント・マネージャで定義した各エージェントにキューを関連付ける必要があります。

Oracle Workflow をインストールすると、4 つの標準 Workflow エージェントに対して標準キューがそれぞれ自動的に作成されます。これらのキューのペイロード・タイプには、標準の WF\_EVENT\_T 構造が使用されます。13-27 ページの「[標準エージェント](#)」および 8-248 ページの「[イベント・メッセージ構造](#)」を参照してください。

次の表に、標準キューを示します。

表 2-2

キュー表	キュー名	説明
WF_IN	WF_IN	デフォルトの受信キュー
WF_OUT	WF_OUT	デフォルトの送信キュー
WF_DEFERRED	WF_DEFERRED	遅延サブスクリプション処理用のデフォルト・キュー
WF_ERROR	WF_ERROR	エラー処理用のデフォルト・キュー

標準ワークフロー・キューに取り込まれたメッセージは、デフォルトで7日間保持されます。この保持時間は、PL/SQL プロシージャ `DBMS_AQADM.Alter_Queue` を使用して、必要に応じて変更できます。これらのキューの設定は、保持時間は変更しないでください。

ただし、WF\_DEFERRED および WF\_ERROR のリスナーをスケジュールして、ビジネス・イベント・システムの遅延サブスクリプション処理およびエラー処理を有効にする必要があります。また、イベント・メッセージを伝播するときに WF\_IN および WF\_OUT を使用する場合は、WF\_IN のリスナーと WF\_OUT の伝播をスケジュールします。13-62 ページの「ローカル・インバウンド・エージェントのリスナーのスケジュール」および 13-67 ページの「ローカル・アウトバウンド・エージェントの伝播のスケジュール」を参照してください。

イベント・メッセージの伝播用に専用のキューを設定することもできます。キューを設定するには、キュー表およびキューを作成してから、キューを起動する必要があります。

- キュー表を作成するには、PL/SQL プロシージャの `DBMS_AQADM.Create_Queue_Table` を使用します。構文は、次のとおりです。

```
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');
```

標準の Workflow 形式をキューに使用する場合は、キューのペイロード・タイプを WF\_EVENT\_T に指定します。これらのキューでは、Oracle Workflow が提供する標準のキュー・ハンドラ WF\_EVENT\_QH が使用されます。別のペイロード・タイプを使用してキューを定義する場合は、キュー・ハンドラを作成して、標準の Workflow 形式からキューが要求する形式に変換する必要があります。7-21 ページの「キュー・ハンドラの標準 API」を参照してください。

storage\_clause パラメータを使用して、キュー表を作成する表領域を指定できます。大きな容量のキューが必要な場合は、表領域を指定することをお勧めします。

- キューを作成するには、PL/SQL プロシージャの DBMS\_AQADM.Create\_Queue を使用します。構文は、次のとおりです。

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

---

**注意：** 作成したキューに対して、他のデータベース・ユーザーがメッセージをエンキューまたはデキューする場合は、PL/SQL プロシージャの DBMS\_AQADM.Grant\_Queue\_Privilege を使用して、それらのユーザーに適切な権限を付与する必要があります。

---

- キューを起動するには、PL/SQL プロシージャの DBMS\_AQADM.Start\_Queue を使用します。構文は、次のとおりです。

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

Oracle Workflow には、wfevquec.sql および wfevqued.sql というサンプル・スクリプトが用意されています。wfevquec.sql スクリプトを変更してキューを設定したり、wfevqued.sql スクリプトを変更してキューを削除できます。これらのスクリプトは、Oracle Workflow のスタンドアロン版の場合はサーバー上の Oracle Workflow の sql サブディレクトリに、Oracle Applications embedded Workflow の場合は \$FND\_TOP の sql サブディレクトリに格納されています。

「セットアップのチェック」Web 画面を使用して、作成したキューが正しく設定されているかどうかを確認することができます。13-56 ページの「[ビジネス・イベント・システムのセットアップのチェック](#)」を参照してください。

---

**注意：** SQL\*Plus リリース 8.1.6 では、キュー表から USER\_DATA 列を選択することはできません。Workflow キューからイベント・メッセージのペイロードを選択する場合は、USER\_DATA を選択できる SQL\*Plus リリース 8.1.7 以降を使用する必要があります。

---

#### 関連項目：

『Oracle9i アプリケーション開発者ガイド - アドバンスト・キューイング』の「管理インターフェイス」

『PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の「DBMS\_AQADM」

## 手順 WF-15 WF\_EVENT\_OMB\_QH キュー・ハンドラの設定

Oracle8i でビジネス・イベント・システムを使用しているときに、Oracle Message Broker (OMB) を使ってシステム間でイベント・メッセージを伝播する場合は、この手順を実行する必要があります。WF\_EVENT\_OMB\_QH キュー・ハンドラは、標準の Workflow イベントのメッセージ構造 WF\_EVENT\_T を、OMB キューが要求する OMBAQ\_TEXT\_MSG 構造に変換します。

WF\_EVENT\_OMB\_QH の設定が完了したら、ビジネス・イベント・システムのエージェントにこのキュー・ハンドラを割り当てます。エージェントでは、OMB を使用して実装した伝播プロトコルが使用されます。13-24 ページの「[エージェント](#)」を参照してください。

---

**注意：** Oracle9i を使用している場合は、この手順を実行する必要はありません。Oracle9i では、Oracle Message Broker のかわりに、Oracle Advanced Queuing のメッセージ・ゲートウェイおよびインターネット・アクセス機能を使用して、イベント・メッセージを伝播できます。

---

**コンテキスト：** この手順を実行する必要があるのは、1 度のみです。

### WF\_EVENT\_OMB\_QH の設定

1. OMB を使用して、イベント・メッセージの伝播に使用する AQ キューを作成します。AQ キューは、Oracle Workflow スキーマで作成したシングル・コンシューマ・キューでなければなりません。受信キューと送信キューをそれぞれ 1 つ以上作成する必要があります。たとえば、WF\_OMB\_IN および WF\_OMB\_OUT というキューを作成します。
2. スクリプト wfquhndos.pls を実行して、WF\_EVENT\_OMB\_QH パッケージの PL/SQL 仕様を作成します。このスクリプトは、Oracle ホームの wf/sql サブディレクトリに格納されています。
3. スクリプト wfquhndob.pls を実行して、WF\_EVENT\_OMB\_QH パッケージの PL/SQL 本体を作成します。このスクリプトは、Oracle ホームの wf/sql サブディレクトリに格納されています。

#### 関連項目：

7-21 ページ [「キュー・ハンドラの標準 API」](#)

8-248 ページ [「イベント・メッセージ構造」](#)

8-264 ページ [「WF\\_EVENT\\_T および OMBAQ\\_TEXT\\_MSG 間のマッピング」](#)



## Oracle Workflow のアクセス保護の概要

アクセス保護は、シード・データの提供者によって作成されたワークフロー・シード・データが、シード・データの利用者によって変更されないようにする機能です。シード・データの提供者とは、他の組織（シード・データの利用者）がワークフロー・プロセスの定義やカスタマイズに使用するシード・データを作成する組織を指します。Oracle Workflow では、シード・データは次のどちらかを指します。

- 特定の使用者のニーズにあわせてカスタマイズされるワークフロー・オブジェクト定義
- 標準であり、提供者によって将来アップグレードされる可能性があるため、カスタマイズされないように保護するワークフロー・オブジェクト定義

たとえば、Oracle Workflow 開発チームは、標準項目タイプというシード・データの提供者です。標準項目タイプには、カスタム・ワークフロー・プロセスに組み込むことができる標準アクティビティが含まれます。組織の本社にある開発チームが、標準項目タイプのアクティビティを参照する、カスタム・ワークフロー・プロセスの定義を作成することがあります。この場合は、本社のチームが標準項目タイプのシード・データの利用者となります。

本社のチームが、作成したカスタム・ワークフロー定義を他の地域のオフィスのチームに展開するとします。本社のチームは、シード・データの提供者として次のように規定する可能性があります。

- カスタム・ワークフロー定義内の特定のワークフロー・オブジェクトを社内標準として指定し、地方のチームはこれを遵守し、変更できないようにします。
- 展開されたプロセス内のあるオブジェクトをカスタマイズ可能なオブジェクトとして指定し、地方のオフィスがそのニーズにあわせて変更できるようにします。

本社チームは、Oracle Workflow のアクセス保護機能を使用して、両方の要件を満たすことができます。アクセス保護を使用すると、シード・データの提供者は、あるデータを読み取り専用として保護し、他のデータのカスタマイズを許可できます。また、シード・データのアップグレード中にアクセス保護を使用すると、シード・データ提供者は保護されている既存のシード・データを新バージョンで上書きし、一方でカスタマイズ可能なシード・データに加えられたカスタマイズは保存できます。

Oracle Workflow はデータベースに格納されるすべてのワークフロー・オブジェクト定義に保護とカスタマイズのレベルを割り当て、Oracle Workflow の各ユーザーに特定のアクセス・レベルで操作するように要求します。アクセス保護機能は保護、カスタマイズおよびアクセスの各レベルの組合せで構成され、ユーザーが特定のワークフロー・オブジェクトを変更できるかどうかを決定します。3つのすべての場合に、レベルは0～1000の数値で、シード・データの提供者と利用者としての異なる組織間の関連を示します。

Oracle Workflow では、次の範囲のレベルを想定しています。

0-9	Oracle Workflow
10-19	Oracle Application Object Library
20-99	Oracle Applications の開発
100-999	顧客の組織。この範囲の解釈方法は、ユーザーが定義できます。たとえば、100 が本社を示し、101 が地方のオフィスを示す、などです。
1000	制限なし

### アクセス・レベル

Oracle Workflow の各ユーザーは、前述のレベルの範囲に従って特定のアクセス・レベルでシステムを操作します。この場合、Oracle Workflow のユーザーとは、Oracle Workflow Builder またはファイルからデータベースへワークフロー・プロセス定義をロードするワークフロー定義ローダーのプログラムを操作するユーザーを示します。作業時のレベルは作成するシード・データの保護レベルに影響するため、シード・データ提供者は、常に一貫して同じアクセス・レベルで Oracle Workflow Builder を操作する必要があります。

アクセス・レベルを表示するには、次の方法があります。

- Oracle Workflow Builder で、「ヘルプ」メニューから「Oracle Workflow Builder のバージョン情報」を選択します。
- ワークフロー定義ローダーのプログラムを実行して、ワークフロー・プロセス定義をデータベースからファイルにダウンロードする場合は、ワークフロー・サーバー上の環境変数 WF\_ACCESS\_LEVEL の値をチェックします。2-101 ページの「[ワークフロー定義ローダーの使用](#)」を参照してください。

---

---

**注意：** ワークフロー定義ローダーのプログラムでは、環境変数 WF\_ACCESS\_LEVEL に格納されているアクセス・レベルが参照されます。この環境変数は、Oracle Workflow をサーバーにインストールするときに定義する必要があります。この環境変数を定義しなければ、ワークフロー定義ローダーでは、デフォルトのアクセス・レベルである 100 が想定されます。

---

---

---

**注意：** Oracle Applications embedded Workflow をインストールする場合は、この変数を環境ファイル内で定義する必要があります。デフォルトの環境ファイルは APPLSYS.env です。この環境変数を定義しなければ、ワークフロー定義ローダーではデフォルトのアクセス・レベルである 100 が想定されます。環境ファイルの詳細は、該当する製品の Oracle Applications のインストール・マニュアルを参照してください。

---

## 保護レベル

Oracle Workflow Builder でワークフロー・オブジェクトを作成するときに、そのオブジェクトを特定のレベルで保護するかどうかを選択できます。オブジェクトの保護レベルによって、そのオブジェクトを他のユーザーが各自のアクセス・レベルに基づいて変更できるかどうか管理されます。

オブジェクトの保護レベルを変更するには、そのオブジェクトのプロパティ画面の「アクセス」タブを表示します。オブジェクトに対して設定する保護レベルは、現行のアクセス・レベルによって異なります。オブジェクトへのアクセスを管理するには、次の 4 通りの方法があります。

- **全員にアクセスを許可：** デフォルトでは、「アクセス」タブで「カスタマイズを保持」と「このアクセス・レベルでロック」がオフになっていると、保護レベルが 1000 のオブジェクトには誰でもアクセスできます。
- **アクセス・レベルが自分と同じかそれ以上のユーザーにアクセスを制限：** 「アクセス」タブの「オプション」領域で「カスタマイズを保持」をオンにして、自分の現行のアクセス・レベル以上であれば誰でもそのオブジェクトをカスタマイズできるように指定します。カスタマイズ可能のマークを付けるのは、将来そのオブジェクトのバージョンをアップグレードする場合に、他のユーザーによるカスタマイズを上書きしないことが確実な場合のみにしてください。
- **自分と同じかそれ以下のアクセス・レベルのユーザーにアクセスを制限：** 「このアクセス・レベルでロック」をオンにすると、そのオブジェクトは保護され、変更できるのは自分の現行のアクセス・レベル以下のユーザーのみにになります。上位のアクセス・レベルで操作しているユーザーには、ワークフロー・オブジェクトのアイコンに小さなロック・アイコンが表示され、そのオブジェクトの使用は可能でも変更はできないことを示します。グローバル・アップグレードを使用しない限り変更されない、標準コンポーネントとして定義するオブジェクトを保護してください。このため、常に一貫して同じアクセス・レベルで操作することが重要になります。
- **自分と同じアクセス・レベルのユーザーにアクセスを制限：** 「このアクセス・レベルでロック」と「カスタマイズを保持」の両方をオンにすると、そのオブジェクトを変更できるのは、自分の現行アクセス・レベルと同じレベルで操作するユーザーのみにになります。

次の表に、オブジェクトへのアクセス・レベルを「カスタマイズを保持」および「このアクセス・レベルでロック」オプションの設定別に示します。

表 2-3

カスタマイズを保持	このアクセス・レベルで ロック	オブジェクトに適用される アクセス・レベル
オフ	オフ	オブジェクトはどのアクセス・レベルでも更新できます。
オン	オフ	オブジェクトは、現行のアクセス・レベル以上のアクセス・レベルでのみ更新できます。
オフ	オン	オブジェクトは、現行のアクセス・レベル以下のアクセス・レベルでのみ更新できます。
オン	オン	オブジェクトは、現行のアクセス・レベルと同じアクセス・レベルでのみ更新できます。

**注意：** Microsoft 社の Internet Explorer のベータ版を PC にインストールしている場合は、ファイル comctl32.dll の旧バージョンが自動的にインストールされるため、Oracle Workflow Builder でロック・オブジェクト上にロック・アイコンが表示されないことがあります。この問題を修正するには、Microsoft 社の Internet Explorer の製品版をインストールして、comctl32.dll を最新版に交換してください。

Oracle Workflow の保護レベルとアクセス・レベルは、特定のワークフロー・オブジェクトの変更に関して、それが不可能であるか、または許可されたアクセス・レベルでこのツールにアクセスするユーザーのみが可能であることを示します。ワークフロー・オブジェクトの安全性確保やソース管理のための手段ではありません。

**注意：** Oracle Workflow のほとんどのワークフロー・オブジェクトは、保護レベルが 0 に設定されています。これは、そのオブジェクトが、アクセス・レベル 0 で操作する Oracle Workflow チームによってのみ、変更可能であることを示します。自分のアクセス・レベルを 0 に変更してデータを変更しようとする、特に、Oracle Workflow でシード・データがアップグレードされたために、もともと保護されていたデータに対して行う変更が上書きされる可能性がある場合は、カスタマイズがサポートされなくなります。

カスタマイズ・レベル

オブジェクトを変更してデータベースまたはファイルに保存すると、保護レベルのワークフロー・オブジェクトのみでなく、すべてのワークフロー・オブジェクトに、自分のアクセス・レベルと同じカスタマイズ・レベルが記録されます。たとえば、ワークフロー・オブジェクトがカスタマイズ可能（保護レベル 1000）な場合に、それをアクセス・レベル 100 でカスタマイズすると、オブジェクトのカスタマイズ・レベルは 100 に設定されます。カスタ

マイズ・レベルは、そのオブジェクトをさらに変更できるのは、そのレベルと同じかまたはそれ以上のアクセス・レベルのユーザーに限られることを示します。したがって、この例では、アクセス・レベルが 100 以上のユーザーのみが、そのオブジェクトをカスタマイズできます。オブジェクトのカスタマイズ・レベルより低いアクセス・レベルで操作している場合は、そのワークフロー・オブジェクトのアイコン上に小型のロックが表示され、そのオブジェクトの使用は可能でも変更はできないことを示します。

これにより、シード・データのアップグレード中に、すでにカスタマイズされているカスタマイズ可能オブジェクトが上書きされるのを防ぐことができます。これは、アップグレード時には、カスタマイズされたオブジェクトのカスタマイズ・レベルよりも低いアクセス・レベルで処理を行うワークフロー定義ローダーが常に使用されるためです。

## デフォルトのアクセス・レベルの設定

Oracle Workflow Builder を Microsoft Windows 98、Windows 2000 または Windows NT の PC にインストールする場合は、Oracle Universal Installer により、インストール先の PC とオペレーティング・システムに対してグローバルなデフォルトのアクセス・レベルが割り当てられます。Oracle Workflow Builder のインストール後は、PC 上の個々のユーザーに各自のアクセス・レベルを新しい設定に変更させることができます。これにより、その PC に対して設定されたデフォルトのアクセス・レベルが上書きされます。ユーザーがアクセス・レベルを定義しなければ、Oracle Workflow Builder では PC のデフォルトのアクセス・レベルの値が使用されます。このアクセス・レベルは、Microsoft Windows のレジストリに保存されます。

Oracle Workflow Builder とワークフロー・シード・データを組織の他の部署のユーザーに展開し、提供したシード・データをそれらのユーザーが変更しないようにするには、Oracle Workflow Builder でそのデータの保護レベルよりも高いアクセス・レベルで処理を行うようにできます。たとえば、シード・データ提供者としてアクセス・レベル 100 で処理を行い、作成したシード・データがレベル 100 で保護されている場合、ユーザーまたはシード・データ使用者のデフォルトのアクセス・レベルは 101 以上に設定する必要があります。

Oracle Workflow Builder でユーザーのアクセス・レベルを設定するには、「ヘルプ」メニューから「Oracle Workflow Builder のバージョン情報」を選択するように指示します。「Oracle Workflow Builder のバージョン情報」ウィンドウで、「アクセス・レベル」フィールドを自分のシード・データ保護レベルより大きい数値に変更し、「OK」を選択します。

regedit などのレジストリ・エディタで、

`HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\WorkflowLevel`

の 10 進値を編集することにより、Microsoft Windows のレジストリでアクセス・レベルを直接設定することもできます。

ワークフロー定義ローダーのプログラムの場合は、環境変数 `WF_ACCESS_LEVEL` を定義し、適切なオペレーティング・システム・コマンドで値を設定して、プログラムがプロセス定義をファイルにダウンロードするときに動作するデフォルトのアクセス・レベルを設定します。

---

---

**注意：** 自分のアクセス・レベルを変更することもできますが、Oracle Workflow では、もともと 99 以下のレベルで保護されていたシード・データのカスタマイズはサポートされません。自分のアクセス・レベルを許可されていないレベルに変更して、保護されているデータを変更することは、絶対にしないでください。

---

---

## ワークフロー定義ローダーの使用

Oracle Workflow Builder で「ファイル」メニューの「保存」または「開く」オプションを使用するかわりに、ワークフロー定義ローダーのプログラムを使用してデータベースやフラット・ファイルからプロセス定義をロードしたり保存できます。

データベースをアップグレードする場合は、ワークフロー定義ローダーを使用してユーザーのプロセス定義をフラット・ファイルに保存し、バックアップを作成します。データベースのアップグレード完了後に、再度ローダー・プログラムを使用して、定義をデータベースにアップロードして戻します。ローダー・プログラムを使用して、新バージョンのプロセス定義でデータベースをアップグレードしたり、プロセス定義を他のデータベースに移動できます。

プロセス定義をアップロードまたはアップグレードすると、ワークフロー定義ローダーにより自動的にプロセス定義が検証され、特定のプロセス設計ルールに従っていることが確認されます。ここでは、Oracle Workflow Builder の検証機能と同じ検証が行われます。5-21 ページの「[プロセス定義の検証](#)」を参照してください。

---

**注意：** データベースの既存の定義上にワークフロー定義をアップロードまたはアップグレードすると、アップロード / アップグレード定義のオブジェクトの表示名が、ターゲット・データベース内の異なるオブジェクトにすでに使用されている場合があります。この場合、ワークフロー定義ローダーでは、ターゲット・データベース内で競合している表示名の先頭にアットマーク (@) が追加され、この表示名の競合が自動的に解決されます。その後、アップロード / アップグレード定義がそのまま適用され、警告メッセージが生成されます。

---

---

**注意：** ワークフロー定義ローダーのリリース 2.6.1 を使用すると、Oracle Applications embedded Workflow リリース 11i の全リリースだけでなく、Oracle Workflow スタンドアロン版のリリース 2.6.1、リリース 2.6、リリース 2.5 から、プロセス定義をアップロードおよびダウンロードできます。ただし、ワークフロー定義ローダーを使用して以前のリリースの Oracle Workflow Server にプロセス定義をアップロードすると、それ以降のリリースで導入された新機能をそれらのプロセスに含めることはできません。以前のリリースで使用できない機能の詳細は、3-22 ページの「[各種リリースのサーバーでの Oracle Workflow Builder の使用](#)」を参照してください。

---

➤ **Oracle Workflow スタンドアロン版のワークフロー定義ローダーの実行**

1. ワークフロー定義ローダーは、サーバー上の Oracle ホーム・ディレクトリ構造の bin サブディレクトリにあります。
2. このプログラムをオペレーティング・システム・プロンプトから次のように実行します (<username/password@database> をユーザー名、パスワードおよびデータベースへの Oracle Net 接続文字列または別名に置き換えます)。

- 入力ファイルからデータベースにシード・データのアップグレードを適用するには、次のように入力します。

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

ワークフロー定義ローダーは、デフォルトのアップグレード動作を使用して、ファイル作成者（シード・データ提供者）のアクセス・レベルを使用し、そのアップグレード・ファイルのアクセス・レベル以上のレベルで保護されているオブジェクトを上書きします。アップグレード時には、ローダー・プログラムにより、データベース内のカスタマイズ可能シード・データに対するカスタマイズも保存されます。<input\_file> では、ロードするアップグレード・ファイルのフルパス名を指定します。

- 入力ファイルからデータベースへプロセス定義をアップロードするには、次のように入力します。

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

アップロード・モードは、ワークフロー・プロセスの開発者に役立ちます。このモードでは、開発者は定義をデータベースに保存でき、既存のオブジェクトを不注意にカスタマイズしたために、プロセス定義の一部の要素をアップロードできなくなるかどうかを気にする必要がありません。ワークフロー定義ローダーでは、入力ファイルで指定されたアクセス・レベルが使用されます。<input\_file> では、アップロード元となる入力ファイルのフルパス名を指定します。

- オブジェクトの保護レベルに関係なく、プロセス定義を入力ファイルからデータベースへ強制的にアップロードするには、次のように入力します。

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

<input\_file> では、アップロード元となる入力ファイルのフルパス名を指定します。データベースに保存されているプロセス全体が上書きされるため、強制オプションを使用する場合は、あらかじめファイル内のプロセス定義が正しいことを確認する必要があります。強制オプションは、信頼できる既存ファイルのバックアップを使用して、データベースのデータの整合性に関する問題を修正する場合に役立ちます。また、Oracle Workflow Release 1.0 または 1.0.1 から、旧データ・モデルを反映する .wft ファイルをロードする場合にも役立ちます。



---

**注意：** 強制オプションを使用して Oracle Workflow リリース 1.0 または 1.0.1 からデータベースに .wft ファイルをロードする場合は、ロード後に手動の手順も完了する必要があります。ロードする選択肢タイプを、項目タイプに関連付ける必要もあります。そのためには、Oracle Workflow Builder の「ナビゲータ」ウィンドウで、選択肢タイプを独立した「選択肢タイプ」のブランチから項目タイプに関連した「選択肢タイプ」のブランチにドラッグします。

---

- データベースから出力ファイルに 1 つ以上の項目タイプのプロセス定義をダウンロードするには、次のように入力します。

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

<output\_file> では書き込み先となる出力ファイルのフルパス名を指定し、<item\_typeN> ではダウンロードする各項目タイプの内部名を指定します。また、<item\_typeN> を「\*」に置き換えると、すべての項目タイプを表すことができます（アスタリスクは必ず一重引用符で囲みます）。-d オプションを使用して日付を指定すると（カッコは除く）、その日付に有効だったプロセス定義をダウンロードできます。この日付は、書式 YYYY/MM/DD HH24:MI:SS を使用して指定します。

出力ファイルの拡張子は .wft である必要があります。プロセス定義をダウンロードすると、ローダー・プログラムにより、出力ファイルのアクセス・レベルが環境変数 WF\_ACCESS\_LEVEL に格納されている値に設定されます。

## ► Oracle Applications embedded Workflow のワークフロー定義ローダーの実行

1. Oracle Applications の「Submit Requests」フォームにナビゲートし、ワークフロー定義ローダーのコンカレント・プログラムを発行します。システム管理者は、Oracle Applications と Oracle Workflow をインストールして設定するときに、このコンカレント・プログラムを実行する職責の要求セキュリティ・グループに追加する必要があります。『Oracle Applications System Administrator's Guide』の「Overview of Concurrent Programs and Requests」を参照してください。
2. ワークフロー定義ローダーのコンカレント・プログラムを要求として発行します。『Oracle Applications User's Guide』の「Submitting a Request」を参照してください。
3. 「Parameters」ウィンドウで次のパラメータの値を入力します。

モード	<p>「Download」を指定すると、プロセス定義はデータベースからフラット・ファイルにダウンロードされます。</p> <p>「Upgrade」を指定すると、シード・データのアップグレードが入力ファイルからデータベースに適用されます。ワークフロー定義ローダーではファイル作成者（シード・データ提供者）のアクセス・レベルが使用されるため、そのアップグレード・ファイルのアクセス・レベル以上のレベルで保護されているオブジェクトが上書きされます。データベースのカスタマイズ可能なシード・データに加えられるカスタマイズも、すべて保存されます。</p> <p>「Upload」を指定すると、プロセス定義がフラット・ファイルからデータベースにロードされます。アップロード・モードは、ワークフロー・プロセスの開発者に役立ちます。このモードでは、開発者は定義をデータベースに保存でき、既存のオブジェクトを不注意にカスタマイズしたために、プロセス定義の一部の要素をアップロードできなくなるかどうかを気にする必要がありません。ワークフロー定義ローダーは、入力ファイルで定義されたアクセス・レベルを使用してファイルからプロセス定義をアップロードします。このとき、そのファイルのアクセス・レベル以上のレベルで保護されているデータベース内のオブジェクトが上書きされます。</p> <p>「Force」を指定すると、オブジェクトの保護レベルに関係なく、入力ファイルのプロセス定義がデータベースに強制的にアップロードされます。データベースに格納されているプロセス全体が上書きされるため、ファイル内のプロセス定義が正しいことを確認しておく必要があります。「Force」モードは、既存の信頼できるファイル・バックアップを基にして、データベース内のデータ整合性の問題を解決するときに使用します。</p>
ファイル	<p>プロセス定義のダウンロード先またはアップグレード元、アップロード元となるファイルのフルパス名を指定します。</p>
項目タイプ	<p>「Mode」を「Download」に設定した場合は、「List」ボタンを使用して、ダウンロードするプロセス定義の項目タイプを選択します。</p>

---

**注意：**「Submit Requests」フォームからワークフロー定義ローダーを発行してプロセス定義をファイルにダウンロードする場合は、項目タイプを一度に1つずつダウンロードするようにしか指定できません。同時に複数またはすべての項目タイプをダウンロードする場合は、ワークフロー定義ローダーのコンカレント・プログラムをコマンドラインから発行する必要があります。詳細は、後述の手順6を参照してください。

---

4. 「OK」を選択して「パラメータ」ウィンドウを閉じます。

5. この要求の印刷オプションと実行オプションを変更してから、「Submit」を選択して要求を発行します。
6. 「Submit Requests」フォームを使用せずに、次のコマンドを入力し、ワークフロー定義ローダーのコンカレント・プログラムをコマンドラインから実行することもできます。

アップグレードする場合:           WFLOAD apps/pwd 0 Y UPGRADE  
  file.wft

アップロードする場合:           WFLOAD apps/pwd 0 Y UPLOAD file.wft

強制的にアップロードする場合:   WFLOAD apps/pwd 0 Y FORCE file.wft

ダウンロードする場合:           WFLOAD apps/pwd 0 Y DOWNLOAD  
  file.wft ITEMTYPE1 [ITEMTYPE2  
  ...ITEMTYPEN]

apps/pwd を APPS スキーマのユーザー名とパスワードに置き換え、file.wft をワークフロー・プロセス定義ファイルのファイル仕様に置き換え、ITEMTYPE1、ITEMTYPE2、... ITEMTYPEN をダウンロードする 1 つ以上の項目タイプに置き換えます。また、ITEMTYPE1 を「\*」に置き換えると、すべての項目タイプを同時にダウンロードできます（アスタリスクは必ず一重引用符で囲みます）。

ファイル仕様は次のように指定します。

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

または

```
<native path>
```

## Workflow XML Loader の使用

Workflow XML Loader を使用すると、データベースとフラット・ファイルとの間でビジネス・イベント・システムのオブジェクトの XML 定義をアップロードおよびダウンロードできます。データベースからビジネス・イベント・システムのオブジェクト定義をダウンロードすると、Oracle Workflow によって XML ファイルとして保存されます。データベースにオブジェクト定義をアップロードすると、Oracle Workflow によってソース XML ファイルからデータベース内のビジネス・イベント・システム表にロードされます。必要に応じて、新しい定義が作成されたり、既存の定義が更新されたりします。

ビジネス・イベント・システムのオブジェクトの XML 定義は、次のドキュメント・タイプ定義 (Document Type Definition: DTD) に従って構成されます。

- イベント： 8-308 ページ「WF\_EVENTS DTD」
- イベント・グループ・メンバー： 8-311 ページ「WF\_EVENT\_GROUPS DTD」
- システム： 8-314 ページ「WF\_SYSTEMS DTD」
- エージェント： 8-317 ページ「WF\_AGENTS DTD」
- イベント・サブスクリプション： WF\_EVENT\_SUBSCRIPTIONS DTD (8-320 ページ)

ビジネス・イベント・システムのオブジェクト定義は、標準ダウンロード・モードまたは完全ダウンロード・モードでダウンロードできます。

- 標準ダウンロード・モードを使用した場合は、特定のシステムのオブジェクト定義から標準コピーが作成および保存されます。このコピーを使用すれば、類似した定義を他のシステムで作成できます。このモードでは、Workflow XML Loader によって、オブジェクト定義内の特定のシステム固有のデータがトークンに置き換えられます。たとえば、開発システムのビジネス・イベント・システムのオブジェクト定義を保存して、本番システムにアップロードできるシード・データとして使用するときに選択します。
- 完全ダウンロード・モードを使用すると、データベースに指定されているオブジェクト定義を完全に保存できます。このモードでは、Workflow XML Loader によってデータがトークンに変換されることはありません。すべての値 (システム固有の値を含む) が XML ファイルにコピーされます。たとえば、特定のシステムの本番システムのビジネス・イベント・システムのオブジェクト定義を保存して、そのシステムと対話する別のシステムにレプリケートするときに選択します。

標準ダウンロード・モードでは、ビジネス・イベント・システムのオブジェクト定義にあるシステム固有のデータが次のトークンに置き換えられます。トークンの先頭には、# が付いています。

- **#NEW:** エージェント定義内のエージェント、またはサブスクリプション定義内のイベント・サブスクリプションのグローバル一意識別子を置き換えます。
- **#LOCAL:** エージェント定義またはサブスクリプション定義内の任意の場所にある、ローカル・システムのグローバル一意識別子を置き換えます。

- **#OWNER:** キューを所有するスキーマがエージェント定義内のキュー名およびエージェント・アドレスの一部として表示されるときに、スキーマの名前を置き換えます。
- **#SID:** データベース・システム識別子 (SID) がエージェント定義内のエージェント・アドレスの一部として表示されるときに、そのデータベース・システム識別子を置き換えます。
- **#WF\_IN:** ローカル・システム上の WF\_IN エージェントのグローバル意識別子がイベント・サブスクリプション定義内のソース・エージェント、送信エージェントまたは宛先エージェントとして表示されるときに、そのグローバル意識別子を置き換えます。
- **#WF\_OUT:** ローカル・システム上の WF\_OUT エージェントのグローバル意識別子がイベント・サブスクリプション定義内のソース・エージェント、送信エージェントまたは宛先エージェントとして表示されるときに、そのグローバル意識別子を置き換えます。
- **#WF\_ERROR:** ローカル・システム上の #WF\_ERROR エージェントのグローバル意識別子がイベント・サブスクリプション定義内のソース・エージェント、送信エージェントまたは宛先エージェントとして表示されるときに、そのグローバル意識別子を置き換えます。

ローダーでは、これらのシステム固有の値をトークンに変換することにより、テンプレート定義を作成します。このテンプレート定義を使用して、類似したオブジェクトを他のシステムで作成できます。トークンが含まれているオブジェクト定義をデータベースにアップロードすると、Oracle Workflow によってそれらのトークンがシステムに適した値に置き換えられます。

#### 関連項目：

13-2 ページ「[ビジネス・イベントの管理](#)」

#### ► データベースからのビジネス・イベント・システムの XML 定義のダウンロード

データベース上のビジネス・イベント・システムのオブジェクト定義を XML フラット・ファイルにダウンロードするときは、Workflow XML Loader を手動で実行する以外に、スクリプトを使用してローダーを実行する (Oracle Workflow のスタンドアロン版を使用している場合) こともできます。

Workflow XML Loader を手動で実行するには、`oracle.apps.fnd.wf.WFXLoad` に対して JRE を実行します。Java Runtime Environment、Workflow JAR ファイルが入っているディレクトリ、Oracle JDBC 実装および次の Workflow JAR ファイルを指している CLASSPATH を指定する必要があります。

- `wfjava.jar`: Workflow Java ユーティリティ
- `wfapi.jar`: Workflow Java API

---

**注意：** Oracle9i で Oracle Workflow スタンドアロン版を使用している場合、Workflow JAR ファイルは <ORACLE\_HOME>/jlib ディレクトリに格納されています。Oracle Applications embedded Workflow を使用している場合、Workflow JAR ファイルは <ORACLE\_HOME>/wf/java/oracle/apps/fnd/wf/jar/ ディレクトリに格納されています。

---

たとえば、UNIX では、次のコマンドを使用して Workflow XML Loader を実行します。

```
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>
```

Windows NT では、次のコマンドを使用します。

```
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>
```

Oracle Workflow のスタンドアロン版を使用している場合は、wfxload (UNIX の場合) または wfxload.bat (Windows NT の場合) というサンプル・スクリプトを使用して、Workflow XML Loader を実行できます。これらのスクリプトは、サーバーの Oracle Workflow の admin サブディレクトリに格納されています。たとえば、UNIX では次のコマンドを使用します。

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

Windows NT では、次のコマンドを使用します。

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

Workflow XML Loader を実行するときは、-d オプションまたは -de オプションを使用して、適切なダウンロード・モードを指定します。

- -d: 標準ダウンロード・モード。必要に応じて、オブジェクト定義内のシステム固有のデータが、先頭に # が付いたトークンに変換されます。

- `-de`: 完全ダウンロード・モード。オブジェクト定義が完全にコピーされます。データはトークンに変換されません。

また、ダウンロード・コマンド内の次の変数がパラメータに置き換えられます。

- `<user>`: データベース・アカウントのユーザー名。
- `<password>`: データベース・アカウントのパスワード。
- `<connect_string>`: データベースの接続文字列。接続文字列の形式は、JDBC ドライバのタイプに応じて変わります。
  - JDBC OCI8 ドライバの場合は、TNSNAMES エントリに指定されたデータベース名を次の形式で指定する必要があります。  
`<database_name>`
  - JDBC Thin ドライバの場合は、2 種類の接続文字列を使用できます。一方の接続文字列では、ホスト名、ポート番号およびデータベース・システム識別子 (SID) を次の形式で指定する必要があります。  
`<host_name>:<port_number>:<database_SID>`  
 もう一方の接続文字列では、ホスト名、プロトコル、ポート番号および SID で構成される Oracle Net の名前 - 値ペアを、次の形式で指定する必要があります。  
`(description=(address=(host=<host_name>)(protocol=<protocol>)(port=<port_number>))(connect_data=(sid=<database_SID>)))`
- `<protocol>`: データベース接続に使用する JDBC ドライバのタイプ。JDBC ドライバのタイプには、oci8 または thin を指定できます。
- `<lang>`: XML ファイルの言語の略称。このパラメータでは、大文字 / 小文字が区別されます。US (米語) や JA (日本語) など、Oracle データベース・サーバーの標準言語略称を使用してください。標準言語略称の一覧は、『Oracle9i グローバリゼーション・サポート・ガイド』の「ロケール・データ」を参照してください。
- `<output_file>`: 定義の保存先となる出力ファイルの名前およびフルパス。
- `<object>`: ダウンロードするオブジェクト定義のタイプ。
  - EVENT: イベント、イベント・グループ・メンバーおよびイベント・サブスクリプションの定義
  - SYSTEMS: システム定義
  - AGENTS: エージェント定義
  - ALL: イベント、イベント・グループ・メンバー、システム、エージェント、イベント・サブスクリプションなど、ビジネス・イベント・システムのすべてのオブジェクト定義

---

---

**注意：** Workflow XML Loader では、ローカル・システムに登録されているシステム、エージェントおよびイベント・サブスクリプションの定義だけをダウンロードします。

---

---

- **<key>:** ダウンロードする定義を制限するオプション・キー。キーを指定した場合は、内部名にそのキーが含まれるオブジェクト定義だけが取り出されます。このキー値では、大文字 / 小文字が区別されます。空白を含めることはできません。指定したタイプのオブジェクト定義をすべて取り出す場合は、このパラメータを省略できます。

---

---

**注意：** オブジェクト・タイプに「ALL」を指定した場合、このキーは無視され、ビジネス・イベント・システムのすべてのオブジェクト定義がダウンロードされます。

---

---

---

---

**注意：** Workflow XML Loader をダウンロード・モードで使用するには、リリース 8.1.7 以降のデータベースが必要です。それ以前のリリースの Oracle8i では、ダウンロード・ユーティリティがサポートされていません。以前のリリースでビジネス・イベント・システムのオブジェクトをシステム間でレプリケートするには、ビジネス・イベント・システムの定義済のサブスクリプションを使用して、システムを同期化する必要があります。13-77 ページの「[システムの同期](#)」を参照してください。

ただし、アップロード・モードで Workflow XML Loader を実行する場合は、リリース 8.1.6 以前の Oracle8i を使用できます。

---

---

## ➤ データベースへのビジネス・イベント・システムの XML 定義のアップロード

XML ファイルのビジネス・イベント・システムのオブジェクト定義をデータベースにアップロードする場合は、Workflow XML Loader を手動で実行する以外に、スクリプトを使用してローダーを実行する（Oracle Workflow のスタンドアロン版を使用している場合）こともできます。

Workflow XML Loader を手動で実行するには、`oracle.apps.fnd.wf.WFXLoad` に対して JRE を実行します。Java Runtime Environment、Workflow JAR ファイルが入っているディレクトリ、Oracle JDBC 実装および次の Workflow JAR ファイルを指している CLASSPATH を指定する必要があります。

- `wfjava.jar`: Workflow Java ユーティリティ
- `wfapi.jar`: Workflow Java API



---

**注意：** Oracle9i で Oracle Workflow スタンドアロン版を使用している場合、Workflow JAR ファイルは <ORACLE\_HOME>/jlib ディレクトリに格納されています。Oracle Applications embedded Workflow を使用している場合、Workflow JAR ファイルは <ORACLE\_HOME>/wf/java/oracle/apps/fnd/wf/jar/ ディレクトリに格納されています。

---

たとえば、UNIX では、次のコマンドを使用して Workflow XML Loader を実行します。

```
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>
```

Windows NT では、次のコマンドを使用します。

```
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>
```

Oracle Workflow のスタンドアロン版を使用している場合は、wfxload (UNIX の場合) または wfxload.bat (Windows NT の場合) というサンプル・スクリプトを使用して、Workflow XML Loader を実行できます。これらのスクリプトは、サーバーの Oracle Workflow の admin サブディレクトリに格納されています。たとえば、UNIX では次のコマンドを使用します。

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

Windows NT では、次のコマンドを使用します。

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

Workflow XML Loader を実行するときは、-u オプションを使用して、アップロード・モードでローダーを実行するように指定します。また、次の変数をパラメータに置き換えます。

- <user>: データベース・アカウントのユーザー名。
- <password>: データベース・アカウントのパスワード。

- `<connect_string>`: データベースの接続文字列。接続文字列の形式は、JDBC ドライバのタイプに応じて変わります。
  - JDBC OCI8 ドライバの場合は、TNSNAMES エントリに指定されたデータベース名を次の形式で指定する必要があります。  
`<database_name>`
  - JDBC Thin ドライバの場合は、ホスト名、ポート番号およびデータベース・システム識別子 (SID) を次の形式で指定する必要があります。  
`<host_name>:<port_number>:<database_SID>`
- `<protocol>`: データベース接続に使用する JDBC ドライバのタイプ。JDBC ドライバのタイプには、oci8 または thin を指定できます。
- `<lang>`: XML ファイルの言語の略称。このパラメータでは、大文字 / 小文字が区別されます。US (米語) や JA (日本語) など、Oracle データベース・サーバーの標準言語略称を使用してください。標準言語略称の一覧は、『Oracle9i グローバリゼーション・サポート・ガイド』の「ロケール・データ」を参照してください。
- `<source_file>`: 定義のアップロード元となるソース・ファイルの名前およびフルパス。

---

## ワークフロー・プロセス定義

この章では、Oracle Workflow Builder を使用してワークフロー・プロセスを定義する方法について説明します。

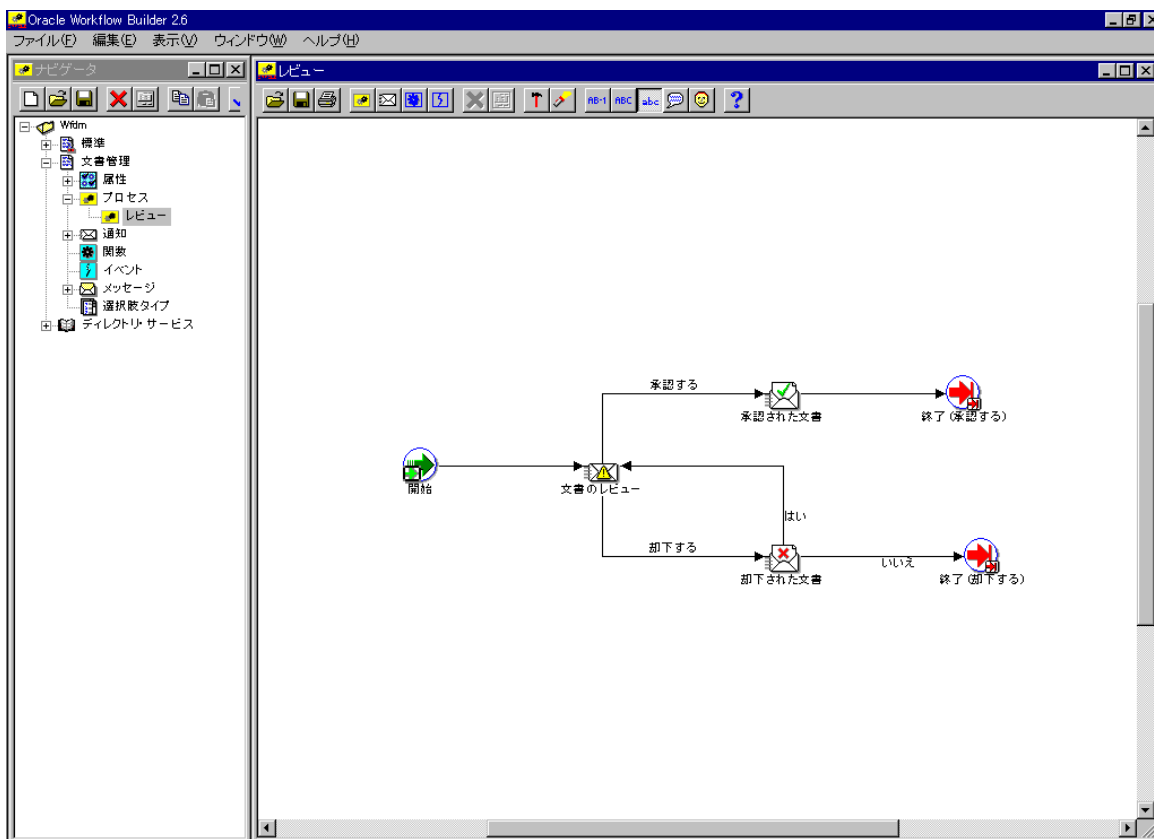
## Oracle Workflow Builder の概要

Oracle Workflow Builder は、ワークフロー・プロセス定義の作成、表示および変更に使用するグラフィカル・ツールです。Oracle Workflow Builder には「ナビゲータ」ウィンドウがあり、このウィンドウでビジネス・プロセスのアクティビティとコンポーネントを定義します。次に、プロセス・ウィンドウでアクティビティを組み合わせて、プロセス・ダイアグラムを作成します。3-8 ページの「[Oracle Workflow Builder によるプロセス定義の作成](#)」を参照してください。

---

**注意：** ワークフロー・プロセス定義は、テキスト・エディタで開いて編集できるフラット・ファイルとしても保存できます。このため、プロセス定義をスクリーン・リーダーが読み上げることができ、ユーザー補助機能を向上させることができます。

---



**注意：**「ナビゲータ」ウィンドウや Oracle Workflow Builder のプロセス・ウィンドウを最大表示にすると、[Alt] キーを使用してキーボードからメニューにアクセスできなくなります。

## ナビゲータ・ツリーの構造

「ナビゲータ」ウィンドウには、Oracle Workflow Builder で開いたり、ロードした各データ・ストアのナビゲータ・ツリー階層が表示されます。データ・ストア（最上位のブランチ）は、ワークフロー・プロセス定義を保存するデータベース接続またはフラット・ファイルです。各データ・ストア内に 1 つ以上の項目タイプのヘッダー（2 次ブランチ）があり、プロセスとそのコンポーネント・オブジェクトの特定セットのグループを表します。各項目タイプのブランチの下に次の 7 つの 3 次ブランチが表示されます。

- 属性： 現行の項目タイプの属性リスト。項目タイプ属性は、項目タイプの特徴を示します。たとえば、ある項目タイプが購買申請の場合、購買申請金額や購買申請 ID などが項目タイプ属性になります。4-2 ページの「[項目タイプ属性](#)」を参照してください。
- プロセス： 現行の項目タイプのプロセス・アクティビティまたはワークフロー・プロセス定義のリスト。5-2 ページの「[プロセス](#)」ウィンドウ」および 4-40 ページの「[アクティビティ](#)」を参照してください。
- 通知： 現行の項目タイプに関連した通知アクティビティのリスト。通知アクティビティは、ユーザーやロールにメッセージを送信します。メッセージには応答を求めるプロンプトが表示されることもあれば、単に情報が表示されることもあります。4-40 ページの「[アクティビティ](#)」を参照してください。
- 関数： 現行の項目タイプに関連した関数アクティビティのリスト。関数アクティビティは、ワークフロー・エンジンで自動的に実行される PL/SQL のストアード・プロシージャを示します。関数アクティビティには、アクティビティ属性が関連付けられることもあります。4-40 ページの「[アクティビティ](#)」を参照してください。
- イベント： 現行の項目タイプに関連したイベント・アクティビティのリスト。イベント・アクティビティは、プロセスによって受信、発生または送信されるビジネス・イベントを表します。4-40 ページの「[アクティビティ](#)」を参照してください。
- メッセージ： 現行の項目タイプに関連した通知アクティビティがユーザーやロールに送信できるメッセージのリスト。メッセージにはメッセージ属性が関連付けられることもあります。4-22 ページの「[メッセージ](#)」を参照してください。
- 選択肢タイプ： 現行の項目タイプに関連した選択肢タイプのリスト。選択肢タイプには、選択肢コードと呼ばれる 1 つ以上の値が関連付けられています。選択肢タイプは、メッセージで参照されたり、考えられる結果タイプとして通知、関数またはプロセスで参照できる値のリストです。4-18 ページの「[選択肢タイプ](#)」を参照してください。

---

---

**注意：** それぞれのデータ・ストアには、「ディレクトリ・サービス」のブランチも含まれています。「ディレクトリ・サービス」のブランチには、Oracle Workflow データベースからロードしたディレクトリ・サービスのロールがすべて表示されます。5-25 ページの「[ロール](#)」を参照してください。

---

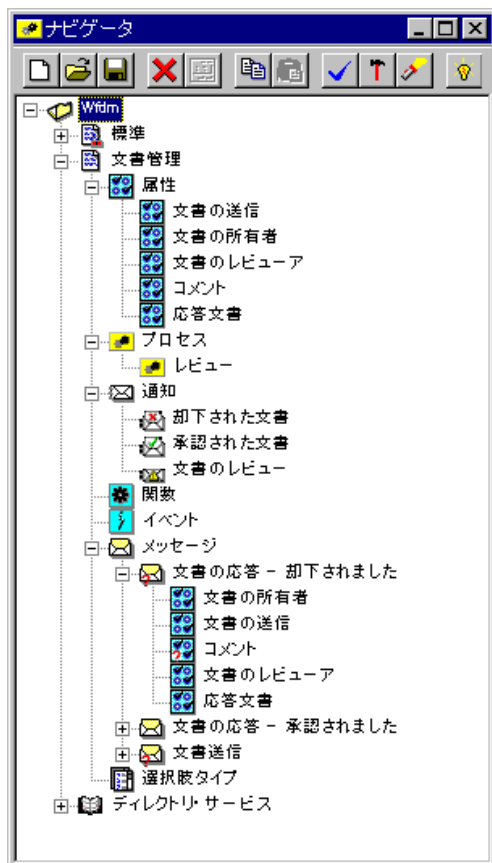
---

データ・ストアがデータベース接続されており、そのデータベースに、Oracle Workflow Builder にロードされていない他の項目タイプが含まれている場合は、「項目タイプの非表示」というブランチが表示されます。「項目タイプの非表示」をダブルクリックすると、「項目タイプの表示」ウィンドウが表示され、Oracle Workflow Builder に他の項目タイプをロードできます。

## ナビゲータ・ツリーの表示

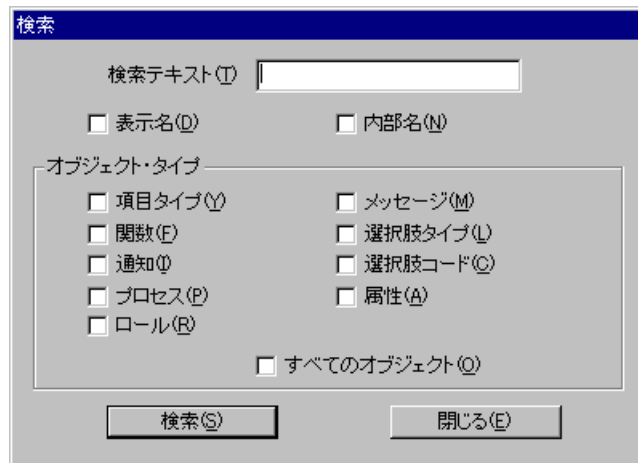
ナビゲータ・ツリーは、ファイル・システムの階層とよく似た形で編成され、対象コンポーネントが見つかるまで、先頭にプラス記号 (+) が付いているブランチをより詳細なサブブランチに展開していくことができます。サブブランチは、展開元ブランチの下にインデントされて表示されます。展開されたブランチの先頭にはマイナス記号 (-) が付きます。ブランチにプラスやマイナスの記号が付かなくなるまで、展開していくことができます。ナビゲータ・ツリーを展開または縮小するには、マウスを使用する方法と、キーボードの矢印キーを使用する方法があります。

「ナビゲータ」ウィンドウには、そのウィンドウ内でのアクションの実行に使用できるツールバーもあります。A-8 ページの「[ナビゲータのツールバー](#)」を参照してください。





## ▶ ナビゲータ・ツリーでのオブジェクトの検索



1. 「編集」メニューから「検索 ...」を選択して「検索」ウィンドウを表示します。このウィンドウで、ナビゲータ・ツリーでオブジェクトを検索する検索基準を指定します。
2. 「検索テキスト」フィールドに検索するテキストを入力します。検索時には大文字 / 小文字は区別されず、フィールドで指定したテキスト・パターンが検索されます。
3. このテキストをオブジェクトの「表示名」または「内部名」で検索するように指定します。
4. この検索範囲を限定する場合はオブジェクト・タイプを指定し、すべてのオブジェクトのプロパティ画面内でテキストを検索する場合は「すべてのオブジェクト」を選択にします。
5. 「検索」ボタンをクリックします。
6. 「編集」メニューから「再検索」を選択すると、「検索」ウィンドウで直前に定義した検索基準を使用して検索を繰り返せます。

## Oracle Workflow Builder によるプロセス定義の作成

Oracle Workflow Builder を使用する前に、プロセスで達成する内容を計画する必要があります。特に、必要なアクティビティ、各アクティビティの順序、プロセスのブランチを左右する結果、通知する相手や通知内容などを決定します。Oracle Workflow には、デモンストレーション・ワークフローのサンプルがいくつか用意されています。15-2 ページの「[サンプル・ワークフロー・プロセス](#)」を参照してください。

ワークフロー・プロセスの定義を作成するには、次の方法があります。

- トップダウン・デザイン： 上位レベルからプロセス定義を設計する場合は、最初にプロセス・ダイアグラムとアクティビティの概要を決定してから、各アクティビティでサポートしているオブジェクトを具体的に作成できます。3-11 ページの「[トップダウンによるプロセス定義の作成](#)」を参照してください。
- ボトムアップ・デザイン： 最初に詳細な部分を定義して設計する場合は、最初に下位のプロセスでサポートしているオブジェクトの定義を作成してから、上位レベルのプロセス・ダイアグラムを作成します。3-9 ページの「[ボトムアップによるプロセス定義の作成](#)」を参照してください。

### クイック・スタート・ウィザード

クイック・スタート・ウィザードを使用すると、プロセス定義のテンプレートを使用して、最初からプロセス定義を簡単に作成できます。クイック・スタート・ウィザードでは、まずプロセスに関する新しい項目タイプを作成し、必要な最低限の情報を入力するように要求されます。次に、基本のプロセス・ダイアグラムを作成し、それぞれの詳細なアクティビティを具体的に定義します。クイック・スタート・ウィザードでテンプレートを設定した後は、トップダウンまたはボトムアップのアプローチを使用して設計を完成します。3-19 ページの「[クイック・スタート・ウィザードの概要](#)」を参照してください。

### バージョンと有効日

新規のアクティビティを作成すると、Oracle Workflow Builder によりバージョン番号が割り当てられます。既存のアクティビティを変更するたびに、バージョン番号が更新されます。アクティビティの新バージョンはデータベースに保存され、旧バージョンが上書きされることはありません。Oracle Workflow では、アクティビティの有効日も設定され、常にアクティビティの 1 バージョンのみが有効になっています。Oracle Workflow では、プロセスの実行中は、そのプロセスの開始時に有効だったアクティビティのバージョンが使用されます。プロセスの途中でアクティビティのバージョンが切り替わることはありません。プロセス自体がアクティビティであるため、そのプロセスのインスタンスが完了するまで、プロセス定義は変化しないので注意してください。

また、Oracle Workflow Builder では、有効日に従ってプロセス定義を保存し、ロードすることもできます。たとえば、過去に有効になっていた定義を Oracle Workflow Builder にロードできます。また、定義をデータベースに保存し、将来の任意の時点で有効にすることもできます。

Oracle Workflow Builder では、項目タイプ、項目タイプ属性、メッセージおよび選択肢タイプなど、定数と見なされるオブジェクトのバージョン情報を管理しないので注意してください。これらのオブジェクトの場合は常に最新の定義が適用されるため、これらのオブジェクトを変更した場合、変更前と矛盾しないかどうかを必ず考慮する必要があります。変更が既存のプロセスに影響する場合は、既存のオブジェクトを編集するのではなく、新規のオブジェクトを作成する必要があります。

#### 関連項目：

4-65 ページ「[Oracle Workflow Builder のオブジェクトの変更](#)」

### プロパティ画面の「編集」ボタンの使用

Oracle Workflow Builder でオブジェクトを作成するには、そのオブジェクトのプロパティ画面で情報を入力します。入力する情報には、値リストから選択できるものもあります。Oracle Workflow Builder の他のプロパティ画面で定義された値がドロップ・ダウン・フィールドに表示される場合は、ドロップ・ダウン・リストの右に「編集」ボタンが表示されます。ドロップ・ダウン・リストで値を選択すると、隣の「編集」ボタンを選択し、その値のソース・プロパティ画面を表示して編集できます。ソース・プロパティ画面の設定を完了して「OK」または「キャンセル」をクリックすると、作業していた元のプロパティ画面に戻ります。

たとえば、通知アクティビティを作成する場合は、そのアクティビティの「結果タイプ」を指定する必要があります。「結果タイプ」のドロップ・ダウン・フィールドを使用して、値「なし」または事前定義の選択肢タイプを選択できます。選択肢タイプを選択した場合は、隣の「編集」ボタンを選択し、その選択肢タイプのプロパティ画面を表示できます。選択肢タイプのプロパティ画面を表示または編集した後は、「OK」または「キャンセル」を選択すると、通知アクティビティのプロパティ画面に戻ります。

#### ▶ ボトムアップによるプロセス定義の作成

1. Oracle Workflow Builder を起動するには、「Oracle - <SID NAME>」プログラム・グループの「Application Development」フォルダに入っている「Oracle Workflow Builder」アイコンをダブルクリックします。Windows NT 4.0 以上を使用している場合は、「スタート」メニューから適切なプログラム・フォルダの「Oracle Workflow Builder」アイコンを選択する方法もあります。
2. 「ファイル」メニューから「新規作成」を選択し、新しいプロセス定義のための作業領域を作成します。

---

**提案：** クイック・スタート・ウィザードを使用して、新しいプロセス定義の概略を最初に作成することもできます。一度クイック・スタート・ウィザードで新しい項目タイプとプロセス・アクティビティを作成すると、手順 4 に進んで、新しい項目タイプとプロセス・アクティビティでサポートするオブジェクトの定義を開始できます。3-19 ページの「[クイック・スタート・ウィザードの使用](#)」を参照してください。

---

3. 新しい項目タイプを作成します。項目タイプでは、プロセスで管理する作業項目を分類します。4-6 ページの「[項目タイプの作成](#)」を参照してください。
4. 項目タイプの属性を定義して項目タイプを具体的に記述し、プロセス内のアクティビティでこれらの属性の情報を参照できるようにすることができます。4-8 ページの「[項目タイプまたはアクティビティ属性の定義](#)」を参照してください。
5. 新規の選択肢タイプを作成します。4-18 ページの「[選択肢タイプの作成](#)」を参照してください。

アクティビティを定義する前に、その「結果タイプ」を表す選択肢タイプを定義する必要があります。結果タイプは、アクティビティの完了時に考えられる結果のリストです。選択肢タイプとアクティビティを定義した後、選択肢をナビゲータ・ツリーのアクティビティにドラッグし、その選択肢をアクティビティの結果タイプに設定できます。選択肢タイプは、項目タイプ属性、アクティビティ属性、メッセージまたはメッセージ属性で参照することもできます。

6. 新規のメッセージを作成します。4-27 ページの「[メッセージの作成](#)」を参照してください。

プロセスの通知アクティビティを作成する場合は、通知アクティビティで送信するメッセージを最初に作成する必要があります。新しいメッセージをナビゲータ・ツリーの通知アクティビティにドラッグすると、対象アクティビティにメッセージを割り当てることができます。

メッセージのメッセージ属性も作成できます。動的な内容を表すように実行時にトークンで置き換えられるメッセージには、「送信」タイプのメッセージ属性を組み込むことができます。また、「応答」タイプのメッセージ属性を定義して、通知の受信者が受信時に応答するように設定できます。4-32 ページの「[メッセージ属性の定義](#)」を参照してください。

7. 新規のプロセス・アクティビティ、通知アクティビティ、関数アクティビティ、またはイベント・アクティビティを作成します。標準項目タイプに関連付けられている事前に定義された標準アクティビティを使用することもできます。4-40 ページの「[アクティビティ](#)」および 6-2 ページの「[標準アクティビティ](#)」を参照してください。

上位プロセス・ダイアグラムを表すプロセス・アクティビティを、1 つ以上は定義する必要があります。プロセス・ダイアグラムにより、プロセス内のすべてのアクティビティ間の関連が設定されます。

8. プロセス・ダイアグラムを作成します。

該当するプロセス・アクティビティの「プロセス」ウィンドウを表示し、ワークフロー・プロセスを定義するアクティビティとトランジションのダイアグラムを作成します。アクティビティは、ナビゲータ・ツリーから「プロセス」ウィンドウにドラッグできます。5-5 ページの「[プロセスの図示](#)」を参照してください。

9. 「ファイル」メニューから「保存」または「名前を付けて保存」を選択し、作業内容を保存します。3-16 ページの「[作業内容の保存](#)」を参照してください。

10. Oracle Workflow Server からアクセスできるデータベース内で、PL/SQL 関数アクティビティによってコールされる PL/SQL ストアド・プロシージャを作成します。この操作には、SQL\*Plus または Oracle Procedure Builder を使用できます。8-1 ページの「[Oracle Workflow の API](#)」および 7-3 ページの「[関数アクティビティがコールする PL/SQL プロシージャの標準 API](#)」を参照してください。

**関連項目：**

- 3-12 ページ「[プロセス定義の変更](#)」
- 4-64 ページ「[Oracle Workflow Builder のオブジェクトの削除](#)」
- 4-65 ページ「[Oracle Workflow Builder のオブジェクトの変更](#)」
- 3-25 ページ「[項目タイプの定義](#)」 [Web 画面](#)

► **トップダウンによるプロセス定義の作成**

1. Oracle Workflow Builder を起動するには、「Oracle - <SID NAME>」プログラム・グループの「Application Development」フォルダに入っている「Oracle Workflow Builder」アイコンをダブルクリックします。Windows NT 4.0 以上を使用している場合は、「スタート」メニューから適切なプログラム・フォルダの「Oracle Workflow Builder」アイコンを選択する方法もあります。
2. クイック・スタート・ウィザードを使用して、新しいプロセス定義の概略を作成します。新規の項目タイプとプロセス・アクティビティに必要な情報を指定します。3-19 ページの「[クイック・スタート・ウィザードの使用](#)」を参照してください。
3. 「プロセス」ウィンドウが表示され、「開始アクティビティ」と「終了アクティビティ」のノードが示されます。開始ノードと終了ノードの間に新しいアクティビティ・ノードを定義し、プロセスのダイアグラムを作成します。5-8 ページの「[プロセスのノードの定義](#)」を参照してください。

標準項目タイプに関連付けられている事前に定義された標準アクティビティを使用することもできます。6-2 ページの「[標準アクティビティ](#)」を参照してください。

4. アクティビティ間の遷移を示して、プロセスをモデル化します。5-5 ページの「[プロセスの図示](#)」を参照してください。
5. 「ファイル」メニューから「保存」または「別名保存」を選択し、作業内容を保存します。3-16 ページの「[作業内容の保存](#)」を参照してください。

---

---

**注意：** Oracle Workflow では、作業内容を保存すると、不正なプロセス定義や不足している情報が自動的に検証され、該当するものがあれば「ワークフロー・エラー」検証ウィンドウに表示されます。「ワークフロー・エラー」ウィンドウは非モーダル・ウィンドウで、このウィンドウを開いたまま、プロセスに戻って識別されたエラーを修正できます。また、作業内容をそのまま保存してから、エラーを修正することもできます。他の文書で後で参照できるように、情報をクリップボードに貼り付けるには、「コピー」ボタンを使用して情報をコピーします。エラーを修正せずに作業内容を保存すると、次にこのプロセス定義をオープンしたときに「ワークフロー・エラー」ウィンドウが表示されます。

---

---

**関連項目：**

- 3-12 ページ [「プロセス定義の変更」](#)
- 4-64 ページ [「Oracle Workflow Builder のオブジェクトの削除」](#)
- 4-65 ページ [「Oracle Workflow Builder のオブジェクトの変更」](#)
- 3-25 ページ [「項目タイプの定義」 Web 画面](#)

➤ **プロセス定義の変更**

1. Oracle Workflow Builder を起動するには、「Oracle - <SID NAME>」プログラム・グループの「Application Development」フォルダに入っている「Oracle Workflow Builder」アイコンをダブルクリックします。Windows NT 4.0 以上を使用している場合は、「スタート」メニューから適切なプログラム・フォルダの「Oracle Workflow Builder」アイコンを選択する方法もあります。
2. 「ファイル」メニューから「オープン」を選択し、データベースへの接続をオープンするか、変更するプロセス定義が含まれているファイルをオープンします。3-13 ページの[「既存データ・ストア内のプロセス定義へのアクセス」](#)を参照してください。
3. 変更するプロセス定義に関連付けられている既存の項目タイプを選択し、展開します。
4. 項目タイプ、項目タイプ属性、選択肢、メッセージ、メッセージ属性、プロセス・アクティビティ、通知アクティビティ、関数アクティビティまたはアクティビティ属性を変更できます。4-6 ページの[「項目タイプの作成」](#)、4-8 ページの[「項目タイプまたはアクティビティ属性の定義」](#)、4-18 ページの[「選択肢タイプの作成」](#)、4-27 ページの[「メッセージの作成」](#)、4-32 ページの[「メッセージ属性の定義」](#)または4-40 ページの[「アクティビティ」](#)を参照してください。
5. プロセス・アクティビティの「プロセス」ウィンドウを表示し、プロセス・ダイアグラムを変更することもできます。5-5 ページの[「プロセスの図示」](#)を参照してください。
6. 「ファイル」メニューから「保存」または「別名保存」を選択し、作業内容を保存します。3-16 ページの[「作業内容の保存」](#)を参照してください。

**関連項目：**

4-64 ページ「[Oracle Workflow Builder のオブジェクトの削除](#)」

4-65 ページ「[Oracle Workflow Builder のオブジェクトの変更](#)」

3-25 ページ「[「項目タイプの定義」 Web 画面](#)」

## 項目タイプのオープンと保存

すべてのプロセスは、項目タイプに関連付けられます。1つの項目タイプに、1つ以上のプロセスを含めることができます。項目タイプは、データベースやフラット・ファイルに保存できます。作業結果をデータベースに保存すると、実際には現行データ・ストア内で変更があった情報がすべて保存されます。フラット・ファイルに保存すると、実際には現行データ・ストア内のすべての情報がファイルに保存されます。また、項目タイプをデータベースやフラット・ファイルから Oracle Workflow Builder にロードすることもできます。項目タイプをオープンすると、その項目タイプに関連した属性、メッセージ、選択肢、通知、関数およびプロセスすべてが自動的に取り出されます。

---

---

**注意：** ワークフロー・プロセス定義のコピーを常にフラット・ファイルとして保存し、そのファイルをソース・コントロール・システムにチェックインして、プロセス定義の作業用バージョンを保守してください。データベースにアクセスする他のユーザーが定義を更新できるように、データベースに格納されたプロセス定義は、ソース・コントロール・バージョンとして使用しないでください。

---

---

---

---

**注意：** Oracle Workflow Builder からデータベースに接続するには、インストールされている Oracle Workflow Builder の言語が、データベースにインストールされている Oracle Workflow Server の使用可能な言語のいずれかと一致している必要があります。

---

---

### ▶ 既存データ・ストア内のプロセス定義へのアクセス

1. Oracle Workflow Builder を起動するには、「Oracle - <SID NAME>」プログラム・グループの「Application Development」フォルダに入っている「Oracle Workflow Builder」アイコンをダブルクリックします。Windows NT 4.0 以上を使用している場合は、「スタート」メニューから適切なプログラム・フォルダの「Oracle Workflow Builder」アイコンを選択する方法もあります。Oracle Workflow Builder で、「ファイル」メニューから「開く」を選択します。



2. データベースまたはファイルを選択し、プロセス定義が関連付けられている項目タイプを含むソースに接続します。
3. ファイルを開く場合： ファイルのフルパスを指定して「OK」を選択するか、「参照」をクリックし、ファイル位置を指定してファイル（拡張子 .wft）を開きます。

---

**注意：** Microsoft 98/2000 や Windows NT 4.0 の「エクスプローラ」、または Microsoft Windows NT の「ファイル マネージャ」から、.wft ファイルをナビゲータ・ツリーにドラッグ・アンド・ドロップし、そのファイルを Oracle Workflow Builder で開くこともできます。

---

---

**注意：** 「参照」を使用してファイルを検索してオープンすると、オープンしたファイルが含まれている現行のディレクトリが、次にファイルをオープンするときのデフォルトのディレクトリとなります。このデフォルトのディレクトリは、次に「参照」を使用して他のファイルを検索するまで保持されます。

---

4. データベース接続を開く場合： データベースのユーザー名とパスワードを入力します。データベースの別名または接続文字列を入力し、「OK」を選択します。



---

**注意：** Oracle Applications embedded Workflow を使用している場合は、APPS スキーマを使用してデータベースに接続します。

---

5. 特定の日に有効であったプロセス定義を取り出す場合は、「有効日」フィールドで日時を指定して、そのデータをデータベースから取り出します。日時の指定に使用する書式は、Windows の「コントロール パネル」の「地域」設定で定義した日時の書式に基づきます。



6. データ・ストアに複数の項目タイプが存在する場合は、「項目タイプの表示」ウィンドウが表示されます。表示する項目タイプを「非表示」リストから選択し、「<<」を選択して「表示」リストに移動します。「OK」を選択し、選択した項目タイプをナビゲータ・ツリーにロードします。
7. 現行のデータ・ストア内で非表示の項目タイプを表示して変更する場合は、ナビゲータ・ツリーの「項目タイプの非表示」のブランチをダブルクリックして、「項目タイプの表示」ウィンドウを表示し、表示する項目タイプを選択します。また、「項目タイプの表示」ウィンドウを表示するには、「ファイル」メニューから「項目タイプの表示 / 非表示」を選択する方法もあります。

---

---

**注意：** 項目タイプは、相互に参照している場合でも、データ・ストア間では任意の順序でコピーできます。ただし、外部キーの参照によって、検証エラーとなることがあります。このようなエラーは、外部キーの参照を解決するために、場合によっては他の項目タイプを新しいデータ・ストアへコピーする必要があるので、注意が必要です。参照されている項目タイプがすべて、コピー先の新しいデータ・ストアにコピーされると、新しいデータ・ストアの最終プロセス定義が有効になります。

---

---

8. 作業の完了後に、「ファイル」メニューから「保存」を選択し、変更内容を保存して即時に有効にします。3-16 ページの「[作業内容の保存](#)」を参照してください。

**関連項目：**

3-18 ページ「[コマンド・プロンプトからの Oracle Workflow Builder の起動](#)」

► **作業内容の保存**

1. 「ファイル」メニューから「保存」を選択し、変更内容を保存して即時に有効にします。

「保存」コマンドを使用すると、現在選択しているデータ・ストアの変更済オブジェクトがすべて（非表示でも）データ・ストアに保存されます。特定の項目タイプのみを保存する場合は、新規のデータ・ストアを作成し、保存する項目タイプを新規ストアにコピーして、新規ストアを保存する必要があります。

---

---

**注意：** Oracle Workflow Builder では、2 つのモードのどちらかで作業結果をデータベースに保存できます。「ヘルプ」メニューから「Oracle Workflow Builder のバージョン情報」ダイアログ・ボックスを表示すると、「カスタマイズされたオブジェクトの変更を可能にする」というチェック・ボックスが表示されます。このチェック・ボックスをオンにすると、編集結果はアップロード・モードで保存され、以前にカスタマイズしたオブジェクトと、変更のためにアクセスした保護対象のオブジェクトが上書きされます。このチェック・ボックスをオフにすると、Oracle Workflow Builder はアップグレード・モードで実行されます。このモードでは、変更のためにアクセスした保護対象のオブジェクトの編集結果のみが保存され、以前にカスタマイズしたオブジェクトは上書きされません。この 2 つのモードは、ワークフロー定義ローダーのプログラムのアップグレードおよびアップロード動作と同じです。デフォルトでは、このチェック・ボックスはオフになっています。4-17 ページの「[オブジェクトのアクセス・レベルの設定](#)」および 2-101 ページの「[ワークフロー定義ローダーの使用](#)」を参照してください。

---

---

2. 作業結果を別のデータ・ストア（データベースまたはフラット・ファイル）に保存する場合や、現在のシステム日付とは異なる有効日付でデータベースに保存する場合は、「ファイル」メニューから「別名保存」を選択します。「別名保存」ウィンドウで、プロセス定義を保存するファイルまたはデータベースを指定し、プロセス定義がデータベースで有効になる日付を指定します。「有効日」フィールドを空白にすると、変更を保存した直後から有効にすることができます。8-13 ページの「バージョン / 有効日付」を参照してください。

---

**注意：** 将来の有効日を指定して作業結果をデータベースに保存し、同じ Oracle Workflow Builder セッションでプロセスの変更を続けた後に、「ファイル」メニューから「保存」を選択すると、プロセス定義は以前に指定した有効日を使用して同じデータベースに自動的に保存されます。

---

3. Oracle Workflow では、作業内容を保存すると、不正なプロセス定義や不足している情報が自動的に検証され、該当するものがあれば「ワークフロー・エラー」検証ウィンドウに表示されるため注意してください。情報を修正してから作業内容を保存することも、作業内容を保存してからエラーを修正することもできます。後で参照するために、クリップボードに情報を貼り付けるには、「コピー」ボタンを使用して「ワークフロー・エラー」ウィンドウの情報をコピーします。エラーを修正せずに作業内容を保存すると、次にこのプロセス定義をオープンしたときに「ワークフロー・エラー」ウィンドウが表示されます。
4. 「ファイル」メニューから「ストアを閉じる」を選択し、現行データベースまたはファイル・データ・ストアへの接続を終了します。
5. 「ファイル」メニューから「終了」を選択し、Oracle Workflow Builder を終了します。

---

---

**注意：**「ファイル」メニューから「ストアを閉じる」および「終了」オプションを使用できるのは、「ナビゲータ」ウィンドウが現行ウィンドウの場合のみです。

---

---

### ► コマンド・プロンプトからの Oracle Workflow Builder の起動

Windows のアイコンをダブルクリックして Oracle Workflow Builder を起動するのではなく、コマンド・プロンプト（Windows98 では MS-DOS プロンプトと呼ぶ）からコマンドを入力して、接続先のファイルやデータベースを指定することもできます。

1. 「コマンドプロンプト」ウィンドウで、特定のワークフロー・データ・ファイルを指定して次のコマンドを入力し、Oracle Workflow Builder を起動します。

<filename.wft> はデータ・ファイルのフルパス名を表します。

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

2. 特定のデータベース接続を指定して Oracle Workflow Builder を起動するには、コマンド・プロンプトから次のコマンドを入力します。<username/password@connect> は、接続先データベースのアカウント情報を表します。

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

---

---

**注意：** Oracle Workflow Builder を Microsoft Windows NT 4.0 以上で実行する場合は、Windows の「エクスプローラ」上でワークフロー・データ・ファイル（.wft）をダブルクリックすると、そのファイルが自動的に開き、Oracle Workflow Builder が起動します。

---

---

---

---

**注意：** Oracle Applications embedded Workflow を使用している場合は、APPS スキーマを使用してデータベースに接続します。

---

---

3. Oracle Workflow Builder を起動し、データ・ストア内で指定した項目タイプをオープンするには、手順 1 または 2 の適切なコマンドを次のように修正します。<item\_type> は、オープンする項目タイプの内部名です。

```
-E <item_type>
```

たとえば、次のようになります。

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

4. Oracle Workflow Builder を起動し、データ・ストア内で指定したプロセス・ダイアグラムをオープンするには、手順 1 または 2 の適切なコマンドを次のように修正します。<item\_type:process> は、オープンする項目タイプとプロセスの内部名です。

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

たとえば、次のようになります。

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

#### 関連項目：

2-101 ページ「ワークフロー定義ローダーの使用」

5-24 ページ「ワークフロー・プロセスのショートカット・アイコンの作成」

## クイック・スタート・ウィザードの概要

クイック・スタート・ウィザードを使用すると、ワークフロー・プロセスの設計をその場で開始できます。このウィザードでは、最初に `wftemplate.wft` というファイル（ワークフロー・プロセスの作成に必要なすべての必須オブジェクトの概略）がロードされ、プロセスのダイアグラムを作成する「プロセス」ウィンドウが表示されます。クイック・スタート・ウィザードを開始した後は、ボトムアップまたはトップダウンのアプローチを使用して、ワークフロー・プロセスの定義を完了します。

### ▶ クイック・スタート・ウィザードの使用

1. 「ファイル」メニューから「クイック・スタート・ウィザード」を選択します。

2. 「Workflow クイック・スタート・ウィザード」ウィンドウに、次の必須情報の入力を求めるプロンプトが表示されます。

■ 新規項目タイプ

- 内部名： 大文字のみを使用し、半角英数字で 8 文字以内の内部名を指定します。すべての Oracle Workflow API、SQL スクリプトおよび PL/SQL プロシージャでは、項目タイプの識別時に内部名が参照されます。

---

**注意：** 定義された項目タイプの内部名を更新するには、`wfchitt.sql` という特別な SQL スクリプトを使用する必要があります。16-9 ページの「[wfchitt.sql](#)」を参照してください。

---

---

**注意：** 内部名には、コロン (:) や空白を使用しないでください。

---

- 表示名： 項目タイプを表す変換可能な表示名を入力します。
- 維持タイプ： 項目タイプの実行ステータス情報として「一時」または「永続」を指定します。
- 日数：「維持タイプ」が「一時」の場合は、項目タイプのインスタンスが終了してから、実行ステータス情報が削除可能になるまでの日数を指定します。4-4 ページの「[維持タイプ](#)」を参照してください。

■ 新規プロセス

- 内部名： 内部名を大文字のみで指定します。

---

**注意：** 定義されたアクティビティの内部名を更新するには、`wfchact.sql` という特別な SQL スクリプトを使用する必要があります。16-8 ページの「[wfchact.sql](#)」を参照してください。

---

---

**注意：** 内部名には、コロン (:) や空白を使用しないでください。

---

- 表示名： プロセス・アクティビティを表す変換可能な表示名を入力します。表示名は、「プロセス」ウィンドウのタイトル・バーにも表示されます。

3. クイック・スタート・ウィザードにより、次の処理が実行されます。

- 「ナビゲータ」ウィンドウに「タイトルなし - n」というデータ・ストアが作成されます。
- 「Workflow クイック・スタート・ウィザード」ウィンドウに入力した情報が使用され、データ・ストアに新しい項目タイプとプロセス・アクティビティが作成されます。

- 作成するプロセスに標準のアクティビティを組み込めるように、標準項目タイプが新しいデータ・ストアにロードされます。
  - 定義した新しいプロセス・アクティビティの「プロセス」ウィンドウが開きます。「プロセス」ウィンドウには、「開始アクティビティ」および「終了アクティビティ」が表示されます。
4. 次の2つの方法のどちらかを使用して、プロセスの定義をカスタマイズできます。
- ボトムアップ・デザイン： ワークフロー・ダイアグラムを作成する前に、まずアクティビティを作成し、それらのアクティビティでサポートしているすべてのオブジェクトを作成します。3-9 ページの「[ボトムアップによるプロセス定義の作成](#)」を参照してください。
  - トップダウン・デザイン： ワークフロー・ダイアグラムの作成に必要な、最低限の情報が含まれているアクティビティを作成します。後で各アクティビティおよびサポートしているオブジェクトに戻って、具体的な内容を詳しく記述できます。3-11 ページの「[トップダウンによるプロセス定義の作成](#)」を参照してください。

## 各種リリースのサーバーでの Oracle Workflow Builder の使用

Oracle Workflow Builder リリース 2.6.2 は、Oracle Applications embedded Workflow リリース 11i の全リリースだけでなく、Oracle Workflow スタンドアロン版のリリース 2.6.2、リリース 2.6.1、リリース 2.6 およびリリース 2.5 と互換性があります。

- ビジネス・イベント・システムのコンポーネントや外部 Java 関数アクティビティなど、リリース 2.6 で導入された新機能が含まれているワークフロー・プロセス定義を作成、表示および変更できます。Oracle Workflow Builder では、Oracle Workflow Server リリース 2.6 がインストールされているデータベースに対して、これらのプロセス定義をアップロード / ダウンロードできます。
- リリース 2.6 の機能を利用しない場合は、リリース 2.5 の機能だけが含まれるワークフロー・プロセスを作成、表示および変更できます。Oracle Workflow Builder では、Oracle Workflow Server リリース 2.5 がインストールされているデータベースに対して、これらのプロセス定義をアップロード / ダウンロードできます。

---

**注意：** Oracle Workflow Server リリース 2.6.2 のクライアント・コンポーネントのうち、以前のバージョンと互換性があるのは、Oracle Workflow Builder だけです。Oracle Workflow Mailer リリース 2.6.2 は、以前のバージョンのサーバーと互換性がありません。Oracle Workflow Server リリース 2.6.2 でのみ動作が保証されています。

---

### Embedded Server リリース 2.6、スタンドアロン版 Server リリース 2.6.2 またはリリース 2.6.1 で Oracle Workflow Builder リリース 2.6.2 を使用する

Oracle Applications embedded Workflow リリース 2.6、あるいは Oracle Workflow リリース 2.6.2 またはリリース 2.6.1 スタンドアロン版で Oracle Workflow Builder を使用している場合は、現在使用可能な機能をすべてワークフロー・プロセスで使用できます。これらのプロセス定義をデータベースに保存しておけば、データベースのプロセス定義をオープンして表示または変更することができます。

また、Oracle Workflow Builder リリース 2.5 またはリリース 2.6 を使用して作成した既存のプロセス定義をオープンし、Oracle Applications embedded Workflow リリース 2.6、Oracle Workflow リリース 2.6.2 またはリリース 2.6.1 スタンドアロン版がインストールされているデータベースに保存することもできます。

### スタンドアロン版 Server リリース 2.6 で Oracle Workflow Builder リリース 2.6.2 を使用する

Oracle Workflow リリース 2.6 スタンドアロン版で Oracle Workflow Builder を使用している場合は、現在使用可能な機能のほとんどをワークフロー・プロセスで使用できます。ただし、「イベント・パラメータ」選択肢コードは使用しないでください。この機能は、Oracle Applications embedded Workflow リリース 2.6 で追加されましたが、現在は「イベント・プロパティ」選択肢タイプに関連付けられています。「イベント・パラメータ」選択肢コー



ドは、カスタム・アクティビティでは使用しないでください。また、「イベント・パラメータ」プロパティは、次の標準ワークフロー・アクティビティで選択しないでください。

- イベント・パラメータの取得
- イベント・パラメータの設定
- イベント・パラメータの比較

Oracle Workflow Builder リリース 2.6 を使用して作成した既存のプロセス定義をオープンし、リリース 2.6 のコンポーネントだけを使用して表示または変更し、Oracle Workflow リリース 2.6 スタンドアロン版がインストールされているデータベースに保存することができます。

また、リリース 2.6 のコンポーネントだけを使用して、新しいワークフロー・プロセスを作成することもできます。ただし、Oracle Workflow Builder リリース 2.6.2 の標準項目タイプには、リリース 2.6 のコンポーネントが一部含まれます。Oracle Workflow リリース 2.6 スタンドアロン版がインストールされているデータベースに新しいプロセス定義を保存する場合は、次の手順を実行します。

1. ワークフロー・プロセス定義を新規作成します。
2. 標準項目タイプをデータ・ストアから強制的に削除します。
3. Oracle Workflow スタンドアロン版リリース 2.6 がインストールされているデータベースにプロセス定義を強制的に保存します。
4. データベースからこのプロセス定義を再度オープンします。このプロセス定義には、現在リリース 2.6 の標準項目タイプが含まれています。
5. リリース 2.6 のベース・コンポーネントだけを使用して、ワークフロー・プロセスの定義を続行します。

## スタンドアロン版 Server リリース 2.5 または Embedded Server リリース 2.5 で Oracle Workflow Builder リリース 2.6.2 を使用する

Oracle Workflow リリース 2.5 スタンドアロン版または Oracle Applications embedded Workflow リリース 2.5 で Oracle Workflow Builder を使用している場合は、リリース 2.5 の機能だけをワークフロー・プロセスで使用してください。リリース 2.6 で導入された次の新機能は使用しないでください。

- 「イベント」アクティビティ
- イベント・タイプの項目属性
- 「外部 Java」関数アクティビティ

Oracle Workflow Builder リリース 2.5 を使用して作成した既存のプロセス定義をオープンし、リリース 2.5 のコンポーネントだけを使用して表示または変更し、Oracle Workflow Server リリース 2.5 がインストールされているデータベースに保存することができます。

また、リリース 2.5 のコンポーネントだけを使用して、新しいワークフロー・プロセスを作成することもできます。ただし、**Oracle Workflow Builder** リリース 2.6.2 の標準項目タイプには、リリース 2.6 のコンポーネントが一部含まれます。**Oracle Workflow Server** リリース 2.5 がインストールされているデータベースに新しいプロセス定義を保存する場合は、次の手順を実行します。

1. ワークフロー・プロセス定義を新規作成します。
2. 標準項目タイプをデータ・ストアから強制的に削除します。
3. **Oracle Workflow Server** リリース 2.5 がインストールされているデータベースにプロセス定義を強制的に保存します。
4. データベースからこのプロセス定義を再度オープンします。このプロセス定義には、現在リリース 2.5 の標準項目タイプが含まれています。

## 「項目タイプの定義」Web 画面

Web ベースの「項目タイプの定義」ページを使用すると、Oracle Workflow データベースに格納されているワークフロー定義への個別アクセスが可能になります。このページでは、特定の項目タイプに関連した属性、プロセス、通知、関数、イベント、メッセージおよび選択肢タイプの詳細を表示して、ワークフロー・プロセスを表示したり、その構成を検討できます。

項目タイプ定義を表示するには、最初に「項目タイプの検索」Web 画面を使用して項目タイプを問い合わせます。項目タイプは、有効な日付および時刻に基づいて問い合わせできます。

「項目タイプの定義」ページが表示されます。情報は、2 つのフレームに表示されます。各フレームは Oracle Workflow Builder のようにモデル化されており、内容を簡単に効率よく検討できます。左のフレームには、項目タイプ定義に含まれるすべてのオブジェクトが、展開可能なナビゲータ・ツリーに表示されます。右のフレームには、ナビゲータ・ツリーで選択したオブジェクトの詳細が表示されます。また、いつでも左右どちらかのフレームを選択し、そのフレーム内のすべての情報を Web ブラウザを使用して印刷できます。

### ▶ 項目タイプの問合せ

1. Web ブラウザに次の URL を入力します。

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

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください

---

---

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。

---

---

---

---

**注意：** 「項目タイプの検索」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---

---



2. 「項目タイプ」ドロップ・ダウン・フィールドを使用して、項目タイプを選択します。
3. 「ユーザー設定項目」Web 画面の「日付書式」フィールドで指定した書式を使用して、表示する項目タイプ定義の有効日時を指定します。9-6 ページの「[ユーザー設定項目の設定](#)」を参照してください。
4. 「検索」を選択して「項目タイプの定義」Web 画面に項目タイプを表示します。



## ▶ 項目タイプの定義の検討

1. 「項目タイプの定義」Web 画面には、2つのフレームが表示されます。左のフレームには、Oracle Workflow Builder のナビゲータ・ツリーと同様の階層形式で、項目タイプ定義のコンポーネントが表示されます。右のフレームには、各コンポーネントの詳細が表示されます。
2. 左側のフレームにあるいずれかのコンポーネントのリンクをクリックします。そのコンポーネントの詳細が右側のフレームに表示されます。



---

## ワークフロー・プロセスのコンポーネントの定義

この章では、Oracle Workflow Builder を使用して、ワークフロー・プロセス・ダイアグラムの構成に必要なコンポーネントを定義する方法について説明します。

## ワークフロー・プロセスのコンポーネント

作成するワークフロー・プロセスによっては、プロセスを構成する次のタイプのコンポーネントのすべてまたは一部を定義する必要があります。

- 項目タイプ
- 選択肢タイプ
- メッセージ
- アクティビティ
- 属性
- ロール

### 項目タイプ

項目タイプは、ワークフロー・プロセスを構成するコンポーネントの分類です。プロセス用に作成するコンポーネント（関数アクティビティやメッセージなど）は、すべて特定の項目タイプに関連付ける必要があります。多くの場合、項目タイプを定義して、ワークフロー・プロセスで管理する項目を記述すると理解しやすくなります。たとえば、購買申請を項目タイプとして定義し、特定の ID 番号で識別される購買申請をその項目タイプの項目にすることができます。4-6 ページの「[項目タイプの作成](#)」を参照してください。

### 項目タイプ属性

項目タイプ属性は、特定の項目タイプに関連付けられたプロパティです。項目タイプ属性はグローバル変数として機能し、プロセス内のすべてのアクティビティから参照、更新できます。通常、項目タイプ属性は、ワークフロー・プロセスを完了させるために必要な項目の情報を提供します。たとえば、「ワークフロー・デモンストレーション」という項目タイプには、「購買申請金額」という項目タイプ属性があります。この例で、「購買承認申請」プロセスのアクティビティでは、選択された承認者にその金額の購買申請を承認する権限があるかどうかを判別するために、この項目タイプ属性の値が必要です。

関数アクティビティと同様に、アプリケーションでも、Oracle Workflow Engine API を使用して項目タイプ属性を参照したり設定できます。項目タイプに必要な数だけ、項目タイプ属性を定義して保守できます。プロセス内のアクティビティに必要な情報や、通知メッセージでの送信が必要な情報は、すべて項目タイプ属性として定義する必要があります。4-32 ページの「[メッセージ属性の定義](#)」を参照してください。

#### 関連項目：

C-3 ページ「[項目属性](#)」



## 属性タイプ

次のように、属性は 10 のタイプに分かれています。タイプによって、使用可能な値や属性の使用方法が決まります。

- テキスト： 属性値はテキストの文字列です。
- 数値： 属性値は、指定したオプションの書式マスク付きの数値です。
- 日付： 属性値は、指定したオプションの書式マスク付きの日付です。
- 選択肢： 属性値は、指定した選択肢タイプの選択肢コードの値の 1 つです。
- フォーム： 属性値は、Oracle Applications の内部フォーム機能名とオプションのフォーム機能パラメータです。この属性タイプは、Oracle Workflow のスタンドアロン版には関係ありません。

通知メッセージに、メッセージ属性としてフォーム・タイプの属性を含める場合、「通知の詳細」 Web 画面から表示する通知には添付フォーム・アイコンが表示され、参照されているフォームまでドリルダウンできます。『Oracle Applications Developer's Guide』の「Overview of Menus and Function Security」を参照してください。

- URL： 属性値は、ネットワーク位置の Universal Resource Locator (URL) です。通知メッセージで、メッセージ属性として URL 属性を参照すると、「通知の詳細」 Web 画面から、または HTML 形式の電子メールとして表示した通知には、URL 属性で指定した URL へのアンカーが表示されます。ユーザーはこの URL にアクセスして、アクティビティを完了したり、アクティビティ関連の追加情報を表示できます。
- 文書： 属性値は添付文書です。デフォルト値フィールドで、次のタイプの文書を指定できます。
  - PL/SQL 文書： データベースのデータを文字列として表す文書で、PL/SQL プロシージャから生成されます。
  - PL/SQL CLOB 文書： データベースのデータをキャラクタ・ラージ・オブジェクト (CLOB) として表す文書で、PL/SQL プロシージャから生成されます。

**参照：**4-13 ページの「[文書属性の定義](#)」を参照してください。

- ロール： 属性値はロールの内部名です。通知メッセージにロール・タイプのメッセージ属性が含まれる場合、属性は自動的にロールの表示名に設定されるため、ロールの内部名と表示名について別々の属性を保守する必要がなくなります。また、Web ブラウザから通知を表示する場合は、ロールの表示名がそのロールの電子メール・アドレスを示すハイパーテキスト・リンクになります。属性のデフォルト値を設定するには、最初にデータベースからロールをロードする必要があります。5-25 ページの「[ロール](#)」を参照してください。
- 属性： 属性値は、プロセスで参照を維持する別の既存項目タイプ属性の内部名です。
- イベント： 属性値は、標準 WF\_EVENT\_T 構造のビジネス・イベント・システムのイベント・メッセージです。8-248 ページの「[イベント・メッセージ構造](#)」を参照してください。

---

---

**注意：** イベント・メッセージをイベント・タイプの項目属性で保存すると、URL タイプの項目属性を作成し、イベント・データを参照する URL 属性値を設定することにより、そのイベント・メッセージ内のイベント・データにアクセスできます。8-51 ページの「[SetItemAttribute](#)」を参照してください。

---

---

## 維持タイプ

項目タイプを定義するときに、その維持タイプも指定する必要があります。維持タイプにより、項目タイプのインスタンスごとに維持される実行ステータス情報の期間を制御します。「維持」を「永続」に設定すると、ランタイム・ステータス情報はプロシージャ `WF_PURGE.TotalPerm()` をコールして情報を意図的に削除するまで、無期限に保存されます。

項目タイプの「維持」を「一時」に設定した場合は、維持日数（「*n*」）も指定する必要があります。「一時」項目タイプの各インスタンスの実行ステータス情報は、その完了日以後少なくとも *n* 日間は維持されます。*n* 日間維持された後は、`WF_PURGE` API のいずれかを使用して項目タイプのランタイム・ステータス情報を削除できます。8-114 ページの「[Workflow PURGE API](#)」を参照してください。

項目タイプの「維持」を「同期」に設定すると、その項目タイプのインスタンスは、項目キー `#SYNCH` が指定された状態で強制同期プロセスとして実行されます。強制同期プロセスは、単一 SQL セッション内で開始および終了し、データベース表に対して挿入または更新を行いません。強制同期プロセスではランタイム・ステータス情報が保持されないため、通常は、「同期」維持タイプのプロセスを削除する必要はありません。ただし、同期モードでプロセスを実行するときに、テストまたはデバッグを目的として一意の項目キーを使用した場合は、プロセス・インスタンスのランタイム・ステータス情報が保持されます。この情報を削除するには、項目タイプを「維持」から「一時」に変更してから、任意の `WF_PURGE` API を実行します。実行したら、項目タイプを「一時」から「同期」に戻します。8-15 ページの「[同期プロセス](#)」、[非同期プロセスおよび強制同期プロセス](#)」、8-114 ページの「[Workflow PURGE API](#)」および C-7 ページの「[パフォーマンス改善のためのページ](#)」を参照してください。

---

---

**注意：** Oracle Applications embedded Workflow を使用している場合、不要な項目タイプのランタイム・ステータス情報を削除するには、「ワークフローの不要ランタイム・データのページ」コンカレント・プログラムも使用する必要があります。このコンカレント・プログラムの実行ファイル名は `Oracle Workflow Purge Obsolete Data` で、短縮名は `FNDWFPR` です。8-122 ページの「[ワークフローの不要ランタイム・データのページ](#)」[コンカレント・プログラム](#)」を参照してください。

---

---

## 項目タイプのセレクト関数

項目タイプに複数の実行可能なプロセス・アクティビティが関連している場合、または今後関連付ける予定の場合は、特定の状況でどのプロセス・アクティビティを実行するかを判別

する PL/SQL 関数を定義します。たとえば、同じ項目タイプに 2 つの異なる購買承認申請プロセス・アクティビティが関連付けられている場合があります。Oracle Workflow で実行されるプロセスは、購買申請方法と依頼元に応じて異なる可能性があります。このような場合は、セレクト関数により、特定の状況に適切なプロセスが判別されます。

また、プロセスの実行中に SQL セッションが割込みされた場合に、必要に応じて項目タイプのコンテキスト情報をリセットできるように、セレクト関数を通常のコールバック関数に拡張することもできます。これは、Oracle Applications の使用例で、「通知の詳細」Web 画面から通知を表示し、その通知に関連付けられた他のフォームを起動する場合に特に重要です。Oracle Workflow では、Oracle Application Object Library 関数のセキュリティ・システムにフォームの起動が渡される前に、項目タイプに対するセレクト / コールバック関数が TEST\_CTX モードでコールされて、Oracle Applications コンテキストがテストされます。TEST\_CTX モードでは、セレクト / コールバック関数により、フォームの起動が適切かどうかの判断に必要なあらゆるロジックを実行できます。7-12 ページの「[項目タイプのセレクト関数またはコールバック関数の標準 API](#)」を参照してください。

## 外部文書の統合

文書は、組織運営に大きな影響力を持ちます。デジタル・メディアとワールド・ワイド・ウェブ (WWW) の急激な発展に伴い、印刷物以外のメディアなど様々な書式の電子文書が出現したため、組織はこれらの文書を管理する必要に迫られています。このような文書の情報は、管理および共有されて初めて価値を持ちます。

ワークフロー・プロセスでは、PL/SQL プロシージャにより生成された文書 (PL/SQL 文書または PL/SQL CLOB 文書) を添付できます。文書をワークフロー・プロセスに添付するには、事前定義済の項目属性または「文書」タイプのメッセージ属性で文書を参照します。4-3 ページの「[属性タイプ](#)」、4-8 ページの「[項目タイプまたはアクティビティ属性の定義](#)」および 4-32 ページの「[メッセージ属性の定義](#)」を参照してください。

PL/SQL 文書および PL/SQL CLOB 文書の場合、項目属性またはメッセージ属性の値は、その文書の生成に使用された PL/SQL パッケージおよびプロシージャの名前になります。PL/SQL プロシージャは、Oracle Workflow 標準インタフェースに従う必要があります。PL/SQL プロシージャにより生成される文書は、通知のテキスト内で表示されます。7-15 ページの「[PL/SQL および PL/SQL CLOB 文書の標準 API](#)」を参照してください。

## ▶ 項目タイプの作成

ナビゲータ・コントロールのプロパティ

項目タイプ | ルール | アクセス

内部名(I)

表示名(N)

説明(D)

維持(P)

日数(B)

セレクト(S)

OK キャンセル 適用(A) ヘルプ

1. まだデータ・ストアをオープンしていない場合は、「ファイル」メニューから「新規作成」を選択し、この新規項目タイプを定義する新規データ・ストアを作成します。次に「編集」メニューから「新規項目タイプ」を選択し、ナビゲータ・ツリー内で新規項目タイプを定義します。「項目タイプ」プロパティ画面が表示されます。
2. 各項目タイプには、すべて大文字の内部名（8 文字以内）があります。すべての Oracle Workflow API、SQL スクリプトおよび PL/SQL プロシージャでは、項目タイプの識別時に内部名が参照されます。

---

**注意：** 定義した項目タイプの内部名を更新するには、wfchitt.sql という特別な SQL スクリプトを使用する必要があります。このスクリプトは、設計時に項目タイプの内部名のエラーを訂正する場合にのみ使用してください。プロセスの実行中のインスタンスに関連する項目タイプの名前の変更には、使用しないでください。16-9 ページの「wfchitt.sql」を参照してください。

---



---

**注意：** 内部名には、コロン (:) や空白を使用しないでください。

---

3. 「表示名」には、変換可能で内部名より長くてわかりやすい名前を入力します。また、項目タイプの説明も入力できます。

4. 維持タイプとして「一時」または「永続」を指定します。「維持タイプ」を「一時」に設定した場合は、項目のインスタンスが終了してから、実行ステータス情報が削除されるまでの日数を指定します。4-4 ページの「[維持タイプ](#)」を参照してください。
5. 項目タイプに複数のワークフロー・プロセスが関連している場合や、今後関連付ける予定の場合は、`<package_name>.<procedure_name>` 構文を使用してセレクト関数を指定できます。セレクト関数は PL/SQL のストアド・プロシージャで、この項目タイプに対してワークフローが起動された場合に、ワークフロー・エンジンで実行される特定のプロセス定義を自動的に識別します。また、ワークフロー・エンジンにより新規データベース・セッションが確立されてアクティビティが実行されるたびに、コンテキスト情報がリセットされるように、セレクト関数を一般コールバック関数に拡張することもできます。7-12 ページの「[項目タイプのセレクト関数またはコールバック関数の標準 API](#)」を参照してください。
6. 「適用」を選択して変更内容を保存します。
7. 「ロール」タブを選択し、この項目タイプにアクセスできるロールを指定します。（この機能は、今後のリリースでサポートされます。）
8. 「アクセス」タブを選択し、この項目タイプのアクセス・レベルとカスタマイズ・レベルを設定します。4-16 ページの「[オブジェクトへのアクセス許可の設定](#)」を参照してください。
9. 「適用」を選択して変更内容を保存するか、「OK」を選択し、変更内容を保存してプロパティ画面を閉じるか、「キャンセル」を選択し、変更を取り消してプロパティ画面を閉じます。
10. 作成した項目タイプを表す 2 次ブランチがナビゲータ・ツリーに表示されます。ナビゲータ・ツリーで項目タイプをダブルクリックするか、項目タイプを選択して「編集」メニューから「プロパティ」を選択すると、いつでもこの項目タイプのプロパティを検討または編集できます。
11. プロセス内でグローバル変数として使用する項目タイプ属性を、必要な数を定義します。これらの項目タイプ属性を使用して、作成した関数、通知またはイベントの各アクティビティとの間で値を渡すことができます。4-8 ページの「[項目タイプまたはアクティビティ属性の定義](#)」を参照してください。

**関連項目：**

3-9 ページ「[プロパティ画面の「編集」ボタンの使用](#)」

▶ **項目タイプまたはアクティビティ属性の定義**

1. 項目タイプ属性を作成するには、ナビゲータ・ツリーで項目タイプを選択し、「編集」メニューから「新規属性」を選択します。

ナビゲータ・コントロールのプロパティ

属性 | アクセス |

項目タイプ 文書管理

内部名(N)

表示名(N)

説明(D)

タイプ(T) テキスト

長さ(L)

デフォルト

タイプ(V) 定数

値(V)

OK キャンセル 適用(Alt) ヘルプ

アクティビティ属性を作成するには、ナビゲータ・ツリーでアクティビティを選択し、「編集」メニューから「新規属性」を選択します。

ナビゲータ・コントロールのプロパティ

属性 |

回数 1回のみ

内部名(N)

表示名(N)

説明(D)

タイプ(T) テキスト

長さ(L)

デフォルト

タイプ(V) 定数

値(V)

OK キャンセル 適用(Alt) ヘルプ

どちらの場合も、「属性」プロパティ画面が表示されます。

2. 内部名をすべて大文字で空白を入れずに入力します。すべての Oracle Workflow API、SQL スクリプトおよび PL/SQL プロシージャでは、属性の識別時に内部名が参照されます。

---

**注意：** 定義した属性の内部名を更新するには、wfchita.sql および wfchacta.sql という特別な SQL スクリプトを使用する必要があります。これらのスクリプトは、設計時に属性の内部名のエラーを訂正する場合にのみ使用してください。プロセスの実行中のインスタンスに関連する属性の名前の変更には、使用しないでください。16-8 ページの「wfchita.sql」および 16-8 ページの「wfchacta.sql」を参照してください。

---

---

**注意：** 内部名には、コロン (:) や空白を使用しないでください。

---

3. 表示名を入力します。これは、ナビゲータ・ツリーに表示される名前です。
4. オプションの説明を入力します。
5. 属性のデータ型を選択します。アクティビティ属性を定義する場合であれば、フォーム、URL および文書のデータ型は関係ありません。
6. 属性のデータ型に応じて、次のデフォルト値の情報を指定します。
  - **テキスト：** テキスト属性の最大長と、オプションでデフォルトのテキスト文字列を指定します。
  - **数値：** オプションで、数値の書式マスクとデフォルト値を指定します。
  - **日付：** オプションで、日付の書式マスクとデフォルト値を指定します。
  - **選択肢：** 値を取り出す事前定義済の選択肢タイプを選択します。その選択肢タイプから、デフォルト値の選択肢コードを選択します。
  - **URL：** オプションで、ネットワーク位置への Universal Resource Locator (URL) を「デフォルト」の「値」フィールドに指定し、URL のフレーム・ターゲットを指定します。4-11 ページの「URL 属性の定義」を参照してください。

---

**注意：** 「フレーム・ターゲット」フィールドは、URL タイプのメッセージ属性のみに適用されます。項目タイプ属性やアクティビティ属性には使用されません。

---

- **フォーム：** この属性は、Oracle Applications embedded Workflow にのみ関係します。

「デフォルト」の「値」フィールドで、開発者フォーム機能名と引数文字列（フォーム機能のパラメータ）を指定します（いずれもオプション）。『Oracle Applications Developer's Guide』の「Overview of Menus and Function Security」および 4-12 ページの「**フォーム属性の定義**」を参照してください。

- **文書**: 文書を識別する文字列（オプション）を「デフォルト値」の「値」フィールドに入力します。4-13 ページの「**文書属性の定義**」を参照してください。

---

**注意**: 「フレーム・ターゲット」フィールドは、文書タイプの属性には適用されません。文書属性の「フレーム・ターゲット」フィールドは、今後使用する目的で確保されています。

---

- **ロール**: ロール名を指定します。5-25 ページの「**ロール**」を参照してください。
  - **属性**: 既存の項目タイプ属性のリストから、プロセス内で参照を維持する項目タイプ属性の名前を選択して指定します。
  - **イベント**: 項目タイプ属性を定義している場合は、イベント属性のデフォルト値を指定できません。アクティビティ属性を定義している場合は、イベント項目タイプ属性だけをデフォルト値として指定できます。
7. 項目タイプ属性の場合、オプションのデフォルト値は、入力するか値リストから選択した定数になります。ただし、この定数は、実行時にトークンが置換されるテキスト文字列の場合もあります。

アクティビティ属性の場合は、オプションのデフォルト値として定数または項目タイプ属性を指定できます。デフォルトで項目タイプ属性のすべての値を取得する場合は、「デフォルト」の「値」領域で「項目属性」を選択し、隣りにあるドロップ・ダウン・フィールドを使用して項目タイプ属性を選択します。ここで選択する項目タイプ属性は、アクティビティ自体が関連付けられているのと同じ項目タイプに関連付ける必要があります。また、アクティビティ属性と同じデータ型の項目タイプ属性を選択する必要があります。

---

**注意**: 「テキスト」タイプのアクティビティ属性には、すべての項目属性タイプとの互換性がありますが、他のアクティビティ属性タイプは、いずれも項目属性タイプと完全に一致させる必要があります。

---

---

**注意**: 「選択肢」タイプの属性の場合、デフォルト値はその選択肢タイプに属する選択肢コードにする必要があります。

---

8. 「適用」を選択して変更内容を保存するか、「OK」を選択し、変更内容を保存してプロパティ画面を閉じるか、「キャンセル」を選択し、変更を取り消してプロパティ画面を閉じます。



9. 項目タイプ属性を定義している場合は、「アクセス」タブを選択し、この属性を変更できるアクセス・レベルを設定します。アクティビティ属性では、その親アクティビティのアクセス / 保護レベルが想定されます。4-16 ページの「[オブジェクトへのアクセス許可の設定](#)」を参照してください。
10. 「適用」を選択して変更内容を保存します。
11. 作成した項目タイプ属性は、ナビゲータ・ツリーの「属性」のブランチの下に表示されます。定義した関数アクティビティ属性は、ナビゲータ・ツリーでその定義対象となったアクティビティの下に表示されます。ナビゲータ・ツリーで属性をダブルクリックするか、属性を選択して「編集」メニューから「プロパティ」を選択すると、いつでもその属性のプロパティを検討または編集できます。

---

**注意：** これらの属性がナビゲータ・ツリーに表示される順序は、これらの属性の取出し元となる値リストに表示される順序に関連しています。ナビゲータ・ツリーのドラッグ・アンド・ドロップ機能を使用するか、属性を選択して「編集」メニューから「属性を上へ移動」や「属性を下へ移動」を選択して、属性セット内での属性の順序を変更できます。

---

#### 関連項目：

3-9 ページ「[プロパティ画面の「編集」ボタンの使用](#)」

### ► URL 属性の定義

1. 「属性」プロパティ画面の「デフォルト」の「値」フィールドに、ネットワーク位置への Universal Resource Locator (URL) を指定します。URL には、定数または他の項目属性から戻される値を使用できます。
2. URL には、テキスト文字列の引数文字列を指定できます。また、URL タイプのメッセージ属性を定義している場合は、指定した引数文字列（トークン）が他のメッセージ属性に置換されます。トークンから置換されるメッセージ属性では、定数値を指定したり、項目タイプ属性から返された値を参照したりできます。4-32 ページの「[メッセージ属性の定義](#)」および 4-39 ページの「[属性のトークン置換](#)」を参照してください。

引数文字列内の他のメッセージ属性についてトークンを置換するには、メッセージ属性を次のように指定します。

```
-&message_attr-
```

たとえば、次の文字列は arg1 および arg2 という 2 つの引数を持つ URL を表し、その引数が実行時にはメッセージ属性の値 msgattr1 および msgattr2 でそれぞれトークンを置き換えます。

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

---

---

**注意：** URL タイプのメッセージ属性を定義する場合、引数文字列に `-&#NID-` という特殊トークンを含めることもできます。このトークンはランタイム通知の通知 ID で置換されます。

---

---

3. URL 属性に引数文字列が含まれている場合、次の制限事項を厳守してください。
  - その引数文字列について、文書タイプの別の項目属性でトークンを置換することはできません。
  - その引数文字列について、別のフォーム属性または URL 属性でトークンを置換することはできませんが、他の属性の引数文字列についてそれ以上トークンを置換することはできません。
4. 日付と時間を引数として URL に渡す場合は、`TO_CHAR` を使用して文字列を `YYYY/MM/DD+HH24:MI:SS` の書式にします。同様に、URL がコールする関数の中でも、`TO_DATE` を使用して対応する書式変換を行う必要があります。このような書式設定が必要なのは、マルチバイトのデータベースでは、書式 `DD-MON-YYYY` のうち `MON`（月）に当たる部分が URL 間で使用できない値に変換される可能性があるためです。
5. 操作の完了後に、「OK」を選択します。

#### ▶ フォーム属性の定義

1. フォームの「属性」プロパティ画面の「デフォルト」の「値」フィールドで、開発者フォーム機能名とオプションで引数文字列（フォーム機能のパラメータ）を指定します。
2. デフォルト値は、次の書式で入力してください。

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

`argN` の値は、引用符（"）で囲んだテキスト文字列か、または次のいずれかの方法で別の項目タイプ属性により置換されるトークンでもかまいません。`&item_attr` は項目タイプ属性の内部名を表します。

- `argN="&item_attr"`
- `argN="Value &item_attr"`

**参照：** 4-39 ページの「[属性のトークン置換](#)」を参照してください。

---

---

**注意：** 「フォーム」タイプのメッセージ属性を定義する場合、引数文字列に `&#NID` という特殊トークンを含めることもできます。このトークンはランタイム通知の通知 ID で置換されます。

---

---

3. フォーム属性に引数文字列が含まれている場合は、次の制限事項を厳守してください。

- `argN`の値について、「文書」タイプの別の項目属性でトークンを置換することはできません。
- その引数文字列の値について別のフォーム属性またはURL属性でトークンを置換することはできますが、他の属性の引数文字列についてそれ以上トークンを置換することはできません。

4. 操作の完了後に、「OK」を選択します。

### ► 文書属性の定義

1. 「属性」プロパティ画面の「デフォルト」の「値」フィールドに、文書を識別する文字列を入力します。

文書属性について、次のタイプの文書を識別できます。

- PL/SQL 文書
- PL/SQL CLOB 文書

2. PL/SQL 文書は、データベースのデータを文字列として表すもので、PL/SQL プロシージャにより生成されます。PL/SQL 文書のデフォルト値は、次のように指定します。

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

<procedure> は、PL/SQL パッケージおよびプロシージャの名前をピリオド (.) で区切ったもので置き換えます。<document\_identifier> は、プロシージャに直接渡される PL/SQL 引数文字列に置き換えます。引数文字列では、文書を識別する必要があります。

---

**注意：** PL/SQL プロシージャは、標準 API 書式に準じてください。7-15 ページの「[PL/SQL および PL/SQL CLOB 文書の標準 API](#)」を参照してください。

---

たとえば、次の文字列はプロシージャ `po_wf.show_req` から生成された PL/SQL 文書 `po_req:2034` を表しています。

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

---

**注意：** PL/SQL 文書の最大データ長は、32KB です。文書が 32KB を超える場合は、かわりに PL/SQL CLOB 文書を使用してデータを格納する必要があります。

---

3. PL/SQL CLOB 文書は、データベースのデータをキャラクタ・ラージ・オブジェクト (CLOB) として表すもので、PL/SQL プロシージャから生成されます。PL/SQL CLOB 文書のデフォルト値は、次のように指定します。

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

<procedure> は、PL/SQL パッケージおよびプロシージャの名前をピリオド (.) で区切ったもので置き換えます。<document\_identifier> は、プロシージャに直接渡される PL/SQL 引数文字列に置き換えます。引数文字列では、文書を識別する必要があります。

---

---

**注意：** PL/SQL プロシージャは、標準 API 書式に準じてください。7-15 ページの「[PL/SQL および PL/SQL CLOB 文書の標準 API](#)」を参照してください。

---

---

たとえば、次の文字列はプロシージャ `po_wf.show_req_clob` から生成された PL/SQL CLOB 文書 `po_req:2036` を表しています。

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

PL/SQL CLOB 文書では、メッセージ属性のトークンを置換できません。CLOB の内容は、PL/SQL プロシージャで生成された状態でメッセージ本文に印刷されます。

- CLOB 内でトークンを使用しないでください。
  - 必要な書式は PL/SQL プロシージャで設定してください。
4. PL/SQL または PL/SQL CLOB 文書の文書識別子を動的に生成する場合は、文書識別子について他の項目タイプ属性でトークンを置換できます。項目属性の名前は太文字で記述し、コロンで区切る必要があります。4-39 ページの「[属性のトークン置換](#)」を参照してください。

たとえば、次のようになります。

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

---

---

**注意：** 「文書」タイプのメッセージ属性を定義する場合、引数文字列に `&#NID` という特殊トークンを含めることもできます。このトークンはランタイム通知の通知 ID で置換されます。

---

---

5. 操作の完了後に、「OK」を選択します。

### ▶ 項目タイプのコピー

1. ナビゲータ・ツリーで、コピーする項目タイプを選択します。
2. マウスの左ボタンを押しながら、項目タイプをコピー先のデータ・ストアまたはワークスペースへドラッグします。

「編集」メニューの「コピー」および「貼り付け」コマンドも使用できます。

3. この項目タイプを同じデータ・ストアにコピーすると、「項目タイプ」プロパティ画面に、その項目タイプの新規の内部名と表示名の入力を求めるプロンプトが表示されます。これは、項目タイプには一意の内部名と表示名を与える必要があるためです。操作の完了後に「OK」を選択します。

項目タイプをコピーすると、その項目タイプに関連付けられているコンポーネントもすべてコピーされるため注意してください。ほとんどのコンポーネントの内部名と表示名も一意である必要があるため、これらのコンポーネントに関しても、そのプロパティ画面に内部名と表示名の更新を求めるプロンプトが表示されます。

4. 同じ項目タイプの旧バージョンがすでに存在するデータ・ストアに項目タイプをコピーすると、コピー先データ・ストア内の項目タイプの旧バージョンは、コピー元の項目タイプのバージョンの内容で更新されます。

---

**注意：** 新規のストアに複数の項目タイプをドラッグする場合は、その順序が重要になります。たとえば、ある項目タイプが標準項目タイプのオブジェクトを参照する場合を考えます。その項目タイプと標準項目タイプを新規データ・ストアにコピーする場合は、標準項目タイプを新規データ・ストアにドラッグしてから、他方の項目タイプをドラッグする必要があります。そうしないと、その項目タイプには標準項目タイプに対する未解決の参照が含まれることになります。

---

## ► 属性のコピー

1. ナビゲータ・ツリーで、コピーする属性を選択します。
2. マウスの左ボタンを押しながら、属性をコピー先となるコンポーネントのブランチヘドドラッグします。
3. 属性を同じ項目タイプに関連付けられているコンポーネントにコピーすると、その属性のプロパティ画面が表示されます。

その属性の一意の新しい内部名と表示名を入力します。

操作の完了後に「OK」を選択します。

---

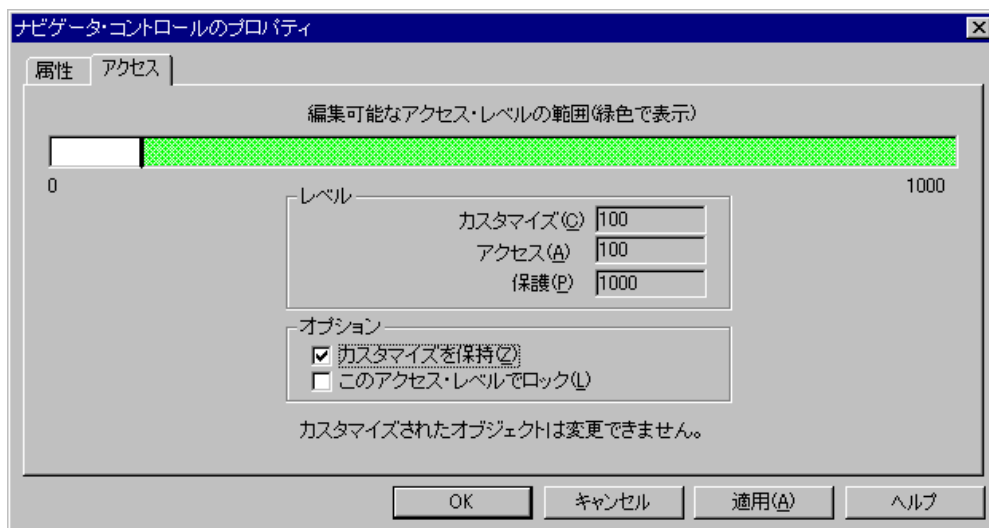
**注意：** 「編集」メニューの「コピー」および「貼り付け」オプションも使用できます。

---

## 関連項目：

3-9 ページ [「プロパティ画面の「編集」ボタンの使用」](#)

## オブジェクトへのアクセス許可の設定



「アクセス」タブの「編集可能なアクセス・レベルの範囲」インジケータ・バーには、オブジェクトを編集できるアクセス・レベルの範囲が相対表示されます。影付きの部分はオブジェクトを編集できるアクセス・レベルを表し、縦棒は現行のアクセス・レベルを表します。2-95 ページの「[Oracle Workflow のアクセス保護の概要](#)」を参照してください。

インジケータ・バーは、影付きの緑色の実線または影付きの緑色の実線と灰色の交差線の組合せとして表示できます。「ヘルプ」メニューの「Oracle Workflow Builder のバージョン情報」ダイアログ・ボックスで、「カスタマイズされたオブジェクトの変更を可能にする」チェック・ボックスのオン / オフによって、表示状態を次のように切り替えることができます。

- オン： 編集可能なアクセス・レベルの範囲を、緑色の実線と灰色の交差線による部分の組合せとして表示できます。灰色の交差線が示すレベルは、通常はカスタマイズ済のオブジェクトを変更できませんが、Oracle Workflow Builder がアップロード・モードで動作しているため変更可能になっているレベルを表します。アップロード・モードは、変更するためにアクセスした保護対象のオブジェクトのみでなく、以前にカスタマイズしたオブジェクトも上書きして、Oracle Workflow Builder で編集結果を保存できることを意味します。
- オフ： 編集可能なアクセス・レベルの範囲は、緑色の実線部分として表示されます。これは、作業結果を保存すると、Oracle Workflow Builder はアップグレード・モードで動作することを示します。この場合、変更のためにアクセスした保護対象のオブジェクトの編集内容のみが保存され、以前にカスタマイズ済のオブジェクトはそのまま残ります。

この2つのモードは、ワークフロー定義ローダーのプログラムのアップグレードおよびアップロード動作と同じです。4-17 ページの「[オブジェクトのアクセス・レベルの設定](#)」および 2-101 ページの「[ワークフロー定義ローダーの使用](#)」を参照してください。

## ▶ オブジェクトのアクセス・レベルの設定

1. プロパティ画面の「アクセス」タブを選択します。
2. 「オプション」フィールドの「カスタマイズを保持」および「このアクセス・レベルでロック」チェック・ボックスを使用して、このオブジェクトを変更できるアクセス・レベルを定義します。このフィールドでオンにするオプションは、「レベル」フィールドに表示される値に直接影響します。

次の表は、各オプションの様々な組合せをオンにした場合に、オブジェクトのカスタマイズ・レベルと保護レベルにどのように影響するかを示しています。この表は、オプションのアクセス・レベルが 100 に設定されていることを前提としています。

表 4-1

オンになっている チェック・ボックス	結果のレベル	オブジェクトの変更が許可されるレベル
なし: オブジェクトはいつでもだれでも更新できます。	カスタマイズ = 0 アクセス = 100 保護 = 1000	オブジェクトはどのレベル (0 ~ 1000) でも更新が可能です。
<b>カスタマイズを保持:</b> ワークフロー定義のアップグレード中は、カスタマイズされたオブジェクトの上書きを禁止します。	カスタマイズ = 100、 アクセス = 100、 保護 = 1000	アクセス・レベルが 100 ~ 1000 の場合にのみ、オブジェクトの更新が可能です。「カスタマイズされたオブジェクトの変更を可能にする」チェック・ボックスがオンの場合は、インジケータ・バーの灰色の交差線で表されるアクセス・レベル 0 ~ 99 でも、カスタマイズされたオブジェクトを更新できます。
<b>このアクセス・レベルでロック:</b> オブジェクトを現在のアクセス・レベルで保護し、オブジェクトのカスタマイズを許可しません。	カスタマイズ = 0、 アクセス = 100、 保護 = 100	アクセス・レベルが 0 ~ 100 の場合にのみ、オブジェクトの更新が可能です。
<b>両方:</b> オブジェクトは、それが保護されているレベルでのみ更新できます。	カスタマイズ = 100、 アクセス = 100、 保護 = 100	アクセス・レベルが 100 以外の場合には、オブジェクトを更新できません。「カスタマイズされたオブジェクトの変更を可能にする」チェック・ボックスがオンの場合は、インジケータ・バーの灰色の交差線で表されるアクセス・レベル 0 ~ 99 でも、カスタマイズされたオブジェクトを更新できます。

3. 「適用」 ボタンを選択して変更内容を保存します。

---

**注意：** オブジェクトは、ナビゲータ・ツリーに赤い小型のロック付きアイコンとして表示されます。これは、オブジェクトの編集が許可されていないアクセス・レベルで操作中の場合、つまり、アクセス・レベルが「編集可能なアクセス・レベルの範囲」インジケータ・バーのうち白い部分に含まれているときは、そのオブジェクトが読み取り専用であることを示します。


---

## 選択肢タイプ

選択肢タイプは、静的な値リストです。このリストは、アクティビティ、項目タイプ、メッセージまたはアクティビティの属性から参照できます。たとえば、アクティビティは選択肢タイプで利用できる結果の値を参照し、メッセージ属性は選択肢タイプを参照して通知の実行者が使用可能な応答のリストを提供します。

選択肢タイプを定義してから、それを特定の項目タイプに関連付けます。ただし、選択肢タイプは項目タイプに固有のものではありません。このため、アクティビティまたは属性を作成する場合は、選択肢タイプが関連付けられている項目タイプに関係なく、現行のデータ・ストアのすべての選択肢タイプを参照できます。4-18 ページの「[選択肢タイプの作成](#)」を参照してください。

### ▶ 選択肢タイプの作成



The screenshot shows a Windows-style dialog box titled "ナビゲータ・コントロールのプロパティ" (Navigator Control Properties). It has two tabs: "選択肢タイプ" (Choice Type) and "アクセス" (Access). The "選択肢タイプ" tab is active. Inside the dialog, there are three input fields: "内部名(I)" (Internal Name), "表示名(N)" (Display Name), and "説明(D)" (Description). The "説明(D)" field is a larger text area. At the bottom of the dialog, there are four buttons: "OK", "キャンセル" (Cancel), "適用(A)" (Apply), and "ヘルプ" (Help).

1. ナビゲータ・ツリーから項目タイプを選択し、「編集」メニューから「新規選択肢タイプ」を選択します。「選択肢タイプ」プロパティ画面が表示されます。



2. 選択肢タイプには、すべて大文字で前後に空白を含まない「内部名」と、変換可能な「表示名」が付いています。すべての Oracle Workflow API、SQL スクリプトおよび PL/SQL プロシージャでは、選択肢タイプの識別時に内部名が参照されます。

---

**注意：** 定義した選択肢タイプの内部名を更新するには、wfchlut.sql という特別な SQL スクリプトを使用する必要があります。このスクリプトは、設計時に選択肢タイプの内部名のエラーを訂正する場合にのみ使用してください。プロセスの実行中のインスタンスに関連する選択肢タイプの名前の変更には、使用しないでください。16-9 ページの「wfchlut.sql」を参照してください。

---

---

**注意：** 内部名には、コロン (:) や空白を使用しないでください。

---

選択肢タイプには任意で説明を加えることもできます。

3. 「アクセス」タブを選択し、この選択肢タイプを変更できるアクセス・レベルを設定します。4-16 ページの「[オブジェクトへのアクセス許可の設定](#)」を参照してください。
4. 「適用」を選択して変更内容を保存するか、「OK」を選択し、変更内容を保存してプロパティ画面を閉じるか、「キャンセル」を選択し、変更を取り消してプロパティ画面を閉じます。
5. 定義した選択肢タイプが、ナビゲータ・ツリーの「選択肢タイプ」のブランチの下に表示されます。ナビゲータ・ツリーで選択肢タイプをダブルクリックするか、選択肢タイプを選択して「編集」メニューから「プロパティ」を選択すると、いつでもこの選択肢タイプのプロパティを検討または編集できます。
6. 作成した選択肢タイプの選択肢コードを定義します。4-20 ページの「[選択肢タイプの選択肢コードの作成](#)」を参照してください。

➤ 選択肢タイプの選択肢コードの作成

1. ナビゲータ・ツリーから選択肢タイプを選択し、「編集」メニューから「新規選択肢コード」を選択します。「選択肢コード」プロパティ画面が表示されます。
2. 選択肢コードの「内部名」と「表示名」を入力します。内部名の前後に空白を入れることはできません。オプションで説明を入力することもできます。すべての Oracle Workflow API、SQL スクリプトおよび PL/SQL プロシージャでは、選択肢コードの識別時に内部名が参照されます。

---

**注意：** 定義した選択肢コードの内部名を更新するには、wfchluc.sql という特別な SQL スクリプトを使用する必要があります。このスクリプトは、設計時に選択肢コードの内部名のエラーを訂正する場合にのみ使用してください。プロセスの実行中のインスタンスに関連する選択肢コードの名前の変更には、使用しないでください。16-9 ページの「[wfchluc.sql](#)」を参照してください。

---



---

**注意：** 内部名には、コロン (:) や空白を使用しないでください。

---

3. 「適用」を選択して変更内容を保存するか、「OK」を選択し、変更内容を保存してプロパティ画面を閉じるか、「キャンセル」を選択し、変更を取り消してプロパティ画面を閉じます。

4. 定義した選択肢コードが、ナビゲータ・ツリーで作成した「選択肢タイプ」のブランチの下に表示されます。ナビゲータ・ツリーで選択肢コードをダブルクリックするか、選択肢コードを選択して「編集」メニューから「プロパティ」を選択すると、いつでもこの選択肢コードのプロパティを検討または編集できます。
5. 特定の選択肢タイプの選択肢コードを追加作成する場合は、手順 1 を繰り返します。

### ▶ 選択肢タイプのコピー

1. ナビゲータ・ツリーで、コピーする選択肢タイプを選択します。
2. 「編集」メニューの「コピー」および「貼り付け」オプションを使用して、選択肢タイプを任意の項目タイプにコピー・アンド・ペーストします。または、選択肢タイプを任意の項目タイプにドラッグ・アンド・ドロップします。
  - 選択肢タイプを同じ項目タイプに再度コピーした場合、または別のデータ・ストアの項目タイプにコピーしたときにその選択肢タイプがいずれかの項目タイプにすでに存在する場合は、プロパティ画面が表示されます。新しい選択肢タイプに対して一意の内部名と表示名を入力してください。操作の完了後に「OK」を選択します。
  - 選択肢タイプを別のデータ・ストアの項目タイプにコピーしたときに、その選択肢タイプがそのデータ・ストアのいずれかの項目タイプに存在しない場合は、元の名前と同じ内部名および表示名で新しい選択肢タイプがコピーされます。新しい名前を入力する必要はありません。

---

**注意：** 選択肢タイプをコピーすると、それに割り当てられている選択肢コードもコピーされます。

---

---

**注意：** 「コピー」および「貼り付け」オプションを使用して、選択肢タイプを同じデータ・ストアの別の項目タイプにコピーすることはできません。ただし、ドラッグ・アンド・ドロップすることはできます。この場合、選択肢タイプは新しい項目タイプに移動します。

---

### ▶ 選択肢コードのコピー

1. ナビゲータ・ツリーで、コピーする選択肢コードを選択します。
2. マウスの左ボタンを押しながら、選択肢コードをコピー先の選択肢タイプまでドラッグします。
3. 選択肢コードを同じ選択肢タイプに、またはこのコードがすでに存在する選択肢タイプへドラッグすると、マウス・ボタンを放したときに、プロパティ画面が表示されるため、新規の選択肢コードの一意の内部名と表示名を入力します。操作の完了後に「OK」を選択します。

---

---

**注意：**「編集」メニューの「コピー」および「貼り付け」オプションも使用できます。

---

---

## メッセージ

ナビゲータ・ツリーの「メッセージ」のブランチには、現行の項目タイプに使用可能なワークフロー・メッセージがすべて表示されます。

通知アクティビティがワークフロー・プロセスのロールに送信するのがメッセージです。メッセージでは、ユーザーに応答やアクションを求めるプロンプトを表示して、プロセスの次のアクティビティを決定するための情報を得ることができます。ワークフロー・メッセージの受信者は実行者と呼ばれます。

各メッセージは、特定の項目タイプに関連付けられます。これにより、メッセージでは、その項目タイプの属性を、そのメッセージが送信された場合の実行時に置き換えられるトークンとして参照できます。

メッセージを定義するときに、そのメッセージで受信者に特別な応答値を要求するよう指定できます。その場合、ワークフロー・エンジンでは、この応答値を使用して、プロセス内の次の適格アクティビティに分岐する方法が決定されます。メッセージの件名と本文に項目タイプ属性を参照するメッセージ属性トークンを含めて、状況依存のコンテンツを持つメッセージを作成することができます。メッセージ関数を使用して、書式化されたメッセージ属性の表または通知履歴の表をメッセージの本文に含めることができます。文書全体または URL を表すメッセージ属性を、通知メッセージに添付することも可能です。さらに、メッセージに固有の応答セクションを生成するメッセージ属性も作成できます。

メッセージを「通知」のブランチにドラッグし、そのメッセージを送信する新しい通知アクティビティを作成できます。また、メッセージを既存の通知アクティビティに直接ドラッグして、そのアクティビティで送られるメッセージを更新できます。

### メッセージの結果

通知アクティビティのためのメッセージを作成する場合は、通知アクティビティの結果タイプが指定されているかどうかに注意する必要があります。結果タイプが指定されている場合は、通知アクティビティの結果として解釈される特別な応答を受信者に要求するようにメッセージを作成してください。ワークフロー・エンジンでは、その結果を使用して、プロセス内の次の「適格」アクティビティに分岐する方法が決定されます。

この特別な応答を求めるメッセージを作成するには、メッセージのプロパティ画面の「結果」タブに必要事項を入力します。入力した情報に基づいて、**RESULT** という内部名を持つ特別な応答メッセージ属性がそのメッセージに作成されます。**RESULT** メッセージ属性はデータ型が「選択肢」であり、通知アクティビティの結果タイプと同じ選択肢タイプに設定されている必要があります。これにより、通知の実行者は、通知アクティビティで予測される結果のリストと合致する応答値候補のリストから選択できます。4-23 ページの「[送信および応答メッセージ属性](#)」を参照してください。

## 送信および応答メッセージ属性

メッセージの作成後に、そのメッセージに関して必要な数だけメッセージ属性を定義できます。メッセージ属性は、ナビゲータ・ツリーでメッセージの下に表示されます。

メッセージ属性のソース（送信または応答）により、メッセージ属性の使用方法が決まります。「送信」のソースでメッセージ属性を定義する場合は、トークンの置換のためにメッセージの件名または本文（あるいはその両方）にそのメッセージ属性を埋め込むことができます。さらに、通知の送信時にはメッセージにメッセージ属性を添付できます。

各メッセージ属性には、特定のデータ型があります。「送信」メッセージ属性の値は、定数または同じデータ型の項目タイプ属性により戻される値です。トークンの置換のために、メッセージ属性をメッセージの件名または本文に埋め込むには、件名またはテキスト本文内で書式 `&MESGATTR` を使用して、メッセージ属性の内部名を指定します。

---

---

**注意：**「文書」メッセージ属性は件名内でトークンの置換ができないため、メッセージの件名には「文書」タイプのメッセージ属性を埋め込まないでください。件名に埋め込まれた「文書」メッセージ属性は無視されます。

ただし、「文書」メッセージ属性は、トークンの置換のためにメッセージ本文に埋め込むことができます。

---

---

「応答」ソースを使用して定義されたメッセージ属性は、メッセージの応答セクションの構成要素となります。「応答」メッセージ属性には、受信者に対して応答入力を求めるプロンプトを表示するインストラクションが用意されています。「応答」メッセージ属性を定義する場合は、属性のデータ型を指定する必要があります。また、オプションで応答のデフォルト値を指定することもできます。デフォルト値には、定数または同じデータ型の項目タイプ属性から戻される値を使用できます。

**参照：**C-4 ページ「[メッセージ属性](#)」

## #HIDE\_REASSIGN 属性

メッセージ属性または通知属性に内部名 `#HIDE_REASSIGN` を指定すれば、「通知の詳細」Web 画面の「再割当て」ボタンを非表示にできます。ユーザーが「ワークリスト」Web 画面で通知を表示すると、デフォルトで「通知の詳細」画面の応答フレームに「再割当て」ボタンが表示されます。ユーザーが通知を再割当てしないようにする場合は、`#HIDE_REASSIGN` 属性を追加して、「再割当て」ボタンを表示するか非表示にするかを制御できます。

`#HIDE_REASSIGN` 属性は、テキスト・タイプにする必要があります。「再割当て」ボタンを非表示にするには、この属性の値を `Y` に設定し、表示するには `N` に設定します。

- 特定のメッセージを使用する通知の「再割当て」ボタンを常に非表示にする場合は、値 `Y` を定数として指定します。

- 特定の場合にのみ「再割当て」ボタンを非表示にするには、値として項目タイプ属性を指定します。次に、このボタンを表示するかどうかを実行時に動的に決定し、それぞれ項目タイプ属性を Y または N に設定するロジックを、ワークフロー・プロセスに組み込みます。

### 関連項目：

10-13 ページ [「他のユーザーへの通知の再割当て」](#)

## #FROM\_ROLE 属性

メッセージ属性に内部名 #FROM\_ROLE を指定すれば、ロール（通知ソース）を指定できます。たとえば、購買申請を送信したことを承認者に通知する場合、購買申請作成者をメッセージの「From Role」として設定します。この属性を持つ通知が再割当てされた場合、新しい受信者への通知の送信時に「From Role」が元の受信者に自動的に設定されます。

各通知の「From Role」は、通知の確認や応答に関する詳細情報として、「ワークリスト」Web 画面に表示されます。また、「通知の検索」画面を使用すれば、「From Role」を基準にして通知を検索できます。

メッセージに対して「From Role」を指定するには、次のプロパティを使用してメッセージ属性を作成します。

- 内部名： #FROM\_ROLE
- 表示名： From Role
- 説明： From Role
- タイプ： ロール
- ソース： 送信
- デフォルトのタイプ： 項目属性
- デフォルト値： メッセージの「From Role」として設定する値を保持する項目属性

### 関連項目：

10-17 ページ [「ワークリストの通知の表示」](#)

10-15 ページ [「通知の検索」](#)

## WF\_NOTIFICATION() メッセージ関数

メッセージ属性トークンの他に、WF\_NOTIFICATION() という特別なメッセージ関数を使用して、状況依存のコンテンツをメッセージの本文に追加することもできます。

WF\_NOTIFICATION() 関数では、指定したパラメータに応じてメッセージ属性の表または通知履歴の表を生成することができます。表は、Oracle Workflow 標準形式で作成されます。



### メッセージ属性表

メッセージ属性の表をメッセージの本文に含めるには、ATTRS オプションの後にメッセージ属性の内部名をカンマで区切って指定して、WF\_NOTIFICATION() をコールします。書式は、次のとおりです。

WF\_NOTIFICATION (ATTRS, <attribute1>, <attribute2>, <attribute3>, ...)

**注意：** WF\_NOTIFICATION() のコールには、スペースや改行を含めないでください。リスト内でパラメータを区切るときはカンマのみを使用してください。

メッセージ属性表には、WF\_NOTIFICATION() コールにリストされている各メッセージ属性の行が格納され、各属性の表示名および値が示されます。

## 通知履歴表

通知履歴表をメッセージの本文に含めるには、次のように **HISTORY** オプションを指定して **WF\_NOTIFICATION()** をコールします。

**WF\_NOTIFICATION(HISTORY)**

通知履歴表には、プロセス内ですでに実行された通知アクティビティの行、およびプロセスの初期送信の行が格納されます。たとえば、複数の承認者に送信される購買承認申請通知の場合、通知履歴表には通知が送信される各承認者の行およびプロセス所有者の行が格納されます。

通知履歴表には、次の列が含まれます。

- 順序： 通知アクティビティの実行順序。プロセス所有者がプロセスを初期送信すると、ゼロ (0) に初期化されます。
- 対象者： 通知の受信者またはプロセス所有者。
- 処理： 通知に対して受信者が行った応答処理。
- 日付： 通知の日付。
- 備考： 受信者からの追加メモ。受信者が通知履歴表のメモを追加できるようにするには、内部名が **WF\_NOTE** の「応答」メッセージ属性を作成する必要があります。

次のプロパティを持つメッセージ属性を定義します。

- 内部名： **WF\_NOTE**
- 表示名： 追加メモ
- 説明： 追加メモ
- タイプ： テキスト
- ソース： 応答

ソースを「応答」として定義した **WF\_NOTE** 属性は、通知応答セクションに表示され、受信者は通知に応答するときにメモを入力できます。**WF\_NOTIFICATION()** 関数は、**WF\_NOTE** 属性に格納されているメモ・テキストを取得し、それを通知履歴表に表示します。

受信者がメモを入力しなかった場合、または **WF\_NOTE** メッセージ属性が通知に定義されていなかった場合、通知履歴表の「備考」列は空白のままになります。

---

**注意：** プロセスの送信時にプロセス所有者が通知履歴表にメモを追加することはできません。通知に応答するときにメモを追加できるのは、通知の受信者のみです。

---



## ▶ メッセージの作成

ナビゲータ・コントロールのプロパティ

メッセージ | 本文 | ロール | アクセス | 結果

内部名(N)

表示名(N)

説明(D)

優先度(P) 標準

OK キャンセル 適用(A) ヘルプ

1. ナビゲータ・ツリーで、メッセージを作成する項目タイプを選択し、「編集」メニューから「新規メッセージ」を選択します。「メッセージ」プロパティ画面が表示されます。
2. メッセージの内部名をすべて大文字で空白を入れずに入力し、表示名を指定します。オプションで説明を入力することもできます。すべての Oracle Workflow API、SQL スクリプトおよび PL/SQL プロシージャでは、メッセージの識別時に内部名が参照されます。

---

**注意：** 定義したメッセージの内部名を更新するには、wfchmsg.sql という特別な SQL スクリプトを使用する必要があります。このスクリプトは、設計時にメッセージの内部名のエラーを訂正する場合にのみ使用してください。プロセスの実行中のインスタンスに関連するメッセージの名前の変更には、使用しないでください。16-10 ページの「wfchmsg.sql」を参照してください。

---



---

**注意：** 内部名には、コロン (:) や空白を使用しないでください。

---

3. メッセージのデフォルトの優先度に「高」、「標準」および「低」を選択します。これは、メッセージの優先度を宛先に伝えるためのものです。メッセージの処理や配信には影響しません。

---

**注意：** このメッセージを通知アクティビティに割り当て、通知アクティビティをノードとしてプロセス・ダイアグラムに組み込むと、このデフォルト優先度を固定の、または実行時に動的に決定される新規の優先度で上書きできます。5-8 ページの「プロセスのノードの定義」を参照してください。

---

---

**注意：** Oracle Workflow の旧バージョンでは、メッセージの優先度は 1（高）～ 99（低）の数値で表されていました。このリリースの Oracle Workflow では、旧バージョンで定義されていたすべてのメッセージ定義の優先度が 1～33= 高、34～66= 標準、67～99= 低として自動的に変換されます。

---

4. 「適用」を選択して変更内容を保存します。



5. 「本文」タブを選択し、メッセージの「本文」プロパティ画面を表示します。
6. 「件名」には、「メッセージ」タブで入力した表示名がデフォルト値として取得されます。デフォルトの件名を選択するか、またはメッセージの新しい件名を入力します。件名には、メッセージの配信時にトークンをランタイム値で置き換えるメッセージ属性を含めることができます。件名にメッセージ属性を含めるには、アンパサンド (&) に続けてメッセージ属性の内部名を指定します。4-23 ページの「送信および応答メッセージ属性」および 4-32 ページの「メッセージ属性の定義」を参照してください。

---

---

**提案：** メッセージ属性には、参照先の項目タイプ属性と同じ名前を付けると、わかりやすくなります。

---

---

7. 「テキスト本文」フィールドに、プレーン・テキストのメッセージ本文を入力します。省略記号「...」ボタンを選択すると、「件名」および「テキスト本文」フィールドの表示を別のウィンドウに展開できます。

Oracle Workflow では、「テキスト本文」フィールドに入力した内容を使用して、通知メッセージのプレーン・テキスト版が生成されます。プレーン・テキスト・メッセージは、プレーン・テキスト・メッセージを表示する電子メール・ソフトウェアで表示できます。

---

---

**注意：** 「テキスト本文」フィールドには、プレーン・テキストのメッセージ本文を入力してください。「テキスト本文」が空の場合、プレーン・テキスト用電子メール・ソフトウェアでメッセージ表示したときに通知が空になります。

---

---

8. 「HTML Body」タブを選択して内容を入力するか、「インポート」を選択して .HTM または .HTML ファイルから内容をインポートすると、オプションで「HTML Body」フィールドに HTML 形式のメッセージ本文を入力できます。また、省略記号「...」ボタンを選択すると、「件名」および「HTML Body」フィールドの表示を別のウィンドウに展開できます。

---

---

**注意：** HTML メッセージ本文を入力またはインポートするときには、<Body>...</Body> の HTML タグを含める必要はありません。これらのタグを使用すると、Oracle Workflow ではタグで囲まれた内容が抽出され、HTML メッセージ本文として使用されます。そのため、<Body> タグより前にある HTML タグや内容は無視されることになります。

---

---

---

---

**注意：** Oracle Workflow Builder では、メッセージ本文の HTML 形式は検証されません。

---

---

Oracle Workflow では、「HTML Body」フィールドに入力した内容を使用して、通知メッセージの HTML 形式版が生成されます。HTML 形式の通知メッセージは、「通知の詳細」Web 画面から、または HTML 形式のメッセージや HTML 形式のメッセージ添付ファイルを表示する電子メール・ソフトウェアから表示できます。

---

---

**注意：**「HTML Body」が空の場合、Oracle Workflow では「テキスト本文」に入力されたメッセージ本文を使用して通知メッセージが生成されます。この場合は、`<pre>...</pre>` の HTML タグ間にプレーン・テキストが挿入されます。

---

---

---

---

**注意：** Oracle Workflow では、HTML メッセージ本文中のアイコンおよび画像ファイルへの参照が完全にはサポートされていません。使用中の Web サーバーではこれらのファイル位置を検出して「通知の詳細」 Web 画面に正しく表示できる場合でも、通知メーカーやサード・パーティの電子メール・アプリケーションでは、ユーザーが電子メール通知の HTML 版を表示するときに、アイコンまたは画像ファイルの位置を識別できません。

---

---

9. テキストまたは HTML 本文にメッセージ属性を埋め込むことができます。Oracle Workflow トークンにより、メッセージ属性は通知の配信時にランタイム値で置き換えられます。メッセージ属性を埋め込むには、「&」に続けてメッセージ属性の内部名を指定します。

---

---

**注意：** メッセージ本文中のテキストは、4000 バイト以内にする必要があります。テキストにトークンの置換のためのメッセージ属性を挿入すると、最終的なメッセージ本文は最大で 32000 バイトまで増えることがあります。

---

---

---

---

**注意：** メッセージの件名または本文には、`&#NID` という特殊トークンも挿入できます。このトークンは、ランタイム通知の通知 ID に置き換えられます。

---

---

また、メッセージ関数 `WF_NOTIFICATION()` を使用して、書式化されたメッセージ属性の表または通知履歴表をテキスト・メッセージや HTML メッセージの本文に含めることができます。

10. 「適用」を選択して変更内容を保存します。
11. 「ロール」タブを選択し、このメッセージにアクセスできるロールを指定します。（この機能は、今後のリリースでサポートされます。）
12. 「アクセス」タブを選択し、このメッセージを変更できるアクセス・レベルを設定します。4-16 ページの「[オブジェクトへのアクセス許可の設定](#)」を参照してください。

13. 通知メッセージによって実行者に応答値を要求し、Oracle Workflow でその応答値を通知アクティビティの結果として解釈させるには、「結果」タブを選択して必要な情報を入力します。ここで指定した情報を使用して、RESULT という特殊な応答メッセージ属性が作成されます。4-22 ページの「[メッセージの結果](#)」を参照してください。

RESULT の表示名および説明を指定します。ドロップ・ダウン・フィールドから選択肢タイプを選択します。ここでは、通知アクティビティの結果タイプに指定した選択肢タイプと同一の選択肢タイプを選択する必要があります。「デフォルト」の「値」フィールドで選択肢コードを選択します。その選択肢コードが、RESULT メッセージ属性のデフォルト値として表示されます。

---

**注意：** 他のタイプのメッセージ属性の作成については、4-32 ページの「[メッセージ属性の定義](#)」を参照してください。

---

14. 「適用」を選択して変更内容を保存するか、「OK」を選択し、変更内容を保存してプロパティ画面を閉じるか、「キャンセル」を選択し、変更を取り消してプロパティ画面を閉じます。
15. 定義したメッセージが、ナビゲータ・ツリーの「メッセージ」のブランチの下に表示されます。ナビゲータ・ツリーでメッセージをダブルクリックするか、メッセージを選択して「編集」メニューから「プロパティ」を選択すると、いつでもこのメッセージのプロパティを検討または編集できます。

メッセージに「結果」が定義されている場合、ナビゲータ・ツリーではそのメッセージ・アイコンに赤い疑問符が重なって表示されるため、「結果」定義を持たないメッセージと区別しやすくなります。

16. このメッセージの件名と本文に含めたメッセージ属性を、すべて定義する必要があります。
17. 項目タイプ属性を参照するメッセージ属性を作成するには、参照される項目タイプ属性をナビゲータ・ツリーで選択し、マウスの左ボタンを押したままメッセージにドラッグします。  
  
表示されたプロパティ画面を編集して、メッセージ属性の「ソース」が正しいかどうか確認します。「デフォルト」の「値」フィールドが自動的に「項目属性」に設定され、元の項目属性が参照されます。
18. 既存の項目タイプ属性に無関係なメッセージ属性を作成することもできます。4-32 ページの「[メッセージ属性の定義](#)」を参照してください。

### ▶ メッセージ属性の定義

ナビゲータ・コントロールのプロパティ

**属性**

メッセージ

内部名

表示名

説明

タイプ

ソース

長さ

デフォルト

タイプ

値

OK キャンセル 適用(Alt) ヘルプ

1. 既存の項目タイプ属性を参照しないメッセージ属性を作成するには、ナビゲータ・ツリーでメッセージを選択し、「編集」メニューから「新規属性」を選択します。  
「属性」プロパティ画面が表示されます。
2. 内部名をすべて大文字で空白を入れずに入力します。すべての Oracle Workflow API、SQL スクリプトおよび PL/SQL プロシージャでは、属性の識別時に内部名が参照されます。

---

---

**注意：** 定義したメッセージ属性の内部名を更新するには、wfchmsga.sql という特別な SQL スクリプトを使用する必要があります。このスクリプトは、設計時にメッセージ属性の内部名のエラーを訂正する場合にのみ使用してください。プロセスの実行中のインスタンスに関連するメッセージ属性の名前の変更には、使用しないでください。16-10 ページの「wfchmsga.sql」を参照してください。

---

---

---

---

**注意：** 内部名には、コロン (:) や空白を使用しないでください。

---

---

3. 「ソース」フィールドで「送信」または「応答」を指定し、この属性で通知メッセージの受信者に情報を提供するか、応答を求めるプロンプトを表示するかを指定します。
4. 表示名を入力します。これは、ナビゲータ・ツリーに表示される名前です。このメッセージ属性のソースを「応答」にした場合、この表示名は応答を求めるプロンプトとしても使用されます。

---

---

**注意：** 「送信」メッセージ属性の場合は、URL のドリルダウン先を記述する「URL」タイプの属性の場合にのみ、プレーン・テキストの電子メール通知に属性の表示名が表示されます。

---

---

5. オプションの説明を入力します。「応答」メッセージ属性の場合は、このフィールドを使用して応答インストラクションを指定します。
6. 属性のデータ型を選択します。
7. 属性の「タイプ」に応じて、次のデフォルト情報を指定します。
  - **テキスト：** テキスト属性の最大長を指定します。
  - **数値：** オプションで、数値の書式マスクとデフォルト値を指定します。
  - **日付：** オプションで、日付の書式マスクとデフォルト値を指定します。
  - **選択肢：** 値を取り出す事前定義済の選択肢タイプの名前を選択します。デフォルト値の選択肢コードを選択します。
  - **URL：** 「デフォルト」の「値」フィールドに、ネットワーク位置への Universal Resource Locator (URL) を指定します。4-11 ページの「URL 属性の定義」を参照してください。

---

---

**注意：** URL タイプの「応答」メッセージ属性は、プレーン・テキスト用電子メール・ソフトウェアから通知を表示すると正しく表示されません。URL タイプの「応答」メッセージ属性を持つメッセージを作成する場合は、ワークフローのユーザーに「通知の詳細」 Web 画面から通知を表示するように指示する必要があります。

---

---

---

---

**注意：** URL タイプの単一の「応答」メッセージ属性により、「通知の詳細」 Web 画面の応答フレームが置き換えられ、通知の受信者はカスタム HTML ページにリンクさせられて通知への応答を完了します。カスタム HTML の応答ドキュメントには、特殊な RESULT 属性（定義されている場合）など、すべての「応答」メッセージ属性の参照が含まれている必要があります。また、通知への応答完了をワークフロー・エンジンに知らせるために、ワークフロー・エンジンの CompleteActivity() API のコールも含まれている必要があります。

---

---

- **フォーム：** この属性は、Oracle Applications embedded Workflow にのみ関係します。

「デフォルト」の「値」フィールドで、開発者フォーム関数名とオプションの引数文字列（フォーム関数のパラメータ）を指定します。『Oracle Applications Developer's Guide』の「Overview of Menus and Function Security」および 4-12 ページの「[フォーム属性の定義](#)」を参照してください。

---

---

**注意：** 「フォーム」タイプの「送信」および「応答」メッセージ属性が表示されるのは、「通知」 Web 画面が Oracle Applications から起動される場合のみです。通知メッセージに「フォーム」タイプの「送信」メッセージ属性が含まれている場合は、添付フォーム・アイコンが使用可能になります。通知の受信者は、添付フォーム・アイコンをクリックし、メッセージ属性で定義されたフォーム関数にドリルダウンできます。

---

---



---

---

**注意：** メッセージに「フォーム」タイプの「応答」メッセージ属性が含まれている場合は、通知の「応答」セクションに表示される添付フォーム・アイコンを選択するだけで、指定されたフォームに直接ドリルダウンできます。このフォームは、通知応答が完了したことがワークフロー・エンジンに通知されるように、ワークフロー・エンジンの `CompleteActivity()` API のコールを使用してコーディングする必要があります。8-20 ページの「[Workflow Engine API](#)」を参照してください。

---

---

- **文書：** 文書を識別する文字列を「デフォルト」の「値」フィールドに入力します。4-13 ページの「[文書属性の定義](#)」を参照してください。
- **ロール：** ロール名を指定します。通知メッセージにロール・タイプのメッセージ属性が含まれる場合、属性は自動的にロールの表示名に設定されるため、ロールの内部名と表示名について別々の属性を保守する必要がなくなります。また、Web ブラウザから通知を表示する場合は、ロールの表示名がそのロールの電子メール・アドレスを示すハイパーテキスト・リンクになります。属性のデフォルト値を設定するには、最初にデータベースからロールをロードする必要があります。5-25 ページの「[ロール](#)」を参照してください。
- **イベント：** デフォルト値としてイベント項目タイプ属性を指定します。

---

---

**注意：** メッセージ属性のデータ型として「属性」を指定しないでください。このデータ型は、通知メッセージには無意味であり、ワークフロー通知システムでもサポートされません。

---

---



---

---

**注意：** 「応答」メッセージ属性のタイプが「日付」、「数値」、「テキスト」、「文書」または「ロール」の場合は、通知の受信者に対して、それぞれ日付、数値、テキスト値、文書、ロール（内部名または表示名）による応答を求めるプロンプトが表示されます。

「応答」メッセージ属性のタイプが選択肢の場合は、値リストから応答を選択するように求めるプロンプトが通知の受信者に表示されます。

---

---

8. メッセージ属性のタイプが「URL」の場合は、「フレーム・ターゲット」を指定します。このメッセージ属性をメッセージ内で参照すると、フレーム・ターゲットに何を指定したかに応じて、URL がオープンします。フレーム・ターゲットは、次のとおりです。
  - **新規ウィンドウ：** URL は新しい名称未設定のブラウザ・ウィンドウにロードされます。
  - **同じフレーム：** URL は、その URL 属性を参照する要素と同じフレームにロードされます。

- 親フレーム： URL は現在のフレームのすぐ上位の FRAMESET にロードされます。現在のフレームに上位フレームがなければ、この値は「同じフレーム」と等価です。
  - フル・ウィンドウ： URL は、元のフル・ウィンドウにロードされ、他のフレームがすべて取り消されます。現在のフレームに上位フレームがなければ、この値は「同じフレーム」と等価です。
9. メッセージの属性が「送信」で、タイプが「文書」の場合は、「内容の添付」をオンにして、その属性の内容を通知メッセージに添付できます。「通知」 Web 画面・インタフェースから通知を表示すると、通知メッセージ本文に続いて文書アイコンが表示され、このアイコンをクリックすると添付されたメッセージ属性の内容が示されます。電子メールから通知を表示した場合、添付ファイルの表示方法は、電子メール通知の環境設定によって異なります。10-2 ページの「[電子メールによる通知の閲覧](#)」を参照してください。

---

---

**注意：** 通知メッセージには、文書属性の添付も埋込み（トークンの置換を使用）もでき、両方行うことも可能です。

---

---

10. メッセージの属性が「送信」で、タイプが「URL」の場合は、「内容の添付」をオンにして、「通知参照」という添付ファイルを通知メッセージに添付できます。この添付ファイルには、「内容の添付」がオンになっているメッセージの各 URL 属性へのリンクが含まれています。URL へのリンクを選択すると、その URL にナビゲートできます。

---

---

**注意：** 通知メッセージには、URL 属性の添付も埋込み（トークンの置換を使用）もでき、両方行うことも可能です。

---

---

11. メッセージ属性の場合は、デフォルト値として定数または項目タイプ属性を指定できます。デフォルトで項目タイプ属性の値全体を直接参照する場合は、「項目属性」を選択し、ドロップ・ダウン・フィールドを使用して項目タイプ属性を選択します。ここで選択する項目タイプ属性は、メッセージ自体が関連付けられているのと同じ項目タイプに関連付ける必要があります。また、メッセージ属性と同じデータ型の項目タイプ属性を選択する必要があります。

---

---

**注意：** 「テキスト」タイプのメッセージ属性には、すべての項目属性タイプとの互換性がありますが、他のメッセージ属性タイプは、いずれも項目属性タイプと完全に一致させる必要があります。

---

---

12. 「適用」を選択して変更内容を保存するか、「OK」を選択し、変更内容を保存してプロパティ画面を閉じるか、「キャンセル」を選択し、変更を取り消してプロパティ画面を閉じます。

13. 定義したメッセージ属性は、ナビゲータ・ツリーで定義対象となったメッセージの下に表示されます。ナビゲータ・ツリーで属性をダブルクリックするか、属性を選択して「編集」メニューから「プロパティ」を選択すると、いつでもその属性のプロパティを検討または編集できます。ナビゲータ・ツリーの「応答」メッセージ属性アイコンには、「送信」メッセージ属性アイコンと区別できるように、赤い疑問符が表示されます。

---

**注意：** メッセージ属性では、その親メッセージのアクセス / 保護レベルが想定されます。

---



---

**注意：** ナビゲータ・ツリーで「応答」メッセージ属性を表示する順序は、これらの属性が通知メッセージの応答セクションに表示される順序に関連します。ナビゲータ・ツリーのドラッグ・アンド・ドロップ機能を使用するか、属性を選択して「編集」メニューから「属性を上へ移動」や「属性を下へ移動」を選択して、属性セット内での属性の順序を変更できます。

---

#### 関連項目：

4-37 ページ「[「応答」メッセージ属性の例](#)」

3-9 ページ「[プロパティ画面の「編集」ボタンの使用](#)」 電子メールによる

10-2 ページ「[電子メールによる通知の閲覧](#)」

### 「応答」メッセージ属性の例

次の例は、デフォルト値のない「応答」メッセージ属性のサンプル・セットを使用して、電子メール通知の「応答」セクションが通知システムによって生成される状況を示しています。

次の表に、「応答」メッセージ属性の例をいくつか示します。

**表 4-2**

内部名	タイプ	書式 / 選択肢タイプ	表示名	説明
RESULT	選択肢	WFSTD_APPROVAL	アクション	承認するかどうか。
COMMENT	テキスト	2000	レビュー・コメント	
REQDATE	日付	DD-MON-YYYY	必須の日付	必須の日付がない場合は空白のままにします。
MAXAMT	数値		最大金額	これは承認額の最大値です。

テンプレートによる応答方式の場合は、次のボイラープレート・テキストを使用して、電子メール通知の「応答」テンプレート・セクションが生成されます。

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

テンプレートによる応答の電子メール通知に表示される応答テンプレートの部分

承認するかどうか。

アクション:""

承認する

却下する

レビュー・コメント:""

必須の日付がない場合は空白のままにします。

必須の日付:""

これは承認額の最大値です。

最大金額:""

直接応答方式の場合は、次のボイラープレート・テキストを使用して、電子メール通知の「応答」セクションが生成されます。

Enter the <Display Name> on line <Sequence>. <Description> <Type\_Hint>

<Display Name> は、メッセージ属性の表示名に置き換えられます。<Sequence> は、ナビゲータ・ツリーにすべての「応答」メッセージ属性とともに表示される相対順序番号に置き換えられます（つまり、順序の決定時には、「送信」メッセージ属性の存在は無視されます）。<Description> は、メッセージ属性の説明に置き換えられます。また、<Type\_Hint> は、メッセージ属性が次のデータ型のいずれかと一致する場合は、右側の文で置換されます。

<u>タイプ</u>	<u>Type_Hint</u>
選択肢	値は次のどちらかにしてください。 <list of lookup codes>
日付	値は日付である必要があります [ 形式は "<format>" ]。
数値	値は数値である必要があります [ 形式は "<format>" ]。

タイプ	Type_Hint
テキスト	値は <format> バイト以内にします。

### 直接応答の電子メール通知に表示される応答セクションの部分

- 1 行目にアクションを入力してください。承認しますか? 値は次のどちらかにしてください。
- 承認する  
却下する
- 2 行目にレビュー・コメントを入力してください。値は 2000 バイト以内にします。
- 3 行目に必須の日付を入力してください。必須の日付がない場合は空白のままにします。値は「DD-MON-YYYY」形式の日付にします。
- 4 行目に最大金額を入力してください。これは承認額の最大値です。値は数値にします。

### ► 属性のトークン置換

- Oracle Workflow では、属性のランタイム・トークンの置換がサポートされています。属性は、他の属性の中に埋め込むことも、メッセージの件名および本文に埋め込むこともできます。属性を埋め込むには、トークンの置換に使用する属性を、&attr\_name のように指定します。attr\_name は、属性の内部名です。

トークンの置換を実行すると、Oracle Workflow によって属性の内部名とその値がフェッチされます。トークンの置換が必要な属性が、他の属性とネストされている場合、属性の内部名の長さに従って、ネストされた属性リストが整理され、最も長い内部名から順に属性の置換が開始されます。

**注意：** 必要なメッセージ本文の内容を表示するために、2 階層より深くメッセージ属性をネストする必要がある場合、PL/SQL 文書タイプのメッセージ属性を作成することを検討してください。4-5 ページの「[外部文書の統合](#)」を参照してください。

### ► メッセージのコピー

1. ナビゲータ・ツリーで、コピーするメッセージを選択します。
2. マウスの左ボタンを押しながら、メッセージをコピー先の「項目タイプ」ブランチまでドラッグします。
3. マウス・ボタンを離すと、新規メッセージのプロパティ画面が表示されます。

---

---

**注意：**「編集」メニューの「コピー」および「貼り付け」オプションも使用できます。

---

---

4. 新規の内部名と表示名を入力します。
5. メッセージのプロパティをさらに変更します。
6. 操作の完了後に「OK」を選択します。

---

---

**注意：**メッセージをコピーすると、そのメッセージに割り当てられているメッセージ属性もコピーされます。

---

---

## アクティビティ

アクティビティは、プロセスを完了させるために行う作業の単位です。アクティビティには、通知アクティビティ、関数アクティビティ、イベント・アクティビティまたはプロセス・アクティビティがあります。通知アクティビティは、ワークフローのユーザーにメッセージを送信します。メッセージには、単にユーザーに情報を提供するものと、なんらかのアクションを要求するものがあります。関数アクティビティは、PL/SQL のストアド・プロシージャまたは外部プログラムをコールして、自動化関数を実行します。イベント・アクティビティは、ビジネス・イベントを受信、発生または送信します。プロセス・アクティビティはモデル化されたワークフロー・プロセスで、他のプロセス内にアクティビティとして組み込んで、サブプロセスとして使用できます。

アクティビティは、ナビゲータ・ツリーでそれぞれの「プロセス」、「通知」、「関数」または「イベント」ヘッダーの下に編成されます。ナビゲータ・ツリーでアクティビティの定義を作成、編集および削除できます。また、アクティビティをツリーから「プロセス」ウィンドウにドラッグし、プロセス・ダイアグラムにそのアクティビティの新しい使用方法を作成することもできます。各アクティビティは、プロセス・ダイアグラム内でアイコンとして表示されます。

Oracle Workflow には、「標準」という項目タイプが用意されており、定義するあらゆるプロセスに使用できるように一般的なアクティビティが含まれています。たとえば、2 つの値を比較するなど、標準的な機能を実行するアクティビティがあります。6-2 ページの「[標準アクティビティ](#)」を参照してください。

また、Oracle Workflow には、「システム:エラー」という項目タイプも用意されており、そこに含まれる標準エラー・プロセスとアクティビティを使用して、カスタム・エラー・プロセスを作成できます。エラー・プロセスをプロセス・アクティビティに割り当てることができます。エラーが発生すると、エラー・プロセスによってエラーの対処方法が Oracle Workflow に通知されます。6-23 ページの「[デフォルト・エラー・プロセス](#)」を参照してください。

## 通知アクティビティ

ワークフロー・エンジンは、通知アクティビティに達すると、通知システムに `Send()` API コールを発行し、割り当てられた実行者にメッセージを送信します。通知によって送信されるメッセージはユーザーが定義します。メッセージとして、情報メモや、実行者に応答入力を求めるプロンプトを使用できます。実行者が通知アクティビティに応答すると、その応答が通知システムにより処理され、通知アクティビティを完了したため次の「適格」アクティビティの処理を継続するように、ワークフロー・エンジンに通知されます。4-45 ページの「[通知アクティビティの作成](#)」を参照してください。

プロセスに通知アクティビティをノードとして組み込む場合は、通知アクティビティの実行者を指定します。実行者として、特定のロールまたは動的にロール名を戻す項目タイプ属性を指定できます。5-8 ページの「[プロセスのノードの定義](#)」および 5-25 ページの「[ロール](#)」を参照してください。

通知アクティビティを定義するときに、オプションで次のことも可能です。

- 「[ロールの拡張](#)」をオンにして、ロール内の各ユーザーに、通知メッセージの個別コピーを送信できます。ユーザーが応答するか閉じるまで、通知はユーザーの通知キューに残ります。

---

**注意：** ロールの全ユーザーが閲覧する必要のあるブロードキャスト・タイプのメッセージを送信する場合は、ロールを拡張してください。

---

通知アクティビティのロールを拡張しなければ、割り当てた実行者ロールに通知メッセージが 1 通送信され、その通知はロールの全ユーザーの通知キューで表示されます。ロールのユーザーが 1 人でも通知に応答するか、または通知を閉じると、この通知はロールの全ユーザーの通知キューから削除されます。

- 通知の送信後に通知の状態更新に応じてワークフロー・エンジンにより実行される、通知後関数を指定します。ワークフロー・エンジンでは、受信者が通知に応答したか、通知を転送したか、通知を譲渡したか、または通知がタイムアウトになったかに応じて、通知後関数が `RESPOND`、`FORWARD`、`TRANSFER` または `TIMEOUT` のいずれかのモードで実行されます。通知システムが `RESPOND` モードで通知後関数の実行を終了すると、ワークフロー・エンジンは `RUN` モードで通知後関数を再実行します。

たとえば、通知転送先のロールを制限する場合は、受信者が通知の転送を試行すると `FORWARD` モードで通知後関数が実行されるように指定できます。通知後関数はロールを監査して、転送を許可するか、またはエラーを戻して転送を拒否します。8-14 ページの「[通知後関数](#)」および 8-197 ページの「[通知モデル](#)」を参照してください。

通知後関数を作成するには、関数アクティビティの場合と同じ `PL/SQL API` を使用する必要があります。7-3 ページの「[関数アクティビティがコールする PL/SQL プロシージャの標準 API](#)」を参照してください。

「[ロールの拡張](#)」をオンにして、通知後関数を指定すると、独自の投票集計アクティビティを作成できます。4-58 ページの「[投票アクティビティ](#)」を参照してください。

## 関数アクティビティ

関数アクティビティは、PL/SQL ストアド・プロシージャ、またはそれによりコールされる外部プログラムによって定義されます。通常、関数アクティビティは、プロセスにおいて完全に自動化された手順を実行するために使用します。PL/SQL ストアド・プロシージャであるため、関数アクティビティは通常の引数を受け入れて実行後の結果を戻すことができます。

ストアド・プロシージャのパラメータを渡すと、パラメータをアクティビティ属性として外部から識別できます。アクティビティ属性の値は、アクティビティをプロセスのノードとして定義するときに設定できます。このようなアクティビティ属性は、現行のアクティビティにのみ有効で、項目タイプ属性のようにグローバルではないため注意してください。5-17 ページの「[アクティビティ属性値の定義](#)」を参照してください。

外部プログラムとして、関数アクティビティではペイロード情報を Oracle Advanced Queuing の送信キューにエンキューでき、外部エージェントがそれをデキューして利用します。外部エージェントは同様に、更新した属性と完了結果を受信キューにエンキューし、ワークフロー・エンジンがこれを使用して処理します。

外部 Java プログラムとして、関数アクティビティではペイロード情報を Oracle Advanced Queuing の送信キューにエンキューでき、Java 関数アクティビティ・エージェントがそれをデキューして利用します。Java プログラムの結果は、受信キューにエンキューされ、ワークフロー・エンジンがこれを使用して処理します。現在、この機能を使用できるのは Oracle Workflow のスタンドアロン版のみです。4-47 ページの「[関数アクティビティの作成](#)」を参照してください。

## イベント・アクティビティ

イベント・アクティビティは、ワークフロー・プロセス内のビジネス・イベント・システムのビジネス・イベントを表します。関数を実行したり事前定義済エージェントにイベントを送信したりするときに、イベント・サブスクリプションの標準オプションより複雑なビジネス・イベントの処理やルーティング・ロジックをモデル化するときは、イベント・アクティビティをワークフロー・プロセスに組み込みます。13-2 ページの「[ビジネス・イベントの管理](#)」を参照してください。

イベント・アクティビティは、ビジネス・イベントを受信、発生または送信できます。

- 受信イベント・アクティビティは、特定のプロセスに対して「開始」アクティビティとしてマークすることができます。このプロセスでは、常にイベントを受信できます。また、受信イベント・アクティビティをそのプロセス内に配置すれば、プロセスが「受信」アクティビティに遷移したときにだけイベントを受信します。

受信アクティビティがイベントを受信すると、ワークフロー・エンジンはノードのイベントの詳細の指定に従って、イベント名、イベント・キーおよびイベント・メッセージを項目タイプ属性に格納し、イベント・メッセージのパラメータ・リストに含まれるすべてのパラメータをプロセスの項目タイプ属性として設定し、パラメータに対応する属性がまだ存在しない場合は新しい項目属性を作成し、イベント・アクティビティからスレッドの実行を続行します。「受信」アクティビティがイベントをすでに受信している場合は、「再開封時」フラグによってワークフロー・エンジンがアクティビティを再実



行するかどうかが決まります。4-56 ページの「[オプションのアクティビティ詳細の定義](#)」を参照してください。

イベントが別のワークフロー・プロセスの呼出しイベント・アクティビティによって発生した場合は、そのプロセスの項目タイプと項目キーがイベント・メッセージ内のパラメータ・リストに追加されます。ワークフロー・エンジンは、イベントを受信するプロセスの親として、指定されたプロセスを自動的に設定し、既存の親の設定を上書きします。8-81 ページの「[SetItemParent](#)」を参照してください。

- 呼出しイベント・アクティビティは、イベントに関する情報を取り出し、ビジネス・イベント・システムに対してイベントを発行します。ビジネス・イベント・システムは、そのイベントに対してサブスクリプションを実行します。「呼出し」アクティビティは、ノードのイベントの詳細に従って、イベント名、イベント・キーおよびイベント・データを取り出します。イベントの詳細は、項目タイプ属性を使用して実行時に動的に決定されます。イベント名は、イベント・アクティビティ・ノードの事前定義済みの定数として指定することもできます。

「呼出し」アクティビティは、定義されているアクティビティ属性の名前と値を取り出し、イベント・メッセージのパラメータ・リストに設定します。別のプロセスがイベント・メッセージを後で受信した場合、ワークフロー・エンジンはイベント・パラメータをそのプロセスの項目タイプ属性として設定します。8-248 ページの「[イベント・メッセージ構造](#)」を参照してください。

「呼出し」アクティビティは、現行のワークフロー・プロセスの項目タイプと項目キーをイベント・メッセージのパラメータ・リストに自動的に設定します。別のプロセスがイベント・メッセージを後で受信した場合、ワークフロー・エンジンはその項目タイプと項目キーを使用して、イベントを受信するプロセスの親として、イベントを呼び出したプロセスを自動的に設定します。8-81 ページの「[SetItemParent](#)」を参照してください。

- 送信イベント・アクティビティは、ノードのイベントの詳細に従って、イベント名、イベント・キー、イベント・メッセージ、送信エージェントおよび受信エージェントを取り出します。関連 ID がイベント・メッセージに指定されていない場合は、Oracle Workflow によって関連 ID がプロセスの項目キーに自動的に設定されます。次に、送信イベント・アクティビティは、送信エージェントのイベントを受信エージェントに直接送信します。イベントの詳細は、項目タイプ属性を使用して実行時に動的に決定されます。イベント名、送信エージェントおよび受信エージェントは、イベント・アクティビティ・ノードの事前定義済みの定数として指定することもできます。4-51 ページの「[イベント・アクティビティの作成](#)」および 5-13 ページの「[イベント・ノードのイベントの詳細の定義](#)」を参照してください。

---

**注意：**「送信」エージェント・アクティビティでは、ビジネス・イベント・システムに対してイベントを発行しないため、サブスクリプション処理は実行されません。

---

## プロセス・アクティビティ

プロセス・アクティビティは、特定の関連を持つアクティビティの集まりを表します。プロセス・アクティビティが他のプロセスに含まれている場合、そのプロセスはサブプロセスと呼ばれます。つまり、プロセス内のアクティビティは、それ自身がプロセスになることもあります。この階層の深さには制限はありません。4-54 ページの「[プロセス・アクティビティの作成](#)」を参照してください。

---

---

**注意：** Oracle Workflow では、1 つのプロセス階層内でサブプロセス・アクティビティを複数回使用することはできません。

---

---

### 関連項目：

C-4 ページ「[サブプロセス](#)」

## アクティビティ・コスト

各関数アクティビティおよびイベント・アクティビティには、コストが関連付けられています。コストは、ワークフロー・エンジンがアクティビティを実行するために必要な秒数を表す値です。ワークフロー・エンジンがそのアクティビティを実行する所要時間が不明な場合は、コストの見積りを入力し、後でパフォーマンスに関する情報が集まってから更新できます。通常、複雑で長時間かかるアクティビティには、高いコストを割り当てる必要があります。

コストの有効範囲は、0.00 ～ 1,000,000.00 です。

---

---

**注意：** Oracle Workflow Builder にコストを秒単位で入力して表示しても、実際には 100 分の 1 秒単位に換算されてデータベースに格納されます。

---

---

通常の処理では、ワークフロー・エンジンは 1 つのアクティビティの実行を完了してから次のアクティビティに移ります。場合によっては、処理に長時間がかかるため、バックグラウンド処理の方が適しているアクティビティもあります。

指定したしきい値よりコストが高いアクティビティをバックグラウンド処理にまわして延期するように、ワークフロー・エンジンを定義できます。ワークフロー・エンジンでは、次に適格な保留中のアクティビティの処理が続行されます。このアクティビティは、プロセスの別の並列分岐で発生することがあります。

ワークフロー・エンジンのデフォルトのしきい値は、50/100 秒です。これより高いコストのアクティビティは、バックグラウンド・エンジンにまわされます。バックグラウンド・エンジンは、特定タイプのアクティビティのみを実行するようにカスタマイズできます。ワークフロー・エンジンのしきい値は、SQL\*Plus で設定できます。2-40 ページの「[手順 WF-8 バックグラウンドのワークフロー・エンジンの設定](#)」、2-44 ページの「[エンジンのしきい値の設定](#)」および C-6 ページの「[アクティビティの遅延](#)」を参照してください。

## ▶ 通知アクティビティの作成

1. ナビゲータ・ツリーで、通知を作成する項目タイプを選択し、「編集」メニューから「新規通知」を選択します。「アクティビティ」プロパティ画面で、通知アクティビティを定義します。  
  
また、ナビゲータ・ツリーでメッセージを選択し、同じ項目タイプの「通知」ブランチにドラッグ・アンド・ドロップして、そのメッセージを送信する通知アクティビティを作成することもできます。
2. 通知アクティビティには、内部名（すべて大文字で空白を含みません）と表示名が必要です。表示名は、プロセス・ダイアグラムに表示される変更可能な名前です。説明を使用して、このアクティビティに関する説明を入力します。

---

---

**注意：** 定義したアクティビティの内部名を更新するには、`wfchact.sql` という特別な SQL スクリプトを使用する必要があります。このスクリプトは、設計時にアクティビティの内部名のエラーを訂正する場合にのみ使用してください。プロセスの実行中のインスタンスに関連するアクティビティの名前の変更には、使用しないでください。16-8 ページの「[wfchact.sql](#)」を参照してください。

---

---

---

---

**注意：** 内部名には、コロン (:) や空白を使用しないでください。

---

---

3. このアクティビティの結果タイプ（事前に定義された選択肢タイプ）を指定します。結果タイプによって、アクティビティから戻される可能性がある結果のリストが表示されます。完了したアクティビティからの戻り値に応じて、ワークフロー・ダイアグラムが分岐することがあります。4-6 ページの「[項目タイプの作成](#)」を参照してください。

アクティビティから値が戻されない場合や、ワークフロー・プロセスが戻り値に依存しない場合は、結果タイプとして「<なし>」を選択できます。

4. この通知を送信するメッセージの名前を選択します。4-27 ページの「[メッセージの作成](#)」を参照してください。
5. この通知を複数のユーザーで構成されるロールに割り当てる予定があり、この通知のコピーをロールの各ユーザーに送信する場合は、「ロールの拡張」をオンにします。「ロールの拡張」をオフにすると、通知のコピーはロールに対して 1 通のみ送信されます。4-41 ページの「[通知アクティビティ](#)」を参照してください。
6. オプションで、「関数名」フィールドに PL/SQL ストアド・プロシージャを指定できます。このプロシージャは通知後関数と呼ばれ、通知アクティビティに処理のロジックをまとめることができます。ワークフロー・エンジンでは、受信者が通知に応答したか、通知を転送したか、通知を移動したか、または通知がタイムアウトになったかに応じて、この通知後関数が RESPOND、FORWARD、TRANSFER または TIMEOUT のいずれかのモードで実行されます。通知システムが RESPOND モードで通知後関数の実行を終了すると、ワークフロー・エンジンは RUN モードで通知後関数を再実行します。7-3 ページの「[関数アクティビティがコールする PL/SQL プロシージャの標準 API](#)」および 8-14 ページの「[通知後関数](#)」を参照してください。

「ロールの拡張」をオンにし、この通知アクティビティに特殊な結果を含むメッセージを割り当て、「関数名」フィールドで、この通知の各受信者から返信された応答を集計するカスタム PL/SQL ストアド・プロシージャの名前を指定します。プロシージャを指定するには、書式 `<package_name>.<procedure_name>` を使用します。4-58 ページの「[投票アクティビティ](#)」を参照してください。

7. アクティビティを識別するアイコンを選択します。`.ico` ファイルに保存されているアイコンを使用して、アクティビティの処理をシンボル化できます。2-80 ページの「[手順 WF-12 Oracle Workflow へのカスタム・アイコンの追加](#)」を参照してください。

「参照」を選択すると、ワークフロー・アイコン・サブディレクトリ内のアイコン・ファイルが表示されます。

また、Windows の「エクスプローラ」や「ファイル マネージャ」からアイコン・ファイルをナビゲータ・ツリーのアクティビティ上にドラッグ・アンド・ドロップし、アクティビティにアイコンを設定することもできます。

8. 「適用」を選択して変更内容を保存します。
9. 「詳細」タブを選択し、アクティビティの詳細オプションを表示して変更します。4-56 ページの「[オプションのアクティビティ詳細の定義](#)」を参照してください。
10. 「ロール」タブを選択し、この通知アクティビティにアクセスできるロールを指定します。（この機能は、今後のリリースでサポートされます。）
11. 「アクセス」タブを選択し、この通知を変更できるアクセス・レベルを設定します。4-16 ページの「[オブジェクトへのアクセス許可の設定](#)」を参照してください。
12. 「OK」を選択して変更を保存し、プロパティ画面を閉じます。
13. 通知アクティビティが、ナビゲータ・ツリーで「通知」の下に表示されます。ナビゲータ・ツリーでアクティビティをダブルクリックするか、アクティビティを選択して「編集」メニューから「プロパティ」を選択するか、キーボードで [Enter] キーを押すと、いつでもこのアクティビティのプロパティを検討または編集できます。

#### 関連項目：

3-9 ページ「[プロパティ画面の「編集」ボタンの使用](#)」

### ▶ 関数アクティビティの作成

ナビゲータ・コントロールのプロパティ

アクティビティ | 詳細 | ロール | アクセス

内部名(I)

表示名(N)

説明(D)

アイコン(I)  

関数名(F)

関数タイプ(T)

結果タイプ(R)

コスト(C)

1. ナビゲータ・ツリーで、関数を作成する項目タイプを選択し、「編集」メニューから「新規関数」を選択します。「アクティビティ」プロパティ画面で、関数アクティビティを定義します。
2. 関数アクティビティには、内部名（すべて大文字で空白を含みません）と表示名が必要です。表示名は、プロセス・ダイアグラムに表示される変換可能な名前です。説明を使用して、このアクティビティに関する説明を入力します。

---

**注意：** 定義したアクティビティの内部名を更新するには、`wfchact.sql` という特別な SQL スクリプトを使用する必要があります。このスクリプトは、設計時にアクティビティの内部名のエラーを訂正する場合にのみ使用してください。プロセスの実行中のインスタンスに関連するアクティビティの名前の変更には、使用しないでください。16-8 ページの「[wfchact.sql](#)」を参照してください。

---

---

**注意：** 内部名には、コロン (:) や空白を使用しないでください。

---

3. このアクティビティで実行する関数の名前を入力します。「関数タイプ」フィールドで、この関数のタイプ（「PL/SQL」関数、「外部」関数または「外部 Java」関数）を指定します。

PL/SQL 関数の場合は、関数タイプを「PL/SQL」に設定し、関数を `<package_name>.<procedure_name>` として指定します。この関数は、標準 API に従って定義する必要があります。7-3 ページの「[関数アクティビティがコールする PL/SQL プロシージャの標準 API](#)」を参照してください。

外部関数アクティビティの場合は、関数タイプを「外部」に設定します。ワークフロー・エンジンで「送信」キューにエントリがエンキューされ、そのエントリと、Workflow スキーマ名および項目タイプで構成される値との相関値が次の書式で設定されます。

`<schema_name><item_type>`

**参照：** 8-167 ページの「[Workflow QUEUE API](#)」を参照してください。

この種のレコードを「送信」キュー内で検索するには、独自のキュー・ハンドラを作成する必要があります。このキュー・ハンドラでは、レコードに関連する処理を実行し、処理結果を「受信」キューにシードする必要があります。バックグラウンド・エンジンにより、受信キューにあるメッセージが処理され、元のワークフロー・プロセスが再開されます。8-9 ページの「[遅延処理](#)」を参照してください。

外部 Java 関数アクティビティの場合は、関数タイプを「外部 Java」に設定し、カスタム Java クラスのクラス名を関数名として入力します。現在、この機能を使用できるのは Oracle Workflow のスタンドアロン版のみです。カスタム・クラスがパッケージ内にある場合は、次の書式でクラス名の先頭にパッケージ名を付けます。

`<customPackage>.<customClass>`

Java クラスは、標準 API に従って定義する必要があります。7-8 ページの「[関数アクティビティがコールする Java プロシージャの標準 API](#)」を参照してください。

ワークフロー・エンジンがエントリを「送信」キューにエンキューします。Java 関数アクティビティ・エージェントは、このタイプのメッセージをデキューし、Java プログラムを実行して、結果を「受信」キューにエンキューします。バックグラウンド・エンジンにより、受信キューにあるメッセージが処理され、元のワークフロー・プロセスが再開されます。2-81 ページの「[手順 WF-13 Java 関数アクティビティ・エージェントの設定](#)」および 8-9 ページの「[遅延処理](#)」を参照してください。

---

**注意：** これらの「送信」キューおよび「受信」キューは、ビジネス・イベント・システムに使用するキューとは異なります。8-167 ページの「[Workflow QUEUE API](#)」を参照してください。

---



---

**注意：** 外部 Java 関数アクティビティを実行するには、CLASSPATH に JAR ファイルを指定する必要があります。

---

4. このアクティビティの結果タイプ（事前に定義された選択肢タイプ）を指定します。結果タイプによって、アクティビティから戻される可能性がある結果のリストが表示されます。完了したアクティビティからの戻り値に応じて、ワークフロー・ダイアグラムが分岐することがあります。4-18 ページの「[選択肢タイプの作成](#)」を参照してください。

アクティビティから値が戻されない場合や、ワークフロー・プロセスが戻り値に依存しない場合は、結果タイプとして「<なし>」を選択できます。

5. この関数アクティビティのコストを指定します。4-44 ページの「[アクティビティ・コスト](#)」を参照してください。
6. アクティビティを識別するアイコンを選択します。.ico ファイルに保存されているアイコンを使用して、アクティビティの処理をシンボル化できます。2-80 ページの「[手順 WF-12 Oracle Workflow へのカスタム・アイコンの追加](#)」を参照してください。

「参照」を選択すると、ワークフロー・アイコン・サブディレクトリ内のアイコン・ファイルが表示されます。

また、Windows の「エクスプローラ」や「ファイル マネージャ」からアイコン・ファイルをナビゲータ・ツリーのアクティビティ上にドラッグ・アンド・ドロップし、アクティビティにアイコンを設定することもできます。

7. 「適用」を選択して変更内容を保存します。
8. 「詳細」タブを選択し、アクティビティのオプションの詳細を表示して変更します。4-56 ページの「[オプションのアクティビティ詳細の定義](#)」を参照してください。

9. 「ロール」タブを選択し、この関数アクティビティにアクセスできるロールを指定します。（この機能は、今後のリリースでサポートされます。）
10. 「アクセス」タブを選択し、この関数を変更できるアクセス・レベルを設定します。4-16 ページの「[オブジェクトへのアクセス許可の設定](#)」を参照してください。
11. 関数アクティビティが、ナビゲータ・ツリーで「関数」の下に表示されます。ナビゲータ・ツリーでアクティビティをダブルクリックするか、アクティビティを選択して「編集」メニューから「プロパティ」を選択するか、キーボードで [Enter] キーを押すと、いつでもこのアクティビティのプロパティを検討または編集できます。
12. 作成する関数に入力引数が必要な場合は、その引数を関数アクティビティの属性として Oracle Workflow Builder に渡すことができます。関数アクティビティ属性は、プロセスでそのアクティビティが使用されるたびに変更可能な値を持つパラメータとして機能します。関数アクティビティ属性は、関数アクティビティに限定され、プロセスに対してグローバルではありません。4-8 ページの「[項目タイプまたはアクティビティ属性の定義](#)」を参照してください。

項目タイプ属性を参照する関数アクティビティ属性を作成するには、参照される項目タイプ属性をナビゲータ・ツリーで選択し、マウスの左ボタンを押したまま関数アクティビティにドラッグします。「デフォルト」の「値」フィールドが自動的に「項目属性」に設定され、元の項目属性が参照されます。

プロセスにノードとして関数アクティビティを含めると、そのノードに固有の値を関数アクティビティ属性に割り当てることができます。5-17 ページの「[アクティビティ属性値の定義](#)」を参照してください。

### 関連項目：

3-9 ページ「[プロパティ画面の「編集」ボタンの使用](#)」



## ▶ イベント・アクティビティの作成

1. ナビゲータ・ツリーで、イベントを作成する項目タイプを選択し、「編集」メニューから「新規イベント」を選択します。「アクティビティ」プロパティ画面で、イベント・アクティビティを定義します。
2. イベント・アクティビティには、内部名（すべて大文字で空白を含みません）と表示名が必要です。表示名は、プロセス・ダイアグラムに表示される変更可能な名前です。説明を使用して、このアクティビティに関する説明を入力します。

---

**注意：** 定義したアクティビティの内部名を更新するには、wfchact.sql という特別な SQL スクリプトを使用する必要があります。このスクリプトは、設計時にアクティビティの内部名のエラーを訂正する場合にのみ使用してください。プロセスの実行中のインスタンスに関連するアクティビティの名前の変更には、使用しないでください。16-8 ページの「wfchact.sql」を参照してください。

---



---

**注意：** 内部名には、コロン (:) や空白を使用しないでください。

---

3. アクティビティを識別するアイコンを選択します。.ico ファイルに保存されているアイコンを使用して、アクティビティの処理をシンボル化できます。2-80 ページの「手順 WF-12 Oracle Workflow へのカスタム・アイコンの追加」を参照してください。

「参照」を選択すると、ワークフロー・アイコン・サブディレクトリ内のアイコン・ファイルが表示されます。

また、Windows の「エクスプローラ」や「ファイル マネージャ」からアイコン・ファイルをナビゲータ・ツリーのアクティビティ上にドラッグ・アンド・ドロップし、アクティビティにアイコンを設定することもできます。

4. アクティビティの「イベント・アクション」を選択します。

- 受信： ビジネス・イベント・システムからイベントを受信します。
- 呼出し： ビジネス・イベント・システムにイベントを発行します。
- 送信： ビジネス・イベント・システムにイベントを再発行せずに、特定のイベント・エージェントのイベントを別のエージェントに直接送信します。

---

**注意：** 選択したイベント・アクションに応じて、次のイベントの詳細の一部またはすべてに対して項目タイプ属性を定義する必要があります。

- イベント名
- イベント・キー
- イベント・メッセージ
- イベント・データ
- 送信エージェント
- 宛先エージェント

イベント・アクティビティをノードとしてプロセスに含める場合は、項目タイプ属性を使用して、そのノードに必要なイベントの詳細情報の格納場所または検索場所を指定できます。イベントの詳細に使用する項目タイプ属性は、イベント・アクティビティが関連付けられている項目タイプと同じ項目タイプに関連付ける必要があります。4-8 ページの「[項目タイプまたはアクティビティ属性の定義](#)」および 5-13 ページの「[イベント・ノードのイベントの詳細の定義](#)」を参照してください。

---

5. 受信イベント・アクティビティを定義している場合は、「イベント・フィルタ」を入力すれば、受信できるイベントを指定できます。

- 指定したイベントだけを受信するには、完全な内部イベント名を入力します。

---

**注意：** イベント・フィルタには1つのイベントだけを指定します。イベント・フィルタにイベント・グループを指定することはできません。

---

- すべてのイベントをアクティビティで受信するには、「イベント・フィルタ」フィールドを空白のままにします。

**参照：**13-6 ページの「[イベントの定義](#)」を参照してください。

6. アクティビティのオプションのコストを入力します。イベント・アクションが「呼出し」または「送信」のイベント・アクティビティでは、コストを使用して、実行に時間がかかるアクティビティをバックグラウンド・エンジンに委任することができます。4-44 ページの「[アクティビティ・コスト](#)」を参照してください。
7. 「適用」を選択して変更内容を保存します。
8. 「詳細」タブを選択し、アクティビティのオプションの詳細を表示して変更します。4-56 ページの「[オプションのアクティビティ詳細の定義](#)」を参照してください。
9. 「ロール」タブを選択し、この関数アクティビティにアクセスできるロールを指定します。（この機能は、今後のリリースでサポートされます。）
10. 「アクセス」タブを選択し、このイベントを変更できるアクセス・レベルを設定します。4-16 ページの「[オブジェクトへのアクセス許可の設定](#)」を参照してください。
11. イベント・アクティビティが、ナビゲータ・ツリーで「イベント」の下に表示されます。ナビゲータ・ツリーでアクティビティをダブルクリックするか、アクティビティを選択して「編集」メニューから「プロパティ」を選択するか、キーボードで [Enter] キーを押すと、いつでもこのアクティビティのプロパティを検討または編集できます。
12. 呼出しイベント・アクティビティでは、発行したイベントのイベント・メッセージにパラメータを追加する場合、それらのパラメータを呼出しイベント・アクティビティの属性として定義できます。イベントが発生すると、アクティビティ属性がパラメータとしてイベント・メッセージのパラメータ・リストに追加されます。別のプロセスがイベント・メッセージを後で受信した場合、ワークフロー・エンジンはイベント・パラメータをそのプロセスの項目タイプ属性として設定します。アクティビティ属性の値は、プロセス内での呼出しイベント・アクティビティの使用方法にあわせて変更できます。イベント・アクティビティ属性は、イベント・アクティビティに限定され、プロセスに対してグローバルではありません。4-8 ページの「[項目タイプまたはアクティビティ属性の定義](#)」を参照してください。

項目タイプ属性を参照するイベント・アクティビティ属性を作成するには、参照される項目タイプ属性をナビゲータ・ツリーで選択し、マウスの左ボタンを押したままイベント・アクティビティにドラッグします。「デフォルト」の「値」フィールドが自動的に「項目属性」に設定され、元の項目属性が参照されます。

プロセスにノードとしてイベント・アクティビティを含めると、そのノードに固有の値をイベント・アクティビティ属性に割り当てることができます。5-17 ページの「[アクティビティ属性値の定義](#)」を参照してください。

**注意：** 呼出しイベント・アクティビティでは、現行のワークフロー・プロセスの項目タイプと項目キーも、イベント・メッセージのパラメータ・リストに自動的に設定されます。別のプロセスがイベント・メッセージを後で受信した場合、ワークフロー・エンジンはその項目タイプと項目キーを使用して、イベントを受信するプロセスの親として、イベントを呼び出したプロセスを自動的に設定します。8-81 ページの「[SetItemParent](#)」を参照してください。

➤ **プロセス・アクティビティの作成**

ワークフロー・プロセス・ダイアグラムを作成する前に、ナビゲータ・ツリーでプロセス・ダイアグラムを表すプロセス・アクティビティを作成する必要があります。

1. ナビゲータ・ツリーで、プロセス・アクティビティを作成する項目タイプを選択し、「編集」メニューから「新規プロセス」を選択します。「アクティビティ」プロパティ画面で、プロセス・アクティビティを定義します。  
  
プロセス・アクティビティが閉じており、それを再表示する場合は、ナビゲータ・ツリーでそのプロセス・アクティビティを選択して、[Enter] キーを押すか、またはマウスを右クリックしてポップアップ・メニューから「プロパティ」を選択します。
2. プロセス・アクティビティには、内部名（すべて大文字で空白を含みません）と表示名が必要です。表示名は、プロセス・ダイアグラムに表示される変換可能な名前です。説明を使用して、このアクティビティに関する説明を入力します。

---

---

**注意：** 定義したアクティビティの内部名を更新するには、wfchact.sql という特別な SQL スクリプトを使用する必要があります。このスクリプトは、設計時にアクティビティの内部名のエラーを訂正する場合にのみ使用してください。プロセスの実行中のインスタンスに関連するアクティビティの名前の変更には、使用しないでください。16-8 ページの「wfchact.sql」を参照してください。

---

---

---

---

**注意：** 内部名には、コロン (:) や空白を使用しないでください。

---

---

3. このアクティビティの結果タイプ（事前に定義された選択肢タイプ）を指定します。結果タイプによって、このプロセスから戻される可能性がある結果のリストが表示されます。4-18 ページの「[選択肢タイプの作成](#)」を参照してください。

プロセスの完了に関して特定の結果を記録する必要がない場合は、結果タイプとして「<なし>」を選択できます。

4. アクティビティを識別するアイコンを選択します。.ico ファイルに保存されているアイコンを使用して、アクティビティの処理をシンボル化できます。2-80 ページの「[手順 WF-12 Oracle Workflow へのカスタム・アイコンの追加](#)」を参照してください。

「参照」を選択すると、ワークフロー・アイコン・サブディレクトリ内のアイコン・ファイルが表示されます。

また、Windows の「エクスプローラ」や「ファイル マネージャ」からアイコン・ファイルをナビゲータ・ツリーのアクティビティ上にドラッグ・アンド・ドロップし、アクティビティにアイコンを設定することもできます。

5. 「実行可能」をオンにすると、このアクティビティが表すプロセスを上位レベルのプロセスとして開始し、単独で実行できます。プロセス・アクティビティがサブプロセスであり、上位レベルのプロセスからコールされた場合に限り実行するのであれば、「実行可能」をオフにします。8-22 ページの「[CreateProcess](#)」を参照してください。

---

---

**注意：** Oracle Workflow では、1 つのプロセス階層内でサブプロセス・アクティビティを複数回使用することはできません。1 つのプロセス内でサブプロセスを 2 回以上使用する場合は、必要なインスタンスごとにサブプロセスの個別コピーを作成する必要があります。

---

---

6. 「適用」を選択して変更内容を保存します。
7. 「詳細」タブを選択し、アクティビティのオプションの詳細を表示して変更します。4-56 ページの「[オプションのアクティビティ詳細の定義](#)」を参照してください。

8. 「アクセス」タブを選択し、このプロセスを変更できるアクセス・レベルを設定します。プロセス・アクティビティに対して設定したアクセス・レベルによって、そのプロセス・ダイアグラムにアクセスして編集できるユーザーが決定されます。4-16 ページの「オブジェクトへのアクセス許可の設定」を参照してください。
9. 「適用」を選択して変更内容を保存するか、「OK」を選択し、変更内容を保存してプロパティ画面を閉じるか、「キャンセル」を選択し、変更を取り消してプロパティ画面を閉じます。
10. プロセス・アクティビティが、ナビゲータ・ツリーで「プロセス」の下に表示されます。アクティビティを選択して「編集」メニューから「プロパティ」を選択するか、キーボードで [Enter] キーを押すと、いつでもこのアクティビティのプロパティを参照および編集できます。

**関連項目：**

3-9 ページ「プロパティ画面の「編集」ボタンの使用」

► **オプションのアクティビティ詳細の定義**

ナビゲータ・コントロールのプロパティ

アクティビティ | **詳細** | ロール | アクセス

エラー項目タイプ① WFERROR

エラー・プロセス②

有効日③ 2002/06/18

再到達時④ リセット

バージョン⑤ 0

OK キャンセル 適用(A) ヘルプ

1. アクティビティのプロパティ画面の「詳細」タブを選択します。
2. プロセス・アクティビティを作成する場合、現在のプロセスでエラーが発生した場合に実行するエラー・プロセスを指定できます。エラー・プロセスを持つ項目タイプの内部名を「エラー項目タイプ」入力し、実行するエラー・プロセス・アクティビティの内部名を「エラー・プロセス」に入力してください。エラー・プロセスの項目タイプは、現在の Oracle Workflow Builder セッションでオープンしていなくても、ここで定義でき

ることに注意してください。6-23 ページの「[デフォルト・エラー・プロセス](#)」を参照してください。

3. 「有効日」には、このバージョンのアクティビティをワークフロー・エンジンで実行できるようになる日付が表示されます。「有効日」フィールドが空白の場合、アクティビティはただちに有効になります。

有効日は、「ファイル」メニューの「別名保存」オプションを使用して変更内容を保存するときに設定します。アクティビティに対するすべての変更内容には、保存するときに同じ有効日が適用されます。

4. 「再開封時」に値を指定して、このアクティビティに2度以上推移した場合のワークフロー・エンジンでの処理を決定します。このアクティビティがループ内で再実行される最初のアクティビティの場合は、「再開封時」を設定し、ワークフロー・エンジンによるループの処理方法を指定する必要があります。ループ内の最初のアクティビティは、ピボット・アクティビティとも呼ばれます。ループ内のそれ以外のアクティビティについては、「再開封時」の値は無関係です。

「再開封時」が「無視」に設定されている場合、ワークフロー・エンジンではアクティビティが1回のみ実行され、その後の再実行では無視されます。

「再開封時」が「リセット」に設定されている場合、ワークフロー・エンジンでは、ピボット・アクティビティから逆の順序でループを通過してループ内の完了したアクティビティがリセットされ、各アクティビティが CANCEL モードで実行されます。各関数の CANCEL モードには、直前の操作を取り消す特別なロジックを含めることができます。その後、ワークフロー・エンジンはループ内を本来の順序で通過し、ピボット・アクティビティから始めて各アクティビティを RUN モードで再実行します。

「再開封時」が「ループ」に設定されている場合、ピボット・アクティビティと、それに続くループ内の全アクティビティが、リセットされず最初の実行どおりに再度実行されます。8-11 ページの「[ループ](#)」を参照してください。

5. 検証中のアクティビティの改訂が、バージョン番号で識別されます。アクティビティに対する最新の更新が確実に使用されるように、そのアクティビティの有効な最新バージョン番号が使用されます。
6. 「適用」を選択して変更内容を保存します。

## ▶ アクティビティのコピー

1. ナビゲータ・ツリーで、コピーするアクティビティを選択します。
2. マウスの左ボタンを押しながら、アクティビティをコピー先の「項目タイプ」ブランチまでドラッグします。
3. アクティビティを同じ項目タイプ内でコピーすると、プロパティ画面が表示され、コピーしたアクティビティについて一意の新規内部名と表示名の入力を求めるプロンプトが表示されます。

---

---

**注意：**「編集」メニューの「コピー」および「貼り付け」オプションも使用できます。

---

---

4. 操作の完了後に「OK」を選択します。

---

---

**注意：** 関数アクティビティ、イベント・アクティビティまたは通知アクティビティをコピーすると、それに関連付けられた属性やメッセージもコピーされます。

---

---

## 投票アクティビティ

ロール内のユーザー・グループに通知を送信し、ユーザーからの応答を集計する、投票アクティビティを作成できます。集計結果によって、次のプロセスに進むアクティビティが決まります。

投票アクティビティは通知アクティビティの1つで、最初にユーザー・グループに通知メッセージを送信してから、PL/SQL の通知後関数を実行してユーザーの応答（投票）を集計します。

定義するアクティビティ属性と、通知アクティビティのプロパティ画面にある次の4つのフィールドによって、投票の動作が決定されます。

- 「メッセージ」フィールド
- 「結果タイプ」フィールド
- 「ロールの拡張」チェック・ボックス
- 「関数」フィールド

### ➤ 投票アクティビティの作成

1. 投票アクティビティで集計する応答を含む投票選択肢タイプを作成します。4-18 ページの「[選択肢タイプの作成](#)」を参照してください。
2. 受信者に対して、投票選択肢タイプから1つを選択して応答するように求める投票メッセージを作成します。メッセージの「結果」タブで必要事項を入力します。「結果」タブの選択肢タイプを、手順1で定義した投票選択肢タイプに設定します。4-27 ページの「[メッセージの作成](#)」を参照してください。
3. ナビゲータ・ツリーで、投票アクティビティを作成する項目タイプを選択し、「編集」メニューから「新規通知」を選択します。
4. 内部名（すべて大文字で空白を含みません）と表示名を指定します。説明を使用して、この投票アクティビティに関する説明を入力します。



---

**注意：** 定義したアクティビティの内部名を更新するには、wfchact.sql という特別な SQL スクリプトを使用する必要があります。このスクリプトは、設計時にアクティビティの内部名のエラーを訂正する場合にのみ使用してください。プロセスの実行中のインスタンスに関連するアクティビティの名前の変更には、使用しないでください。16-8 ページの「wfchact.sql」を参照してください。

---



---

**注意：** 内部名には、コロン (:) や空白を使用しないでください。

---

5. 「結果タイプ」フィールドには、投票アクティビティで集計する応答を列挙した選択肢タイプが含まれている必要があります。これは手順 1 で定義した投票選択肢タイプです。
6. 投票アクティビティを識別するアイコンを選択します。
7. 「メッセージ」フィールドで、手順 2 で作成した投票メッセージの名前を選択します。投票メッセージによって、受信者は応答を求められます。応答では、投票選択肢タイプで指定した定義済の値から 1 つが選択されます。
8. ワークフロー・エンジンで、最初に応答したユーザーのみでなくロール内の複数ユーザーからの応答が集計されるように、「ロールの拡張」をオンにします。4-41 ページの「通知アクティビティ」を参照してください。
9. 「関数」フィールドに、ユーザーからの投票を集計する関数を指定します。標準の「はい / いいえ投票」アクティビティによりコールされる PL/SQL プロシージャ WF\_STANDARD.VOTEFORRESULTTYPE を使用できます。WF\_STANDARD.VOTEFORRESULTTYPE は、一般的な集計関数です。関数により集計される応答候補は、投票アクティビティに指定した結果タイプによって定義されます。関数による応答の集計方法は、投票アクティビティに定義したアクティビティ属性によって決定されます。6-10 ページの「「はい / いいえ投票」アクティビティ」を参照してください。  
  
別の方法として、カスタムの集計関数を指定することもできますが、これは関数アクティビティの標準 API に準拠している必要があります。プロシージャを指定するには、書式 <package\_name>.<procedure\_name> を使用します。7-3 ページの「関数アクティビティがコールする PL/SQL プロシージャの標準 API」を参照してください。
10. 「適用」を選択して変更内容を保存します。
11. 「詳細」タブを選択し、アクティビティの「詳細」プロパティ画面を表示して変更します。4-56 ページの「オプションのアクティビティ詳細の定義」を参照してください。
12. 「ロール」タブを選択し、この通知アクティビティにアクセスできるロールを指定します。（この機能は、今後のリリースでサポートされます。）

13. 「アクセス」タブを選択し、この通知を変更できるアクセス・レベルを設定します。4-16 ページの「[オブジェクトへのアクセス許可の設定](#)」を参照してください。
14. WF\_STANDARD.VOTEFORRESULTTYPE 集計関数を使用する場合は、投票に予測される応答ごとに「数値」タイプの「カスタム」アクティビティ属性を作成します。投票に予測される各応答は、投票アクティビティの結果タイプに関連付けられた選択肢コードであることに注意してください。そのため、「カスタム」アクティビティ属性を定義する場合は、アクティビティ属性の内部名が選択肢コードの内部名、つまり応答値と一致する必要があります。

アクティビティ属性の値には、空白か、特定の結果に必要な票数のパーセンテージを表す数値を使用できます。パーセンテージを指定すると、その応答の実際の集計パーセンテージの方が指定したパーセンテージより大きい場合に、結果が一致します。アクティビティ属性値を空白にすると、ワークフロー・エンジンではそのアクティビティ属性の応答がデフォルトとして扱われます。つまり、投票の集計後に特定のパーセンテージに達しなければ、空白のアクティビティ属性に関連付けられている応答のうち最高票数を獲得した応答が結果となります。

---

**注意：** 集計された投票結果がどの応答パーセンテージにも達しない場合に、デフォルトの応答（空白のアクティビティ属性）が指定されていないければ、結果は #NOMATCH になります。投票アクティビティからの「一致しない」トランジションが存在する場合、ワークフロー・エンジンではこのトランジションが選択され、それ以外は「デフォルト」が選択されます。「デフォルト」トランジションが存在しない場合は、その結果に使用可能なトランジションがないことを示すエラー (ERROR:#NOTTRANSITION) が発生します。

---

---

**注意：** 集計された投票結果が複数の応答パーセンテージに達するか、応答パーセンテージには達成しないが、デフォルトの応答のうち同じパーセンテージのものがある場合、結果は #TIE になります。投票アクティビティからの「一致」トランジションが存在する場合、ワークフロー・エンジンではこのトランジションが選択され、それ以外は「デフォルト」が選択されます。「デフォルト」トランジションが存在しない場合は、その結果に使用可能なトランジションがないことを示すエラー (ERROR:#NOTTRANSITION) が発生します。

---

15. 集計関数 WF\_STANDARD.VOTEFORRESULTTYPE を使用する場合は、一連の「カスタム」アクティビティ属性を定義するのみでなく、内部名 VOTING\_OPTION を指定して「投票オプション」というアクティビティ属性を定義する必要があります。「投票オプション」アクティビティ属性を、「はい / いいえ投票」標準アクティビティからコピーすることもできます。

「投票オプション」アクティビティ属性では、投票の集計方法を指定します。指定できる値は、次のとおりです。

- 「参加者全員の投票を待つ」： ワークフロー・エンジンは、すべての投票が行われるまで待ってから、通知を受けた全ユーザーに対する割合として結果を集計します。タイムアウト条件が発生すると、タイムアウト発生前に出された投票の合計に対する割合として投票結果が計算されます。
- 「投票のつど集計」： ワークフロー・エンジンは、通知を受けたすべてのユーザーに対する累積割合を、応答があるたびに集計します。タイムアウト条件が発生すると、応答は合計投票数に対する割合として集計されます。カスタム「応答」アクティビティ属性のどれかの値が空白値の場合、このオプションは意味がなくなるため注意してください。
- 「参加者全員の投票が必要」： すべての投票が行われた後に限り、通知を受けた全ユーザーに対する割合で応答が評価されます。タイムアウト条件が発生すると、標準的なタイムアウト・トランジションに従って進むか、使用可能なトランジションが1つも存在しない場合はエラーが発生し、投票結果は集計されません。

#### 関連項目：

6-10 ページ「[「はい / いいえ投票」アクティビティ](#)」

## 投票方法の例

### 1. 単純過半数

次の表に、「単純過半数」投票方法の各応答に割り当てられている、カスタムの「応答」アクティビティ属性の値を示します。

**表 4-3**

応答	カスタムの「応答」アクティビティ属性の値
A	50
B	50
C	50

50 パーセントを超える票を獲得した応答が結果となります。50 パーセントを超える応答が1つもなければ、結果は該当なしとなります（#NOMATCH）。

### 2. デフォルト付き単純過半数

次の表に、「デフォルト付き単純過半数」投票方法の各応答に割り当てられている、カスタムの「応答」アクティビティ属性の値を示します。

表 4-4

応答	カスタムの「応答」アクティビティ属性の値
A	50
B	50
C	空白

応答 A が 50 パーセントを超える票を得た場合、A が結果となります。同様に、応答 B が 50 パーセントを超える票を得た場合は、B が結果となります。A と B のどちらの得票も 50 パーセントを超えなかった場合は、C が結果となります。

3. 複数のデフォルト付きの単純過半数

次の表に、「複数のデフォルト付きの単純過半数」投票方法の各応答に割り当てられている、カスタムの「応答」アクティビティ属性の値を示します。

表 4-5

応答	カスタムの「応答」アクティビティ属性の値
A	50
B	空白
C	空白

応答 A が 50 パーセントを超える票を得た場合、A が結果となります。A の得票率が 50 パーセント以下の場合は、B または C のうち得票数の高い方が結果となります。A が 50 パーセント以下の得票率で、B と C 両方が同じ数の票を得た場合、結果は一致となります (#TIE)。

4. 人気投票

次の表に、「人気投票」投票方法の各応答に割り当てられている、カスタムの「応答」アクティビティ属性の値を示します。

表 4-6

応答	カスタムの「応答」アクティビティ属性の値
A	空白
B	空白
C	空白

最高得票の応答が結果となります。

## 5. 反対投票

次の表に、「反対投票」投票方法の各応答に割り当てられている、カスタムの「応答」アクティビティ属性の値を示します。

**表 4-7**

応答	カスタムの「応答」アクティビティ属性の値
YES	100
NO	0

応答 NO への投票が 1 票でもあれば、結果は NO となります。

## 6. 評決

次の表に、「評決」投票方法の各応答に割り当てられている、カスタムの「応答」アクティビティ属性の値を示します。

**表 4-8**

応答	カスタムの「応答」アクティビティ属性の値
GUILTY	100
NOT_GUILTY	100

満場一致の応答が要求され、それ以外は該当なしとなります（#NOMATCH）。

### 関連項目：

6-10 ページ「[「はい / いいえ投票」アクティビティ](#)」

## Oracle Workflow Builder のオブジェクトの削除

Oracle Workflow Builder のオブジェクトは、他のオブジェクトによって参照されている場合でも、カスタマイズに対して保護されていないければ削除できます。削除するオブジェクトが他のオブジェクトによって参照されている場合は、「ワークフロー・エラー」ダイアログ・ボックスが表示され、失われる外部キー参照についての警告が示されます。そのまま続行してオブジェクトを削除するか、削除のアクションを取り消すことができます。削除を選択すると、ワークフロー・プロセス定義を保存または検証するときに「ワークフロー・エラー」ダイアログ・ボックスが表示され、定義内に存在する失われた外部キー参照がすべて報告されます。

この処理の結果、無効な外部キーを持つワークフロー定義を Oracle Workflow Builder にロードして修正できるようになります。Oracle Workflow Builder では、欠落している外部キーに対する元の内部名参照が保持され、プロセス定義のロード時にそれが検証エラー・メッセージに表示されます。削除したオブジェクトを、元の内部名を使用して元の項目タイプで再作成すると、プロセス定義内で失われた外部キー参照をリストアできます。

---

**注意：** Oracle Workflow Builder の項目タイプ定義全体を削除することもできます。

---

## Oracle Workflow Builder のオブジェクトの変更

ワークフロー・オブジェクトの定義を変更する前に、その定義に基づくアクティブな作業項目に悪影響を及ぼさないことを確認する必要があります。Oracle Workflow のオブジェクトに対する変更がアクティブな作業項目に及ぼす影響は、そのオブジェクトがバージョン管理をサポートしているかどうかに応じて異なります。

ワークフロー・オブジェクトの場合、バージョン管理とは、オブジェクト自体またはそれを所有するオブジェクトにより、データベース内の同じオブジェクトの複数のオカレンスがバージョン番号、開始日付および終了日付のみで区別され、サポートされることを意味します。たとえば、次の表の投票アクティビティは、WF\_ACTIVITIES 表に同時に存在させることができます。

表 4-9

名前	バージョン	開始日付	終了日付	メッセージ	選択肢タイプ
投票	1	1998 年 1 月 1 日	1998 年 12 月 31 日	投票メッセージ	はい / いいえ
投票	2	1999 年 1 月 1 日	< 空白 >	新規投票メッセージ	承認

バージョン管理をサポートしているワークフロー・オブジェクトを変更すると、元のバージョンと新バージョンの両方がデータベースに存在することになります。そのオブジェクトを参照するアクティブな作業項目は、その開始時に有効だったのと同じバージョンを引き続き使用して完了まで進行します。新バージョンを使用するのは、変更後に開始された新規作業項目のみです。

前述の例で、1998 年 1 月 1 日～1998 年 12 月 31 日に開始される作業項目では、1999 年 1 月 1 日までに完了するかどうかに関係なく、結果オプション Yes または No を持つメッセージ「投票メッセージ」が送信されます。結果オプション「承認」または「否認」を含むメッセージ「新規投票メッセージ」が送信されるのは、1999 年 1 月 1 日以後に開始される作業項目のみです。

**注意：** プロセス定義情報は、すべてバージョン管理対象となります。

ただし、バージョン管理をサポートしていないワークフロー・オブジェクトを変更すると、そのオブジェクトの以前の定義が更新され、データベースには変更後の定義のみが存在することになります。そのオブジェクトを参照するアクティブな作業項目では、いずれも変更後のオブジェクトが使用されます。

変更後のオブジェクトに、作業項目で使用されるワークフロー定義の残りの部分との互換性がなくなると、エラーになることがあります。このようなエラーを回避するには、変更

よって互換性が失われないように、変更の計画時にオブジェクトの参照をすべて考慮する必要があります。

---

**注意：** 可能であれば、アクティブな作業項目をすべて強制終了し、変更後に再起動して、下位互換性が失われる危険性を回避できます。この方法では、再起動した作業項目では、バージョン管理をサポートするかどうかを問わず、すべてのワークフロー・オブジェクトの変更後の定義が使用されます。

---

## バージョン管理をサポートするワークフロー・オブジェクト

次のワークフロー・オブジェクトでは、バージョン管理がサポートされます。

- 通知
- 関数
- イベント
- プロセスとサブプロセス
- プロセス・アクティビティ（ノード）
- アクティビティ属性
- アクティビティ属性値
- アクティビティ・トランジション

Oracle Workflow Builder の前述のオブジェクトのいずれかを変更してデータベースに保存すると、ワークフロー定義ローダーでは、そのオブジェクトの既存の定義は更新されません。かわりに、そのオブジェクトまたは所有オブジェクトの新バージョンが作成されます。

その結果、変更前に開始されていたアクティブな作業項目には影響を与えずに、これらのオブジェクトを変更できます。

たとえば、次のようになります。

- 新規メッセージを参照するように通知アクティビティを更新する場合、その通知はバージョン管理対象となります。
- 新規選択肢タイプを参照するように関数アクティビティを更新する場合、その関数はバージョン管理対象となります。
- 新規 API を参照するように関数アクティビティを更新する場合、その関数はバージョン管理対象となります。
- プロセス・ダイアグラムからプロセス・アクティビティ（ノード）を削除する場合は、そのプロセス内に存在するすべてのプロセス・アクティビティのみでなく、所有プロセスもバージョン管理対象となります。



- アクティビティにアクティビティ属性を追加する場合は、所有アクティビティがバージョン管理対象となります。

前述のどの例でも、変更はそれ以後に開始される作業項目にのみ影響します。

## バージョン管理をサポートしないワークフロー・オブジェクト

次のワークフロー・オブジェクトでは、バージョン管理がサポートされません。

- 項目属性
- メッセージ
- 選択肢
- 関数アクティビティにより参照される PL/SQL コード

Oracle Workflow Builder の項目属性、メッセージまたは選択肢のいずれかを変更してデータベースに保存すると、ワークフロー定義ローダーでは、そのオブジェクトの既存の定義が更新されます。また、関数アクティビティの既存の PL/SQL API を変更すると、その関数アクティビティではデータベースに格納された更新後の API が参照されます。

その結果、前述のオブジェクトのいずれかを変更すると、変更結果はそのオブジェクトを参照するアクティブな作業項目すべてに即時に反映されます。変更によって下位互換性が失われるように慎重に計画を立ててください。

---

---

**注意：** Oracle Workflow Builder では、PL/SQL コードの編集はサポートされません。PL/SQL コードは、単に Oracle Workflow の関数アクティビティで参照されるコードを変更した場合の結果を説明するためののみ列挙してあります。

---

---

## 項目属性

作業項目が開始されると、Oracle Workflow では、その項目タイプに定義された各項目属性のランタイム・コピーが作成されます。ワークフロー・エンジンでは、ワークフロー・プロセス内の関数アクティビティの PL/SQL コードで項目属性が参照されるたびに、これらのランタイム・コピーが参照されます。

作業項目の開始後に新規項目属性を追加しても、アクティブな作業項目には反映されません。ただし、これらの作業項目には、`AddItemAttr()` または `AddItemAttributeArray` API をコールして属性を意図的に作成しない限り、新規項目属性は含まれません。ワークフロー・プロセス内の既存の PL/SQL コードで新規項目属性の参照も追加すると、その参照が原因で、新規項目属性を含んでいないアクティブな作業項目にエラーが発生することがあります。

たとえば、Workflow Engine API をコールして、関数アクティビティの PL/SQL API を新規項目属性と通信するように変更すると、新規に定義された項目属性を持たないアクティブな作業項目については、関数アクティビティが失敗します。

ワークフロー・プロセス内で既存の PL/SQL コードを変更する場合は、新規項目属性の参照を追加するのは、それを必要とするアクティブな作業項目についても項目属性を作成する場合に限るように、慎重に計画する必要があります。4-70 ページの「PL/SQL コード」を参照してください。

---

**注意：** ただし、ワークフロー・プロセスを起動する API には、新規項目属性の参照を追加でき、アクティブな作業項目に関して特に項目属性を作成する必要はありません。たとえば、`SetItemAttribute` または `SetItemAttributeArray` API をコールして、プロセスの開始時に新規項目属性を作成できます。アクティブな作業項目はすでにこのコードを通過しているため、この種の変更の影響を受けません。

---

## メッセージ

ワークフロー・エンジンが通知システムに対してメッセージの送信を要求すると、通知システムは各メッセージ属性の通知表に通知属性を作成します。通知属性の行には、件名行や本文のテキストなど、実行時にメッセージ本文にトークンが置換される変数データが含まれています。

ただし、メッセージ本文は通知表にコピーされません。かわりに、メッセージ本文は、通知がユーザーに表示されるときに、様々な通知システム API により実行時に参照されます。したがって、メッセージ本文の変更は、変更後に送信される通知のみでなく、変更前に送信されたアクティブな作業項目の通知にも影響します。

非互換性エラーが発生する危険性を回避して、メッセージ本文に特定のタイプの変更を加えることができます。これには、次のような変更が含まれます。

- 静的テキストの追加
- 静的テキストの編集
- 静的テキストの削除
- メッセージ属性トークンの削除

たとえば、メッセージ本文に「5 日以内に承認してください。」などの静的センテンスを追加すると、アクティブな作業項目内のすべての通知に、アクセス時にこのセンテンスが組み込まれます。通知システムでは、追加のセンテンスの解決には他の情報は不要なため、変更後のメッセージ本文をエラーなしで表示できます。

ただし、新規に作成されたメッセージ属性用のトークンを追加するなど、不適切な変更を加えると、アクティブな作業項目内の通知が正常に表示されなくなることがあります。変更は、エラーを回避できるように慎重に計画してください。

新規メッセージ属性のトークンをメッセージ本文に追加する必要がある場合は、既存のメッセージを変更するのではなく、新規メッセージを作成して変更を実装する必要があります。通知アクティビティではバージョン管理がサポートされるため、アクティブな作業項目に影響を与えずに、既存の通知アクティビティに新規メッセージを添付できます。

たとえば、「承認者名」などの新規メッセージ属性を作成し、その属性のトークンをメッセージ本文に追加すると、アクティブな作業項目内のすべての通知には、アクセス時に新規トークンが組み込まれます。ただし、変更前に送信された通知には、通知属性として新規メッセージ属性「承認者名」は組み込まれません。通知システムでは、新規メッセージ属性の参照を解決できず、かわりに通知にはトークン「&APPROVER\_NAME」が表示されます。

この例では、元のメッセージ本文を変更するかわりに、新規メッセージ属性を含む新規メッセージを作成し、新規属性のトークンを新規メッセージの本文に追加して、新規メッセージを通知アクティビティに添付する必要があります。変更結果を保存すると、通知アクティビティがバージョン管理対象となります。その後、アクティブな作業項目は引き続き通知アクティビティの旧バージョンを参照し、互換性のないトークンは通知に表示されなくなります。

## 選択肢タイプおよびコード

Oracle Workflow では、選択肢タイプには次の重要な用途があります。

- ワークフロー・アクティビティに考えられる完了ステータス（選択肢コード）を決定します。
- 「応答」メッセージ属性に考えられる完了ステータス（選択肢コード）を決定します。

選択肢タイプに不適切な変更を加えると、その選択肢タイプを参照するアクティブな作業項目が失敗することがあります。

下位互換性の消失によるエラーを回避するには、アクティブな作業項目により参照される選択肢タイプについて、次のガイドラインに従ってください。

- 選択肢タイプを削除しないこと。
- 既存の選択肢タイプから選択肢コードを削除しないこと。
- 既存の選択肢タイプに選択肢コードを追加しないこと。

前述のいずれかの変更を行う必要がある場合は、既存の選択肢タイプを変更するのではなく、新規の選択肢タイプを作成して変更を実装する必要があります。

アクティビティではバージョン管理がサポートされるため、既存のアクティビティには、アクティブな作業項目に影響を与えずに新規の選択肢タイプを添付できます。ただし、ワークフロー・メッセージではバージョン管理がサポートされないため、既存のメッセージ属性には新規の選択肢タイプを添付しないでください。

次の例は、選択肢タイプに不適切な変更を加えたために発生する一部のエラーを示しています。

- 「応答」メッセージ属性で参照される選択肢タイプに選択肢コードを追加すると、ユーザーは通知への応答として新規の選択肢コードを選択できるようになります。ただし、通知アクティビティの旧バージョンでは、新規の選択肢コード用の分離ロジックがモデル化されていません。ユーザーが新規コードを選択すると、プロセスは「ERROR」になります。

- 「応答」メッセージ属性で参照される選択肢タイプから選択肢コードを削除すると、ユーザーはその結果コードを通知への応答として選択できなくなります。

## PL/SQL コード

関数アクティビティではバージョン管理がサポートされますが、基礎となる PL/SQL コードでは、コード自体のバージョン管理を実装しない限り、バージョン管理がサポートされません。バージョン管理なしで PL/SQL コードを変更すると、そのコードを参照するアクティブな作業項目のビジネス・フローが変化します。不適切な変更を行うと、これらの作業項目が失敗することがあります。

関数アクティビティに関する PL/SQL API の変更がアクティブな作業項目に影響しないようにするには、既存の API を変更するのではなく、新規 API を作成して変更を実装する必要があります。関数アクティビティではバージョン管理がサポートされるため、アクティブな作業項目に影響を与えずに、既存の関数アクティビティに新規 API を添付できます。

既存の API を変更する必要があり、かわりに新規 API を作成できない場合は、変更によって下位互換性が失われるように慎重に計画する必要があります。

たとえば、API の戻り値のグループに選択肢コードを追加する予定であれば、最初に関数アクティビティ・ノードに「デフォルト」などの出力トランジションがあり、API がその値を戻すときにワークフロー・エンジンでたどれることを確認する必要があります。この条件が満たされていない場合は、その値が戻されるとプロセスが「ERROR」になります。適切な出力トランジションがない場合は、新規 API に変更を実装し、追加の戻り値について分岐ロジックをモデル化するようにプロセスを更新する必要があります。

たとえば、Workflow Engine API をコールして、関数アクティビティの既存の PL/SQL API を新規項目属性と通信するように変更する場合は、アクティブな作業項目について新規項目属性も作成する必要があります。そうでないと、新規項目属性が定義されていないアクティブな作業項目については、関数アクティビティが失敗します。

新規作成された項目属性をパラメータとして持つ次の API のいずれかをコールする場合に、アクティブな作業項目にその項目属性が定義されていない場合は、関数アクティビティは失敗することがあります。

- 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

アクティブな作業項目について新規項目属性のコピーを作成するには、AddItemAttr() か、AddItemAttributeArray API の 1 つをコールします。このコールは、カスタム・アップグレード・スクリプトまたは例外ハンドラに挿入できます。

- アップグレード・スクリプト： API を変更する前に、その API を参照するアクティブな作業項目について、新規項目属性を作成して挿入するカスタム・アップグレード・スクリプトを記述し、実行します。次の例は、アップグレード・スクリプトの記述方法を示しています。

```
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;
```

---

**注意：** アクティブな作業項目は、WF\_ITEMS.END\_DATE が NULL である項目として識別されます。

---

- 例外ハンドラ： 変更後の API で、新規項目属性の参照の後に、属性が存在しない場合に作成して挿入する例外ハンドラを追加します。次の例は、このような例外ハンドラの記述方法を示しています。

```
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>;
```

**関連項目：**

4-67 ページ [「項目属性」](#)





---

## ワークフロー・プロセス・ダイアグラムの定義

この章では、Oracle Workflow Builder を使用してワークフロー・プロセス・ダイアグラムを定義する方法と、特定のロールに通知アクティビティを割り当てることができるように、データベースからロールをロードする方法について説明します。

## 「プロセス」ウィンドウ

Oracle Workflow Builder の「プロセス」ウィンドウには、特定のプロセスのアクティビティ（アイコン）とトランジション（矢印）がグラフィカルに表示されます。各アクティビティはノードであり、プロセス完了のために実行される論理的な手順です。

ナビゲータ・ツリーからアクティビティを「プロセス」ウィンドウにドラッグ・アンド・ドロップするか、または「プロセス」ウィンドウに直接アクティビティを作成します。アクティビティ・ノードのプロパティは、「プロセス」ウィンドウのノードをマウスの左ボタンでダブルクリックして表示したり編集します。マウスの右ボタンを使用してアクティビティのノード間を矢印で結び、トランジションを定義します。

プロセス・ノードは、通知アクティビティ、関数アクティビティ、イベント・アクティビティおよびプロセス・アクティビティで構成されます。プロセスのダイアグラムで、プロセスにプロセス・アクティビティが含まれている場合、そのプロセス・アクティビティをサブプロセスと呼びます。この階層の深さには制限はありません。サブプロセス・ダイアグラムを「プロセス」ウィンドウに表示するには、上位の「プロセス」ウィンドウでサブプロセス・アクティビティ・ノードをダブルクリックします。

### トランジション

トランジションは、ダイアグラムには矢印で表示され、あるアクティビティが完了して別のアクティビティがアクティブになることを示します。結果タイプが「<なし>」で完了したアクティビティの場合、そこからのトランジションは単に次のアクティビティへの矢印として表示されます。これは、元のアクティビティが完了すると、プロセスが次のアクティビティに進むことを示します。

結果タイプが定義されているアクティビティの場合は、作成するトランジションの矢印をアクティビティに考えられる結果のうちの 1 つに関連付ける必要があります。アクティビティの完了時に戻される結果によって、次に適切なアクティビティが決定されます。次に適切なアクティビティは、完了したアクティビティを起点とする結果ベースのトランジションによって定義されます。たとえば、「承認者に通知」の結果が「否認済」の場合は、「購買申請を却下」に進みます。15-15 ページの「[購買申請プロセス・アクティビティ](#)」を参照してください。

また、結果タイプが定義されているアクティビティについては、デフォルト、任意またはタイムアウトのトランジションを作成できます。ワークフロー・エンジンは、完了結果と一致するトランジションが他になければ、デフォルト・トランジションに従います。任意トランジションには、アクティビティが戻す完了結果にかかわらず従います。そのため、特定の結果に従うアクティビティと並行してワークフロー・エンジンで実行される一般的なアクティビティをプロセスに含めることが可能です。ワークフロー・エンジンは、通知アクティビティが完了前にタイムアウトになると、タイムアウト・トランジションに従います。2-40 ページの「[手順 WF-8 バックグラウンドのワークフロー・エンジンの設定](#)」を参照してください。

アクティビティは 1 つの結果に対して複数のトランジションを持ち、並列分岐を作成できます。

## タイムアウト・トランジション

通知アクティビティが指定した時間内に完了しない場合に、プロセスで別のアクティビティを強制的に実行させるには、その通知アクティビティと別のアクティビティの間をタイムアウト・トランジションで結びます。5-8 ページの「[プロセスのノードの定義](#)」を参照してください。

アクティビティがタイムアウトになると、Oracle Workflow ではそのアクティビティがタイムアウトになったとしてマークされ、そのアクティビティに関連するすべての通知が取り消されます。取り消された通知に応答が必要で、実行者が通知を電子メールで受信する場合にのみ、通知システムでは実行者に取消メッセージを送信します。

その後、処理はプロセス定義で指定されたとおりタイムアウト・トランジションに沿って続行されます。タイムアウトになったアクティビティに、そのアクティビティを起点とするタイムアウト・トランジションがなければ、Oracle Workflow では、そのアクティビティまたはその親プロセスに関連したエラー・プロセスが実行されます。4-56 ページの「[オブションのアクティビティ詳細の定義](#)」を参照してください。

---

**注意：** タイムアウトになったアクティビティを処理するには、事前にバックグラウンド・エンジンを設定する必要があります。2-40 ページの「[手順 WF-8 バックグラウンドのワークフロー・エンジンの設定](#)」を参照してください。

---

## 単一アクティビティへの複数トランジションの作成

プロセス・ダイアグラムで、1 つのアクティビティに至る複数のトランジションを作成できます。このような複数のトランジションは、プロセスが 1 つのノードに進む流れが複数あっても、そのノードを実行するのは 1 度のみであることを示す場合があります。

また、複数のトランジションは、そのアクティビティがループの開始点になるため、そこに複数回進むことを示す場合もあります。このような場合は、そこに到達するたびにそのアクティビティを再実行します。

アクティビティの「再開封時」フラグによって、そのアクティビティに複数回到達した場合に、それを再実行するかどうかが決まります。これは、ループのピボット・アクティビティに関して重要なフラグです。「再開封時」は、アクティビティの「詳細」プロパティ画面で初期設定されます。ただし、プロセスでアクティビティを使用するたびに、「ノード」プロパティ画面でそのノードに対する「再開封時」の設定を変更できます。プロセスがループ内を進む回数を制御する初期アクティビティとして、標準の「ループ・カウンタ」アクティビティも使用できます。8-11 ページの「[ループ](#)」および 6-6 ページの「[ループ・カウンタ](#)」アクティビティ」を参照してください。

---

**提案：** 並列分岐からの複数の入力トランジションがある場合は、常に AND、OR またはカスタムの「結合」アクティビティを使用して、これらの分岐をマージする必要があります。特に、並列分岐をマージした後で、プロセスにループを作成する場合は、この操作が必要です。「結合」アクティビティを使用して並列分岐をマージし、後続のアクティビティをループの開始点として指定すると、エンジンによって実行されるプロセスをさらに簡素化して作成できます。6-2 ページの「標準アクティビティ」を参照してください。

---

### 「開始」アクティビティと「終了」アクティビティの指定

各プロセスには、そのプロセスの開始点を識別する「開始」アクティビティが必要です。論理的にプロセスを開始するノードであれば、どんなノードでも「開始」アクティビティに指定できます。プロセスの開始時に、ワークフロー・エンジンは入力トランジションがない（そのアクティビティを指す矢印がない）「開始」アクティビティから開始します。複数の「開始」アクティビティがある場合は、実行可能な各「開始」アクティビティが実行され、「終了」の結果に到達するまでプロセス内を進みます。実行できる「開始」アクティビティは、任意の順序で実行されます。プロセスには、それぞれ「終了」アクティビティを持つ複数の分岐が含まれていることがあります。ワークフロー・エンジンが「終了」アクティビティに到達すると、まだ処理中の並列分岐があっても、プロセス全体が終了します。

「終了」アクティビティは、プロセスの完了結果を表す結果を戻す必要があります。結果は、そのプロセス・アクティビティの結果タイプの値の 1 つです。

「プロセス」ウィンドウでは、「開始」アクティビティには小さい緑の矢印マーク、「終了」アクティビティにはアクティビティ・ノードのアイコンの右下隅に赤の矢印マークが付いた状態で表示されます。

### プロセスの実行

ワークフロー・プロセスは、アプリケーションがワークフロー・エンジンの `CreateProcess()` および `StartProcess()` API をコールするか、ビジネス・イベント・システムのサブスクリプションがプロセスを起動するイベントを送信したときに開始されます。サブプロセスは、ワークフロー・エンジンがサブプロセスを表すプロセス・アクティビティに進んだときに開始されます。

ビジネス・イベント・システムを使用してワークフロー・プロセスを起動するには、次の手順に従います。

1. ビジネス・イベントを定義します。
2. このビジネス・イベントに対するサブスクリプションを定義します。サブスクリプションのプロパティには、起動するワークフローの項目タイプとプロセスを指定します。

デフォルトでは、起動するワークフロー・プロセスの項目キーとして、イベント・キーが使用されます。カスタム・ルールに基づいて項目キーを生成する場合は、任意の項目

キーを使用して相関 ID をイベント・メッセージに生成する関数を作成し、サブスクリプションのルール関数として割り当てます。

3. ワークフロー・プロセスの起動場所で、Raise() API をカスタム・アプリケーション・コードに追加します。

**関連項目：**

8-20 ページ [「Workflow Engine API」](#)

8-268 ページ [「Raise」](#)

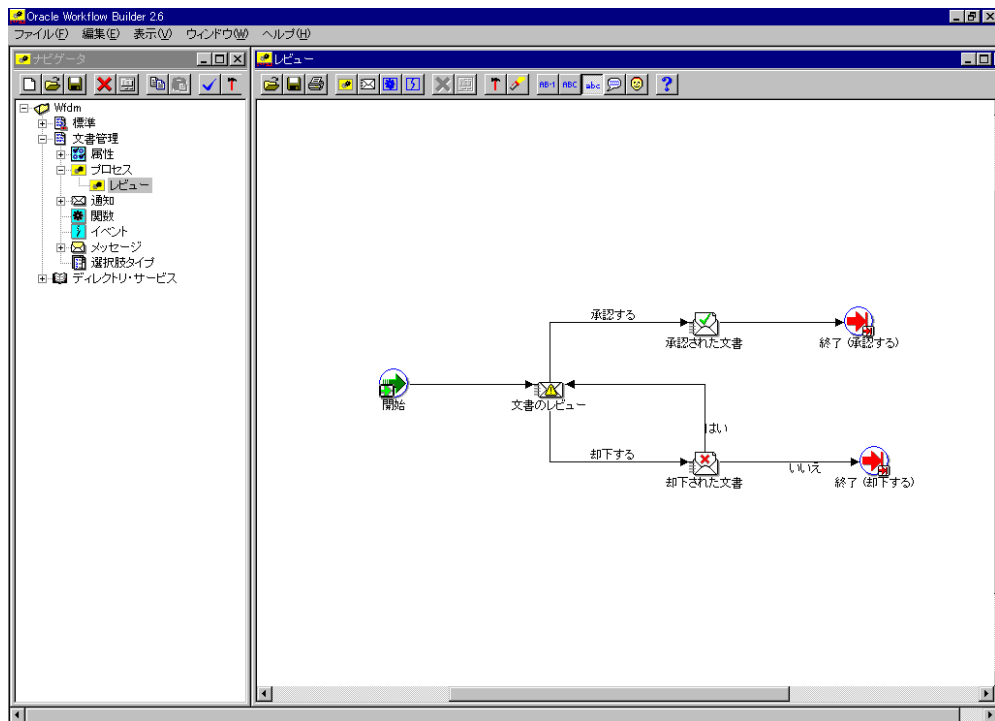
13-2 ページ [「ビジネス・イベントの管理」](#)

## プロセスの図示

ここでは、「プロセス」ウィンドウでワークフロー・プロセスを作成して定義する方法について説明します。

- 5-6 ページ [「ワークフロー・プロセスへのノードの追加」](#)
- 5-8 ページ [「プロセスのノードの定義」](#)
- 5-13 ページ [「イベント・ノードのイベントの詳細の定義」](#)
- 5-17 ページ [「アクティビティ属性値の定義」](#)
- 5-18 ページ [「トランジションの作成と編集」](#)
- 5-20 ページ [「プロセス概要の表示」](#)
- 5-21 ページ [「プロセスの印刷」](#)
- 5-21 ページ [「プロセス・ダイアグラムをクリップボードにコピー」](#)
- 5-21 ページ [「プロセス定義の検証」](#)

➤ ワークフロー・プロセスへのノードの追加



1. プロセス・ダイアグラムの作成を開始するには、最初にプロセス・アクティビティの「プロセス」ウィンドウを表示します。「プロセス」ウィンドウは、次のいずれかの方法でオープンできます。

- ナビゲータ・ツリーで、事前定義済のプロセス・アクティビティをダブルクリックします。
- 事前定義済のプロセス・アクティビティを選択して、[Ctrl]+[E] キーを押します。
- 事前定義済のプロセス・アクティビティを選択し、「編集」メニューから「プロセスの詳細」を選択します。
- クイック・スタート・ウィザードを使用して、新規のプロセス・アクティビティを作成します。3-19 ページの「[クイック・スタート・ウィザードの使用](#)」を参照してください。

「プロセス」ウィンドウが表示され、ウィンドウ・タイトルにプロセス名が表示されます。

2. 次のいずれかの方法を使用して、プロセスに新規ノードを作成します。

- ナビゲータ・ツリーから「プロセス」ウィンドウへ、通知、関数、イベントまたはプロセス・アクティビティをドラッグ・アンド・ドロップします。ドラッグするアクティビティは、ドロップ先のプロセスと同じデータ・ストアに属している必要があります。

---

**注意：** ドロップ先のプロセスと異なるデータ・ストアにあるアクティビティをプロセスにドラッグする場合は、最初にそのアクティビティが属する項目タイプを、このプロセスと同じデータ・ストアにコピーしてください。

---

- 「新規関数」、「新規プロセス」、「新規イベント」または「新規通知」のツールバー・ボタンを選択して、新規のアクティビティを作成します。
  - カーソルを「プロセス」ウィンドウに置き、マウスの右ボタン・メニューから「アクティビティ作成」を選択して新規のアクティビティ・ノードを作成します。
3. また、マウスの右ボタン・メニューを使用して新規ノードを作成することもできます。新規の関数、通知、イベントまたはプロセスを作成できます。「アクティビティ」プロパティ画面が表示され、このノードのアクティビティを選択します。5-8 ページの「[プロセスのノードの定義](#)」を参照してください。
  4. 「プロセス」ウィンドウで、カーソルをアクティビティに合せると、そのアクティビティに関する情報を表示できます。「ラベル名」、「内部名」、「表示名」、「コメント」および「実行者」は、ポップアップ・ヒントの形式で表示されます。
  5. 「プロセス」ウィンドウでアクティビティ・ノードをシングル・クリックすると、ナビゲータ・ツリーが拡張され、選択したノードのマスター・アクティビティがハイライト表示されます。
  6. マウスの右ボタンを押したままソース・アクティビティから宛先アクティビティへドラッグし、2つのアクティビティ・ノード間にトランジション（矢印）を作成します。
  7. ソース・アクティビティに結果コードが関連付けられていなければ、デフォルトではトランジションにラベルは表示されません。この種のトランジションのラベルを表示するように特定して選択すると、「デフォルト」というラベルが表示されます。5-18 ページの「[トランジションの作成と編集](#)」を参照してください。

ソース・アクティビティに結果コードが関連付けられている場合は、宛先アクティビティへのトランジションを作成しようとする、選択肢の値リストが表示されます。トランジションに割り当てる値を選択します。また、アクティビティから他のトランジションの結果と一致しない結果が戻された場合や、アクティビティからなんらかの結果が戻された場合、またはアクティビティがタイムアウトになった場合のトランジションを、それぞれ値「デフォルト」、「任意」または「タイムアウト」を選択して定義することもできます。

ナビゲータ・ツリーから「プロセス」ウィンドウの既存のトランジションに選択肢コードをドラッグ・アンド・ドロップし、そのトランジションの結果を変更することもでき

ます。ドラッグ・アンド・ドロップする選択肢コードは、置き換える選択肢コードと同じデータ・ストアに属し、同じ選択肢タイプである必要があります。

8. プロセス・ダイアグラムで、複数のアクティビティ・ノードとトランジションを含む領域全体を選択し、[Ctrl] または [Shift] キーを押したまま選択範囲を「プロセス」ウィンドウの別の位置にドラッグして、選択範囲のコピーを作成できます。

---

**注意：** Oracle Workflow では、1つのプロセス階層内でサブプロセス・アクティビティを複数回使用することはできません。1つのプロセス内でサブプロセスを2回以上使用する場合は、必要なインスタンスごとにサブプロセスの個別コピーを作成する必要があります。

---

9. 「表示」メニューで「スナップのグリッド」をオンにして、ダイアグラムの完成時にアクティビティ・アイコンをグリッドに貼り付ける必要があります。デフォルトでは、「スナップのグリッド」はオンになっており、設定を変更するとその設定がデフォルトになります。

### 関連項目：

A-9 ページ [「プロセス」ウィンドウのツールバー](#)

## ► プロセスのノードの定義

1. プロセス・アクティビティの「プロセス」ウィンドウをオープンします。
2. 新規の関数、通知、イベントまたはプロセス・ノードを作成するには、まず「プロセス」ウィンドウのツールバーから「新規関数」、「新規通知」、「新規イベント」または「新規プロセス」のアイコンを選択します。次に、「プロセス」ウィンドウ内で、この新規ノードを配置する位置をクリックします。新規ノードのプロパティ画面が表示されます。

---

**注意：** 新規ノードを作成するには、事前に定義されたアクティビティを、ナビゲータ・ツリーから「プロセス」ウィンドウにドラッグ・アンド・ドロップする方法もあります。この方法では、ノードのプロパティ画面に事前定義済の情報が自動的に作成されます。ノードのプロパティ画面を編集する場合は、そのノードをダブルクリックして手順5に進んでください。

---

3. 「項目タイプ」フィールドで、このアクティビティ・ノードを関連付ける項目タイプを選択します。
4. 次のどちらかの方法を選択し、ノードの残りの情報を定義します。
  - 事前に定義されたアクティビティの内部名または表示名を選択します。Oracle Workflow Builder により、「ナビゲータ」ウィンドウに表示されたマスター・アクティビティから、すべてのフィールドに事前定義済の情報が移入されます。



- または、「新規作成」ボタンを選択して新規のアクティビティを定義します。プロパティ画面の他のタブに必要な事項を入力する場合は、次の項を参照してください。
  - 「プロセス」： 4-54 ページ「プロセス・アクティビティの作成」
  - 「関数」： 4-47 ページ「関数アクティビティの作成」
  - 「通知」： 4-45 ページ「通知アクティビティの作成」
  - 「イベント」： 4-51 ページ「イベント・アクティビティの作成」
  - 「詳細」： 4-56 ページ「オプションのアクティビティ詳細の定義」
  - 「ロール」： このタブの情報は、現時点ではサポートされていません。
  - 「アクセス」： 4-17 ページ「オブジェクトのアクセス・レベルの設定」

**注意：** これらのタブで行った変更はすべて、マスター・アクティビティに自動的に伝播されるため、そのアクティビティの他のインスタンスがすべて影響を受けます。ただし、「ノード」と「ノード属性」タブ、およびイベント・アクティビティの「イベントの詳細」タブで行った変更は、現在のノード・アクティビティだけに適用されます。

5. 「ノード」タブを選択し、このノードに固有の情報を指定します。ノードのラベルを指定します。アクティビティは任意のプロセスで複数回使用できるため、「ラベル」フィールドを使用すると、プロセス内で特定のアクティビティのインスタンスに一意の名前を指定できます。デフォルトでは、ラベル名はアクティビティ名ですが、そのアク

ティビティがプロセスで複数回使用されると、アクティビティ名の後ろに -N が追加されます。N は、使用中のアクティビティの N 番目のインスタンスを表します。

---

**注意：** ほとんどの Oracle Workflow API をコールする場合に、アクティビティ名ではなくアクティビティのラベル名を渡す必要があります。8-20 ページの「[Workflow Engine API](#)」を参照してください。

---

6. 「開始」または「終了」を選択し、現行ノードがプロセスの「開始」アクティビティか「終了」アクティビティかを指定します。どちらでもない場合、デフォルトは「標準」です。プロセスには、複数の「開始」ノードと「終了」ノードを含めることができます。

「開始」アクティビティには（開始）というマークが付き、アクティビティ・アイコンに小さい緑の矢印が表示されます。「終了」アクティビティには（終了）というマークが付き、アクティビティ・アイコンに赤の矢印が表示されます。

---

**注意：** 「開始 / 終了」フィールドは、デフォルトではすべてのアクティビティ・ノードについて常に「標準」に設定されます。「Standard Start」または「Standard End」アクティビティを使用する場合にも、「開始 / 終了」フィールドは手動でそれぞれ「開始」または「終了」に編集する必要があります。

---

7. 「終了」ノードの場合、プロセス・アクティビティ全体に結果タイプが関連付けられているときは、プロセスの最終結果の値も選択する必要があります。プロセスの最終結果の値リストは、プロセス・アクティビティの結果タイプとして指定された選択肢タイプから導出されます。
8. このノードに関するコメントを入力できます。
9. 応答が必要な通知、他のプロセス内のサブプロセスであるプロセス・アクティビティ、イベント・アクションが「受信」のイベント・アクティビティ、またはブロック関数アクティビティの場合は、指定した期限までに完了する必要があるかどうかを指定します。指定した期限までにアクティビティが完了しなかった場合、親プロセスを別のアクティビティに進めるようにリダイレクトできます。5-3 ページの「[タイムアウト・トランジション](#)」を参照してください。

アクティビティを指定期限までに完了する必要がなければ、「タイムアウトなし」を選択します。

固定の相対時間までにアクティビティを完了する必要がある場合は、「相対時間」を選択します。日、時間、分の任意の組合せを指定して、アクティビティがタイムアウトとなる期限を指定します。入力した値は、アクティビティ開始日からの分単位の相対オフセット値として解釈されます。相対タイムアウト値を 0 にすると、タイムアウトなしになります。

実行時に動的に算出される一定の相対時間までにアクティビティを完了する必要がある場合は、「項目属性」を選択します。最初に、算出されるタイムアウト値を格納する数値タイプの項目属性を作成し、その項目属性をここで参照する必要があるため注意してください。4-2 ページの「[項目タイプ属性](#)」および 4-8 ページの「[項目タイプまたはアクティビティ属性の定義](#)」を参照してください。

---

**注意：** この属性に格納される動的タイムアウト値は、アクティビティ開始日からの分単位の相対オフセット値として解釈されます。相対タイムアウト値を NULL または 0 にすると、タイムアウトなしになります。

---

10. 通知アクティビティ・ノードの場合、またはイベント・アクションが「送信」のイベント・アクティビティ・ノードの場合は、そのアクティビティのメッセージに割り当てられた優先度を上書きできます。メッセージのデフォルト優先度を維持する場合は、「デフォルト」を選択してください。

デフォルトの優先度を、新しく指定する優先度で上書きする場合は「定数」を選択します。

デフォルトの優先度を、実行時に動的に決定される新しい優先度で上書きする場合は「項目属性」を選択します。最初に、算出される優先度値を格納する数値タイプの項目属性を作成し、その項目属性をここで参照する必要があるため注意してください。4-2 ページの「[項目タイプ属性](#)」および 4-8 ページの「[項目タイプまたはアクティビティ属性の定義](#)」を参照してください。

---

**注意：** 算出される優先度値は、1 ～ 99 の任意の数値です。Oracle Workflow ではこの数値が自動的に優先度 1 ～ 33= 高、34 ～ 66= 標準、67 ～ 99= 低に変換されます。

---

11. 通知アクティビティ・ノードの場合は、アクティビティの実行者を指定します。実行者とは、通知送信先のロールです。固定のロール名、または実行時にロールを動的に決定する項目タイプ属性を選択できます。最初に、算出されるロール名を格納するロール・タイプの項目属性を作成し、その項目属性をここで参照する必要があるため注意してください。4-2 ページの「[項目タイプ属性](#)」および 4-8 ページの「[項目タイプまたはアクティビティ属性の定義](#)」および 5-25 ページの「[ロール](#)」を参照してください。

---

---

**注意：**「実行者」の「タイプ」を「定数」に設定し、データベースに接続してロールをロードしている場合は、「実行者」ドロップ・ダウン・リストから固定のロール名を選択できます。データベースへの接続をオープンせずにファイルのデータ・ストアで作業している場合は、「実行者」フィールドに有効なロールの表示名を直接入力できます。このファイルをデータベースにアップロードすると、ロールは入力したロール表示名に基づいて、データベースに格納されている適切なロール・データに解決されます。

---

---

---

---

**注意：** 通知をマルチ・ユーザーのロールに割り当てると、実際に通知に応答するロールの各ユーザーが追跡されます。8-216 ページの「[Respond](#)」を参照してください。

---

---

---

---

**注意：** Oracle Workflow Builder では、どんなタイプのノード・アクティビティにも実行者を指定できますが、Oracle Workflow では通知アクティビティ・ノードに対する実行者の値のみが考慮されます。

---

---

12. 「適用」を選択して変更内容を保存するか、「OK」を選択し、変更内容を保存してプロパティ画面を閉じるか、「キャンセル」を選択し、変更を取り消してプロパティ画面を閉じます。  
  
プロパティ画面を保存して閉じると、「プロセス」ウィンドウの指定した位置に、アクティビティ・ノードが表示されます。これが新規作成したアクティビティであれば、ナビゲータ・ツリーの適切なブランチに、対応するマスター・アクティビティも作成されます。
13. ノードがイベント・アクティビティの場合は、「イベントの詳細」タブを選択して必要なイベント情報を追加できます。5-13 ページの「[イベント・ノードのイベントの詳細の定義](#)」を参照してください。
14. ノードが関数、通知またはイベント・アクティビティのときに、そのアクティビティにアクティビティ属性が指定されている場合は、「ノード属性」タブを選択してアクティビティ属性に値を割り当てることができます。5-17 ページの「[アクティビティ属性値の定義](#)」を参照してください。
15. ノードがプロセス・アクティビティの場合は、プロセス・アクティビティ・アイコンの右上隅に、小型のサブプロセス・アイコンが表示されます。サブプロセス・アイコンは、そのノードがプロセス・ダイアグラム内のサブプロセスであることを示します。

**関連項目：**

3-7 ページ「ナビゲータ・ツリーでのオブジェクトの検索」

3-9 ページ「プロパティ画面の「編集」ボタンの使用」

**▶ イベント・ノードのイベントの詳細の定義**

イベント・アクティビティに定義したイベント・アクションに基づいて、イベント・ノードにイベントの詳細を定義します。項目タイプ属性を使用して、格納または取得するイベントの詳細情報を定義します。一部の項目は、必ず定義しなければなりません。事前定義済の項目タイプ属性を参照するには、適切なタイプの項目タイプ属性をあらかじめ作成する必要があります。イベントの詳細に使用する項目タイプ属性は、イベント・アクティビティが関連付けられている項目タイプと同じ項目タイプに関連付ける必要があります。4-2 ページの「項目タイプ属性」および 4-8 ページの「項目タイプまたはアクティビティ属性の定義」を参照してください。

1. イベント・アクティビティ・ノードのプロパティ画面を表示します。「イベントの詳細」タブを選択します。

ナビゲータ・コントロールのプロパティ

イベント | 詳細 | ロール | アクセス | ノード | イベントの詳細 | ノード属性 |

メッセージ・データの受信先:

イベント名(E) <なし> [編集]

イベント・キー(K) <なし> [編集]

イベント・メッセージ(M) <なし> [編集]

[OK] [キャンセル] [適用(A)] [ヘルプ]

2. イベント・アクションが「受信」のアクティビティの場合、次のイベントの詳細を入力します。
  - 「イベント名」: (オプション) テキスト・タイプの項目タイプ属性を選択します。ノードが受信したイベント名を格納します。

---

**注意：** 受信イベント・アクティビティは、イベント・フィルタとして指定したイベント名と一致するイベントだけを受信できます。4-51 ページの「[イベント・アクティビティの作成](#)」を参照してください。

---

- 「イベント・キー」：（オプション）テキスト・タイプの項目タイプ属性を選択します。ノードが受信したイベント・キーを格納します。
- 「イベント・メッセージ」：（オプション）イベント・タイプの項目タイプ属性を選択します。ノードが受信したイベント・メッセージを格納します。

---

**注意：** 受信アクティビティがイベントを受信すると、ワークフロー・エンジンは、指定されたイベント名、イベント・キーおよびイベント・メッセージを項目タイプ属性に格納します。パラメータに対応する属性がまだ存在しない場合は、新しい項目属性を作成して、イベント・メッセージのパラメータ・リストに含まれるすべてのパラメータをプロセスの項目タイプ属性として設定します。4-56 ページの「[オプションのアクティビティ詳細の定義](#)」を参照してください。

---

また、イベントが別のワークフロー・プロセスの呼出しイベント・アクティビティによって発生した場合は、そのプロセスの項目タイプと項目キーがイベント・メッセージ内のパラメータ・リストに追加されます。ワークフロー・エンジンは、イベントを受信するプロセスの親として、指定されたプロセスを自動的に設定し、既存の親の設定を上書きします。8-81 ページの「[SetItemParent](#)」を参照してください。

The screenshot shows a dialog box titled 'ナビゲータコントロールのプロパティ' (Navigator Control Properties). It has several tabs: 'イベント' (Event), '詳細' (Details), 'ロール' (Role), 'アクセス' (Access), 'ノード' (Node), 'イベントの詳細' (Event Details), and 'ノード属性' (Node Properties). The 'イベントの詳細' tab is selected. Under the heading 'イベント呼出し:' (Event Callout:), there are three rows of input fields. The first row is for 'イベント名' (Event Name), with a 'タイプ(T)' (Type) dropdown set to '定数' (Constant) and a '値(V)' (Value) text box. The second row is for 'イベント・キー(K)' (Event Key), with a dropdown set to '<なし>' (None) and an '編集' (Edit) button. The third row is for 'イベント・データ(D)' (Event Data), with a dropdown set to '<なし>' (None) and an '編集' (Edit) button. At the bottom of the dialog are four buttons: 'OK', 'キャンセル' (Cancel), '適用(A)' (Apply), and 'ヘルプ' (Help).

3. イベント・アクションが「呼出し」のアクティビティの場合、次のイベントの詳細を入力します。

- 「イベント名」： ノードが呼び出すイベントの名前。定形のイベント名を指定するか、テキスト・タイプの項目タイプ属性を選択します。後者の場合、実行時にイベント名を動的に決定できます。

---

**注意：** 呼び出せるイベントは1つだけです。イベント・グループは呼び出せません。

---

- 「イベント・キー」： テキスト・タイプの項目タイプ属性を選択します。ノードが呼び出したイベントのイベント・キーを格納します。
- 「イベント・データ」： (オプション) テキスト・タイプの項目タイプ属性を選択します。ノードが呼び出したイベントのイベント・データを格納します。

テキスト属性に入力できるデータ長は、最大 4000 バイトです。イベント・データが 4000 バイトを超える場合は、テキスト属性でイベント・データを渡さずに、イベント定義にジェネレート関数を割り当ててイベント・データを生成する必要があります。13-6 ページの「[イベントの定義](#)」を参照してください。

---

**注意：** 「呼出し」アクティビティでは、「イベント名」および「イベント・キー」が必須です。

---

---

**注意：** 呼出しイベント・アクティビティでは、これらのイベントの詳細以外に、ノードのアクティビティ属性を使用して、イベント・メッセージのパラメータ・リストに追加するパラメータを指定できます。別のプロセスがイベント・メッセージを後で受信した場合、ワークフロー・エンジンはイベント・パラメータをそのプロセスの項目タイプ属性として設定します。5-17 ページの「[アクティビティ属性値の定義](#)」を参照してください。

---

また、呼出しイベント・アクティビティでは、現行のワークフロー・プロセスの項目タイプおよび項目キーが、イベント・メッセージのパラメータ・リストに自動的に追加されます。別のプロセスがイベント・メッセージを後で受信した場合、ワークフロー・エンジンはその項目タイプと項目キーを使用して、イベントを受信するプロセスの親として、イベントを呼び出したプロセスを自動的に設定します。8-81 ページの「[SetItemParent](#)」を参照してください。

ナビゲータ・コントロールのプロパティ

イベント | 詳細 | ロール | アクセス | ノード | イベントの詳細 | ノード属性

メッセージ送信:

イベント・メッセージ(M) <なし> [編集]

イベント名  
タイプ(T) 定数  
値(V) [ ]

イベント・キー(K) <なし> [編集]

送信エージェント  
タイプ(Y) 定数  
値(A) [ ]

宛先エージェント  
タイプ(P) 定数  
値(L) [ ]

[OK] [キャンセル] [適用(A)] [ヘルプ]

4. イベント・アクションが「送信」のアクティビティの場合、次のイベントの詳細を入力します。
  - 「イベント・メッセージ」： イベント・タイプの項目タイプ属性を選択します。ノードが送信したイベント・メッセージを格納します。
  - 「イベント名」： (オプション) ノードが送信したイベントの名前を入力します。定形のイベント名を指定するか、テキスト・タイプの項目タイプ属性を選択します。後者の場合、実行時にイベント名を動的に決定できます。ここで入力したイベント名によって、イベント・メッセージに入力されているイベント名の値が上書きされます。
  - 「イベント・キー」： (オプション) テキスト・タイプの項目タイプ属性を選択します。ノードが送信したイベントのイベント・キーが格納されます。ここで入力したイベント・キーによって、イベント・メッセージに入力されているイベント・キーの値が上書きされます。
  - 「送信エージェント」： (オプション) ノードがイベントを送信したときのアウトバウンド・エージェントを入力します。次の書式を使用して、エージェントのエージェント名およびシステム名を指定します。

`<agent_name>@<system_name>`

定形の送信エージェント名を指定するか、テキスト・タイプの項目タイプ属性を選択します。後者の場合、実行時に送信エージェント名を動的に決定できます。ここで入力したアウトバウンド・エージェントによって、イベント・メッセージに入力されている送信エージェントの値が上書きされます。



- 「宛先エージェント」：（オプション）ノードがイベントを送信したときのインバウンド・エージェントを入力します。次の書式を使用して、エージェントのエージェント名およびシステム名を指定します。

<agent\_name>@<system\_name>

定形の宛先エージェント名を指定するか、テキスト・タイプの項目タイプ属性を選択します。後者の場合、実行時に宛先エージェント名を動的に決定できます。ここで入力したインバウンド・エージェントによって、イベント・メッセージに入力されている宛先エージェントの値が上書きされます。

---

**注意：** 送信イベント・アクティビティでは、「イベント・メッセージ」が必須です。また、イベント・メッセージ内に宛先エージェントまたは送信元エージェントを追加するか、このノードのイベントの詳細に宛先エージェントまたは送信エージェントを指定する必要があります。インバウンド・エージェントおよびアウトバウンド・エージェントのどちらも指定しない場合は、イベントを送信できません。

---

---

**注意：** 関連 ID がイベント・メッセージに指定されていない場合は、Oracle Workflow によって関連 ID がプロセスの項目キーに自動的に設定されます。

---

5. 「適用」を選択して変更内容を保存するか、「OK」を選択し、変更内容を保存してプロパティ画面を閉じるか、「キャンセル」を選択し、変更を取り消してプロパティ画面を閉じます。

**関連項目：**

3-9 ページ「プロパティ画面の「編集」ボタンの使用」

4-42 ページ「イベント・アクティビティ」

4-51 ページ「イベント・アクティビティの作成」

► **アクティビティ属性値の定義**

関数または通知アクティビティのアクティビティの属性値は、そのアクティビティがコールする PL/SQL のストアド・プロシージャによって使用されます。呼出しイベント・アクティビティのアクティビティ属性値は、パラメータとしてイベント・メッセージのパラメータ・リストに追加されます。4-8 ページの「項目タイプまたはアクティビティ属性の定義」を参照してください。



1. アクティビティ・ノードのプロパティ画面を表示します。「ノード属性」タブを選択します。
2. 属性を選択します。
3. 「値」フィールドに、この属性の値を入力します。値には、定数、または項目タイプ属性に保存されている値を使用できます。

入力する値は、アクティビティ属性と同じデータ型で、そのアクティビティに関連した PL/SQL 関数で定義されている、実際のアクティビティ・パラメータ自体のデータ型と一致している必要があります。属性タイプは、各属性の名前、説明、値の型および値とともに「attributes summary」フィールドに表示されます。

4. 「適用」を選択して変更内容を保存するか、「OK」を選択し、変更内容を保存してプロパティ画面を閉じるか、「キャンセル」を選択し、変更を取り消してプロパティ画面を閉じます。

## ▶ トランジションの作成と編集

1. マウスの右ボタンを押したまま、ソース・アクティビティから宛先アクティビティへドラッグし、2つのアクティビティ間にトランジションを作成します。

---

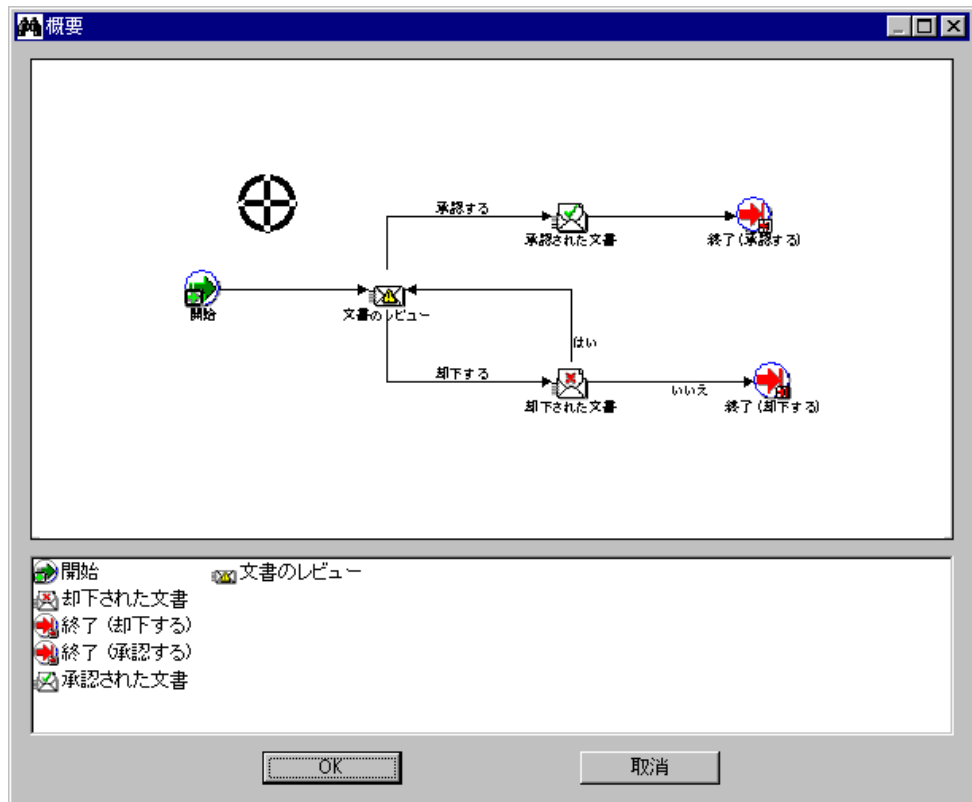
**注意：** 重複するトランジションは、単一の重複しないトランジションとは異なる色で表示されます。

---

2. トランジションを編集するには、そのトランジションを選択します。

3. トランジションのラベルを移動するには、単にマウスでラベルを選択して新しい位置にドラッグします。ラベルがトランジションに貼り付けられます。
4. マウスでトランジションを選択して右ボタンをクリックすると、いつでも次の編集オプション・メニューを表示できます。
  - 「トランジションの削除」： 選択したトランジションが削除されます。
  - 「ロック」： さらに編集するかどうかに応じて、トランジションのロックとロック解除を切り替えます。トランジションがロックされていると、トランジションに対する支点の追加や削除はできませんが、トランジションの削除は可能です。
  - 「ラベルを非表示」： トランジション・ラベルの表示と非表示が切り替わります。
  - 「直線化」： トランジションを曲げている余分な支点が削除され、トランジションが直線になります。
  - 「結果 ...」： トランジションに結果が割り当てられている場合に、このオプションを使用してトランジションの結果ラベルを変更します。ユーザーが選択できる結果ラベルのリストが、別のメニューに表示されます。
5. トランジションを曲げるには、そのトランジションを選択し、マウスの左ボタンを押したままドラッグして支点を作成します。支点を自由に移動して、トランジションの線を曲げることができます。
6. ソース・アクティビティ・ノードにループするトランジションを、次のどちらかの方法で作成できます。
  - マウスの右ボタンを押したまま、ソース・アクティビティからそれ自身へドラッグして、自己ループを作成します。
  - ソース・アクティビティ・ノードから、他の任意のアクティビティ・ノードへのトランジションを作成します。支点を追加してトランジションの線を折り曲げます。次にトランジションの矢印の先を選択し、ソースのアクティビティ・ノードにドラッグします。必要に応じてさらに支点を作成し、ループのトランジションを見やすくします。
7. 1つの支点をトランジションから削除するには、その支点を選択し、他の支点までドラッグして2つの点を結合します。

➤ プロセス概要の表示



1. カーソルをプロセス・ウィンドウにあわせて、マウスの右ボタンで表示するメニューから「概要」を選択します。
2. プロセスの「概要」ウィンドウが表示されます。  
ウィンドウの上部ペインにはプロセス全体のサイズが縮小されたスケッチが表示され、下部ペインにはプロセスに含まれるアクティビティのリストが表示されます。
3. 「概要」ウィンドウのサイズは、プロセス・スケッチが見やすいように変更できます。
4. プロセス・スケッチ・ペインに、ドラッグできるように十字型のカーソルが表示されます。十字型のカーソルを使用して、プロセスのうち「プロセス」ウィンドウに表示する部分を選択します。
5. 下部ペイン内でアクティビティをクリックし、十字型のカーソルをスケッチ内のそのアクティビティに合せます。「OK」を選択してウィンドウを閉じ、「プロセス」ウィンドウ内でそのアクティビティにジャンプします。

また、上部ペイン内で十字型のカーソルをドラッグしてダブルクリックし、「プロセス」ウィンドウ内で目的の領域にジャンプすることもできます。

### ▶ プロセスの印刷

1. 印刷するプロセスを含む「プロセス」ウィンドウを表示します。
2. 「プロセス」ウィンドウをアクティブ・ウィンドウとして選択し、「ファイル」メニューまたはマウスの右ボタンで表示するメニューから「ダイアグラムの印刷」を選択します。

「ダイアグラムの印刷」オプションを選択すると、プロセス・ダイアグラムはピクチャ（メタファイル）として取り込まれ、印刷用に正しいサイズに拡大され、プリンタに送信されます。ダイアグラムが大きい場合は、印刷時に複数ページにまたがる場合があります。ただし、使用中のプリンタ・ドライバによっては、「印刷」ダイアログ・ボックスで印刷時に1ページに収まるようにイメージを縮小できます。

---

**注意：** プロセス・ダイアグラムに使用されているフォントがプリンタで見つからない場合は、プリンタ・ドライバによって類似するフォントに置き換えられたり、テキストが印刷されないことがあります。

---

### ▶ プロセス・ダイアグラムをクリップボードにコピー

1. コピーするプロセスを含む「プロセス」ウィンドウを表示してアクティブにします。
2. 「編集」メニューまたはマウスの右ボタンで表示されるメニューから、「ダイアグラムのコピー」を選択します。

これで、プロセスがメタファイルとビットマップの形式でクリップボードにコピーされます。

3. プロセス・ダイアグラムのメタファイル版またはビットマップ版を別のアプリケーションのウィンドウに貼り付けるには、メタファイルまたはビットマップを貼り付ける方法を、そのアプリケーションのマニュアルで確認する必要があります。

ビットマップ・イメージを編集するには、ビットマップを編集できるアプリケーションにイメージを貼り付ける必要があります。

### ▶ プロセス定義の検証

1. 「ファイル」メニューから「検証」を選択し、現在選択されているデータ・ストアのすべてのプロセス定義を検証します。
2. 次のリストは、「検証」コマンドで実行される検証の例を示しています。
  - プロセスに、1つ以上の「開始」アクティビティと「終了」アクティビティがあるかどうかチェックされます。
  - プロセスに、そのプロセス自体がプロセス・アクティビティとして含まれていないかどうか検証されます。

- 1つのプロセス内で、同じサブプロセスを2度使用できないように制限されます。
- アクティビティの結果がすべて出力トランジションとしてモデル化されているかどうかを検証されます。アクティビティが完了し、その結果が出力トランジションに関連付けられておらず、そのアクティビティの「デフォルト」トランジションが存在しなければ、アクティビティは「エラー」になります。
- 終了ノードとしてマークされているアクティビティ・ノードに、出力トランジションがないかどうかを検証されます。
- 通知アクティビティの結果タイプが、そのメッセージの「RESULT」メッセージ属性に定義されている選択肢タイプと一致しているかどうかを検証されます。
- メッセージ本文でトークンの置換のために参照されるメッセージ属性が、メッセージ定義内に存在しているかどうかを検証されます。
- 別の項目タイプからオブジェクトを参照するプロセスについては、参照される項目タイプに関連付けられた前提条件の項目属性が存在するかどうかを検証されます。

---

**注意：** 定義を正常に実行できなくする潜在的な問題が含まれていないかどうかを識別できるように、新規のプロセス定義を作成した場合に、必ず検証する必要があります。

---

## Oracle Workflow Builder のフォントの変更

Oracle Workflow Builder のウィンドウで使用されるフォントを変更できます。変更は、プログラム内のすべてのウィンドウに適用されます。



► フォントの変更

1. 「表示」メニューから「フォント」を選択し、「フォント」プロパティ画面を表示します。
2. アイコンのラベルに使用するフォントを選択します。このフォントは、Oracle Workflow Builder のすべてのアイコンに使用されます。「サンプル」フィールドに、選択したフォントの表示イメージが表示されます。
3. フォント・スタイルとして「標準」、「太字」、「斜体」または「太字斜体」を選択します。フォントの中には、フォント・スタイルが制限されるものもあります。
4. 必要なフォント・サイズを指定します。フォントには、フォント・サイズが制限されるものもあります。
5. 「下線」または「取り消し線」チェック・ボックスをオンにして適用します。
6. 操作の完了後に、「OK」を選択します。これらのフォント設定は即時に有効になり、次回に Oracle Workflow Builder を起動するときにも使用されます。

## ワークフロー・プロセスのショートカット・アイコンの作成

Windows のデスクトップに、Oracle Workflow Builder へのショートカットを作成できます。ショートカットを使用すると、指定したデータ・ストアに自動的に接続し、そのデータ・ストアから特定の「プロセス」ウィンドウを開いて、Oracle Workflow Builder を起動できます。

### ► Oracle Workflow Builder のショートカットの作成

1. Oracle Workflow Builder を起動します。
2. 「ファイル」メニューから「オープン」を選択してデータ・ストアを開きます。
3. 必要であれば、「プロセス」のブランチを拡張し、1 つ以上のプロセス・アクティビティをダブルクリックして、そのプロセスのプロセス・ウィンドウを開きます。
4. 「ファイル」メニューから「ショートカットの作成」を選択します。
5. デスクトップに表示するショートカット名を入力します。
6. デスクトップ上で新規のショートカット・アイコンをダブルクリックすると、Oracle Workflow Builder が自動的に起動し、ショートカットの作成時に選択されていたデータ・ストアと、開いていたプロセス・ウィンドウが開きます。

ショートカットのデータ・ストアがデータベースの場合は、ショートカットをダブルクリックすると、データベースのパスワード入力を求めるプロンプトが表示されます。



# ルール

Oracle Workflow のルールは、Oracle Workflow ディレクトリ・サービスのデータベースに格納されています。現在、Oracle Workflow Builder では新規のワークフロー・ルールを作成できませんが、データベースに格納されているルールを表示して参照することはできます。

## ワークフロー・プロセスのルールの参照

ワークフロー・プロセスでルールが参照される方法の一例は、通知アクティビティをプロセスにノードとして組み込むときです。そのノードを実行者に割り当てる必要があります。実行者は、指定したルール、または動的にルールの値を戻す項目タイプ属性です。実行者にルールを割り当てるには、最初に Oracle Workflow データベースから Oracle Workflow Builder セッションにルールをロードする必要があります。2-20 ページの「[手順 WF-3 Oracle Workflow のディレクトリ・サービスの設定](#)」および 5-8 ページの「[プロセスのノードの定義](#)」を参照してください。

---

**注意：** 現在、Oracle Workflow Builder ではワークフロー・プロセス内でルールを参照できますが、特定のワークフロー・オブジェクトのプロパティ画面に表示される「ルール」タブは、将来のリリースまでサポートされません。「ルール」タブの目的は、ルールに特定のオブジェクトへのアクセス権を付与することです。

---

## アド・ホックのユーザーおよびルール

Oracle Workflow では、ワークフロー・プロセス内で新規のアド・ホック・ユーザーおよびルールを作成して、ディレクトリ・サービスに追加できます。そのためには、サーバー側で適切な WF\_DIRECTORY API をコールする関数アクティビティを定義し、その関数アクティビティをプロセス・ダイアグラムに含めます。7-3 ページの「[関数アクティビティがコールする PL/SQL プロシージャの標準 API](#)」および 8-124 ページの「[Workflow ディレクトリ・サービス API](#)」を参照してください。

### 関連項目：

5-25 ページ「[ルールのロード](#)」

5-27 ページ「[Oracle Workflow Builder での Builder ディレクトリ・サービスの表示](#)」

### ➤ ルールのロード

1. Oracle Workflow データベースに接続していない場合は、「ファイル」メニューから「オープン」を選択し、データベースに接続して項目タイプを開きます。
2. 「ファイル」メニューから「データベースからルールをロード」を選択します。「ルール選択」ウィンドウが表示されます。SQL 問合せ構文を使用し「ルール検索」フィールドに検索基準を入力して、ルールのサブセットを検索できます。また、検索基準を指定し

ないで「検索」を選択すると、すべてのロールを識別できます。「ロール選択」ウィンドウで、指定したロールが検索され、「問合せ結果」リスト・ボックスに表示されます。



3. 「問合せ結果」リストでロードするロールを選択し、「追加 >>」を選択して「ロードされたロール」リストに移動します。また、「すべて追加 >>」を選択すると、「問合せ結果」リスト内のすべてのロールが「ロードされたロール」リストに移動されます。「OK」を選択すると、選択したロールが **Oracle Workflow Builder** にロードされ、開いている項目タイプのワークフロー・オブジェクトに使用可能になります。

ロール情報の参照を必要とするワークフロー・オブジェクトの場合は、そのプロパティ画面に特定のフィールドが表示されます。これらのフィールドは、次の「ノード」プロパティ画面の例のように、データベースからロードしたロールのリストが表示されるドロップ・ダウン・フィールドです。

4. これらのドロップ・ダウン・フィールドからロールを選択するときに、そのフィールドの右側の「編集」ボタンを選択し、選択したロールのプロパティ・シートを表示することもできます。
5. 選択したロールの情報が、「ロール」プロパティ画面に読み取り専用で表示されます。

---

**注意：** 保存したプロセス定義を Oracle Workflow Builder で再度開くと、プロセス定義をファイルから開いたためにデータベースに接続されていない場合も、そのプロセスによって参照されるロール情報が自動的にロードされます。

---

### ► Oracle Workflow Builder での Builder ディレクトリ・サービスの表示

1. Oracle Workflow Builder でデータベースからロールをロードすると、ナビゲータ・ツリーにディレクトリ・サービス情報が表示されます。5-25 ページの「[ロールのロード](#)」を参照してください。
2. ナビゲータ・ツリーで、「ディレクトリ・サービス」ブランチを拡張します。データベースからロードした全ロールが表示されます。
3. ロールをダブルクリックし、次に示すようなロールに関する読み取り専用情報を表示します。現在、「ディレクトリ・サービス」ブランチでは、ロールの特定ユーザーは表示できないため注意してください。

ナビゲータ・コントロールのプロパティ

ロール

内部名(N)	JA		
表示名(N)	BLEWIS		
説明(D)	BLEWIS		
言語(L)	AMERICAN	地域(I)	AMERICA
電子メール・アドレス(E)	BLEWIS	FAX(F)	
通知環境設定(P)	HTML形式のメール		
ステータス(S)	ACTIVE	有効期限(O)	

OK キャンセル 適用(A) ヘルプ

---

## Oracle Workflow の事前に定義された アクティビティ

この章では、Oracle Workflow の事前に定義されたアクティビティの使用方法について説明します。

## 標準アクティビティ

Oracle Workflow には、プロセスの管理に使用できる一般的なアクティビティがいくつか用意されています。これらのアクティビティは、標準項目タイプに関連付けられていますが、ユーザーが定義するプロセスで使用できます。標準項目タイプは、Oracle Workflow Server に自動的にインストールされます。また、標準項目タイプには、`<ORACLE_HOME>/wf/data/<language>/` サブディレクトリにある `wfstd.wft` ファイルからもアクセスできます。

---

**注意：** 事前に定義されたアクティビティは、Oracle Applications および Oracle Self-Service Web Applications に付属の事前定義済のワークフローにも使用可能です。Oracle Applications に固有のワークフロー・アクティビティの詳細は、該当する Oracle Applications 製品のドキュメントまたはヘルプを参照してください。

---

---

**注意：** ドロップ先のプロセスと異なるデータ・ストアにあるアクティビティをプロセスにドラッグする場合は、最初にそのアクティビティが属する項目タイプを、このプロセスと同じデータ・ストアにコピーしてください。たとえば、`wfexample.wft` に格納されているプロセスを変更し、`wfstd.wft` に保存されている標準アクティビティをそのプロセスに追加するとします。最初に、両方のファイルを Oracle Workflow Builder でデータ・ストアとして開き、`wfstd` にある標準項目タイプをコピーして、`wfexample` データ・ストアに貼り付ける必要があります。これで、`wfexample` データ・ストアの標準アクティビティをプロセスにドラッグできます。

---

## And/Or アクティビティ

複数の並列分岐が 1 つのノードに進む場合は、そのノードが並列分岐のどれか 1 つが完了したときに先に進むのか、すべての並列分岐が完了したときに先に進むのかを指定できます。すべての分岐が完了してから継続する場合は、複数の分岐が集まるノードに「And」アクティビティを使用します。分岐のどれか 1 つが完了した場合にプロセスを継続する場合は、複数の分岐が集まるノードに「Or」アクティビティを使用します。

**And**                      そこに集まるすべての分岐からのアクティビティが完了したときに、完了します。PL/SQL プロシージャ `WF_STANDARD.ANDJOIN` をコールします。

**Or**                        そこに集まる分岐のうち 1 つ以上のアクティビティが完了したときに、完了します。PL/SQL プロシージャ `WF_STANDARD.ORJOIN` をコールします。

## 「比較」アクティビティ

「比較」アクティビティには、2つの数値、日付またはテキスト文字列を比較する標準的な方法が用意されています。

<b>日付の比較</b>	日付タイプの項目タイプ属性の値を一定の日付と比較するときに使用します。
<b>数値の比較</b>	数値タイプの項目タイプ属性の値を定数と比較するときに使用します。
<b>テキストの比較</b>	テキスト・タイプの2つの項目タイプ属性の値を比較するときに使用します。

「比較」アクティビティはすべて、WF\_STANDARD.COMPARE という PL/SQL プロシージャをコールします。

### アクティビティ属性

それぞれの「比較」アクティビティには、次の2つのアクティビティ属性があります。

- テスト値： 参照値と比較する定数値、日付またはテキスト文字列
- 参照値： タイプが「数値」、「日付」または「テキスト」の項目タイプ属性

「比較」アクティビティでは、結果コードに「比較」選択肢タイプを使用します。使用可能な値は、「より大きい」、「より小さい」、「等しい」、「NULL」（項目タイプ属性が NULL の場合）です。設定した値と項目タイプの属性値との比較方法に基づいて、ワークフロー・プロセスを進めることができます。5-17 ページの「アクティビティ属性値の定義」を参照してください。

## 「実行時間の比較」アクティビティ

「実行時間の比較」アクティビティには、プロセス実行時の経過時間を定数のテスト時間と比較する標準的な方法が用意されています。

「実行時間の比較」アクティビティは、WF\_STANDARD.COMPAREEXECUTIONTIME という PL/SQL プロシージャをコールします。

### アクティビティ属性

「実行時間の比較」アクティビティには、次の2つのアクティビティ属性があります。

- テスト実行時間： 経過した実行時間と比較する時間（秒）。
- 上位タイプ： 選択肢コードで、「ルート」または「上位」のどちらかの値を取得します。値「ルート」の場合は、現行のルート・プロセスで経過した実行時間とテスト時間が比較されます。値「上位」の場合は、すぐ上の上位プロセスで経過した実行時間とテスト時間が比較されます。この場合の上位プロセスは、サブプロセスでもかまいません。

このアクティビティでは、結果コードに「比較」選択肢タイプを使用します。使用可能な値は、「より大きい」、「より小さい」、「等しい」、「NULL」（テスト時間が NULL の場合）です。5-17 ページの「アクティビティ属性値の定義」を参照してください。

### 「待機」アクティビティ

「待機」アクティビティは、指定された期間のみプロセスを休止します。次のいずれかの期限まで待機できます。

- 指定日時
- 指定月日
- 指定曜日
- このアクティビティに到達後の一定時間

このアクティビティは、WF\_STANDARD.WAIT という PL/SQL プロシージャをコールします。

### アクティビティ属性

「待機」アクティビティには、次の 6 つのアクティビティ属性があります。

- 待機モード： この属性を使用して、待機時間の計算方法を指定します。次の待機モードから 1 つを選択できます。
  - － 絶対日付：「絶対日付」アクティビティ属性で指定された日付に達するまで、アクティビティを休止します。
  - － 相対時間：「相対時間」アクティビティ属性で指定された日数が経過するまで、アクティビティを休止します。
  - － 毎月何日：「毎月何日」アクティビティ属性で指定された月の特定日付まで、アクティビティを休止します。
  - － 毎週何曜日：「毎週何曜日」アクティビティ属性で指定された特定の曜日まで、アクティビティを休止します。
- 絶対日付：「待機モード」が「絶対日付」に設定されている場合は、絶対日付を入力します。
- 相対時間：「待機モード」が「相対時間」に設定されている場合は、相対時間を < 日数 >.< 日数（小数点以下）> の形式で入力します。たとえば、待機時間が半日（12 時間）の場合は、0.5 と入力します。
- 毎月何日：「待機モード」が「毎月何日」に設定されている場合は、リストから月の日を選択します。選択した日が今月はずでに過ぎている場合、アクティビティは翌月のその日まで待機します。



- 毎週何曜日: 「待機モード」が「毎週何曜日」に設定されている場合は、リストから曜日を選択します。選択した曜日が今週はすでに過ぎている場合、アクティビティは翌週のその曜日まで待機します。
- 毎日何時: 「毎日何時」アクティビティ属性を使用して「待機」アクティビティの休止期限を午前 0 時以外の時刻に設定しない限り、「待機」アクティビティは常に指定日の午前 0 時まで休止します。

**参照:** 5-17 ページの「アクティビティ属性値の定義」を参照してください。

## 「ブロック」アクティビティ

「ブロック」アクティビティでは、なんらかの外部プログラムまたは手動手順が完了し、**CompleteActivity Workflow Engine API** がコールされるまで、プロセスを休止できます。「ブロック」アクティビティを使用して、コンカレント・プログラムの完了など、なんらかの条件が満たされるまでプロセスを延期できます。プログラムが完了し、「ブロック」アクティビティでプロセスを再開する場合は、プログラムで **CompleteActivity** コールを発行するようにしてください。8-71 ページの「**CompleteActivity**」を参照してください。

このアクティビティは、WF\_STANDARD.BLOCK という PL/SQL プロシージャをコールします。

## 「スレッドの遅延」アクティビティ

「スレッドの遅延」アクティビティは、バックグラウンド・キューに対して後続のプロセス・スレッドを延期します。その場合に、対象スレッドの各アクティビティのコストを、ワークフロー・エンジンのしきい値を超えるように変更する必要はありません。このアクティビティは、スレッドがすでに遅延している場合でも、現行のデータベース・セッションを切断し、プロセスのスレッドを中断します。

このアクティビティは、WF\_STANDARD.DEFER という PL/SQL プロシージャをコールします。

## 「プロセスの開始」アクティビティ

「プロセスの開始」アクティビティを使用して、現行のプロセスから他のワークフロー・プロセスを開始できます。このアクティビティは、WF\_STANDARD.LAUNCHPROCESS という PL/SQL プロシージャをコールします。

### アクティビティ属性

「プロセスの開始」アクティビティには、次の 6 つのアクティビティ属性があります。

- 項目タイプ: 開始するプロセスの項目タイプ。ここでは、項目タイプの内部名を指定します。このアクティビティ属性には必ず値を指定します。

- 項目キー： 開始するプロセスの項目キー。値を指定しなければ、項目キーのデフォルトは `<current_item_type>:<current_item_key>-<n>` となります。  
`current_item_type` と `current_item_key` は現行のプロセス・インスタンスを表し、`n` は現行のプロセス・インスタンスで開始されるプロセス番号を表します。このプロセス番号は、1 が初期値となります。
- プロセス名： 開始するプロセスの内部名。プロセス名を指定しなければ、「プロセスの開始」アクティビティでは、開始するプロセスの項目タイプ・セレクト関数でプロセス名をチェックします。
- ユーザー・キー： 開始するプロセスのユーザー定義キー。
- 所有者： 開始するプロセスの所有者として指定されているロール。
- 即時に遅延： 「YES」または「NO」を選択し、開始するプロセスをバックグラウンド・エンジンに対して即時に遅延するかどうか決定します。デフォルトは「NO」で、プロセスがいったん開始されると、そのプロセスが完了するまで、またはプロセスのアクティビティのいずれかが延期されるまで続行します。5-17 ページの「アクティビティ属性値の定義」を参照してください。

### 「Noop」アクティビティ

「Noop」アクティビティは、何の処理も実行しないプロセスホルダ・アクティビティとして機能します。処理を実行しないノードを配置する場所に、このアクティビティを使用できます。このアクティビティをプロセスに含める場合には、将来このアクティビティを何に使用するかを忘れないように、表示名をわかりやすい名前に変更できます。このアクティビティは、WF\_STANDARD.NOOP という PL/SQL プロシージャをコールします。

### 「ループ・カウンタ」アクティビティ

「ループ・カウンタ」アクティビティを使用して、プロセス内の特定のパスを通過するワークフロー・エンジンのトランジション回数を制限します。「ループ・カウンタ」アクティビティの結果として「ループ」または「終了」を指定できます。

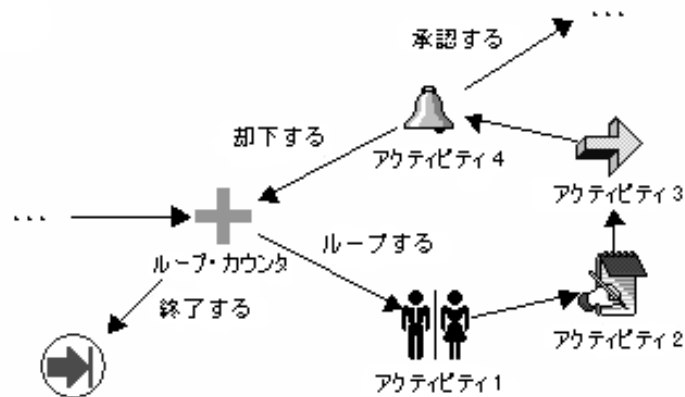
この「ループ・カウンタ」アクティビティは、WF\_STANDARD.LOOPCOUNTER という PL/SQL プロシージャをコールします。

#### アクティビティ属性

「ループ・カウンタ」アクティビティには、「ループの限度」というアクティビティ属性があります。ワークフロー・エンジンが「ループ・カウンタ」アクティビティを通過した回数が「ループの限度」で指定した値よりも小さければ、その「ループ・カウンタ」アクティビティは戻り値「ループ」を戻して完了し、ループ内の次のアクティビティへ進みます。ワークフロー・エンジンが「ループ・カウンタ」アクティビティを通過した回数が、「ループの限度」の値より大きければ、アクティビティは戻り値「終了」を戻して完了し、ループを終了して新しいアクティビティへ進みます。

たとえば、次のダイアグラムのように、ループ内の最初のアクティビティに「ループ・カウンタ」アクティビティを使用することもできます。「ループの限度」アクティビティ属性に指定する値に、そのループを通過できる回数を指定します。「ループ・カウンタ」アクティビティを通過した回数が、「ループの限度」に指定した値より大きければ、そのプロセスは「終了」トランジションに沿って指定されたアクティビティへ移動します。5-17 ページの「アクティビティ属性値の定義」を参照してください。

この例では、エンジンは「ループ・カウンタ」アクティビティからループ内のアクティビティ 1、2、3 および 4 を移動します。アクティビティ 4 の結果が「承認する」の場合、プロセスは「承認する」トランジションを移動します。アクティビティ 4 の結果が「否認する」場合、プロセスは「否認する」トランジションを移動し、「ループ・カウンタ」アクティビティに戻ります。項目が複数回拒否され、「ループ・カウンタ」アクティビティを通過した回数が「ループの限度」値を超えると、プロセスは「終了」トランジションを移動して終了します。



## 「開始」アクティビティ

「開始」アクティビティは、プロセスの開始を示すもので、処理は何も実行しません。このアクティビティは必須ではありませんが、プロセス・ダイアグラムに「開始」アクティビティを入れると、独立したノードとしてビジュアルにプロセスの開始マークを付けることができます。このアクティビティは、WF\_STANDARD.NOOP という PL/SQL プロシージャをコールします。

## 「終了」アクティビティ

「終了」アクティビティは、プロセスの終了を示すもので、処理は何も実行しません。このアクティビティを使用して、アクティビティの「結果タイプ」を指定し、完了したプロセスの結果を戻すことができます。このアクティビティは必須ではありませんが、プロセス・ダイアグラムに「終了」アクティビティを入れると、独立したノードとしてビジュアルにプロセスの終了マークを付けることができます。このアクティビティは、WF\_STANDARD.NOOP という PL/SQL プロシージャをコールします。

## 「ロール解決」アクティビティ

「ロール解決」アクティビティによって、複数のユーザーで構成されるロールから 1 人のユーザーを識別できます。プロセス・ダイアグラムで、通知アクティビティの前に「ロール解決」アクティビティを配置し、その通知アクティビティの実行者を複数のユーザーからなるロールに設定します。「ロール解決」アクティビティは、そのロールから 1 人のユーザーを選択し、そのユーザーに通知アクティビティを割り当てます。

このアクティビティは、WF\_STANDARD.ROLERESOLUTION という PL/SQL プロシージャをコールします。

### アクティビティ属性

「ロール解決」アクティビティの「方法」アクティビティ属性を使用して、ロールの解決方法を指定します。属性の値に「ロード・バランス」が設定されている場合、候補の各ユーザーについて、そのアクティビティから受信したオープン通知がいくつあるかを比較し、オープン通知が最も少ないユーザーを選択します。属性の値に「順次処理」が設定されている場合は、最後にそのアクティビティから通知を受信してからの期間が最も長いユーザーを調べ、ロールからユーザーを順に選択します。5-17 ページの「アクティビティ属性値の定義」を参照してください。

## 「通知」アクティビティ

「通知」関数アクティビティでは通知を送信でき、送信されるメッセージは直前の関数アクティビティによって実行時に動的に決定されます。「通知」アクティビティを使用するには、「通知」アクティビティによって送信される事前定義済メッセージのうちから 1 つを選択するプロセスに、前提条件となる関数アクティビティをあらかじめ用意する必要があります。

---

---

**注意：**「通知」アクティビティはアクセス・レベル0では変更できないようにロックされているため、結果タイプの値は「<なし>」から変更できません。したがって、関数アクティビティによって動的に選択されるメッセージは、結果タイプを持つことがないため、応答を禁止しない情報メッセージになります。

---

---

---

---

**注意：**「通知」アクティビティで応答が必要なメッセージを送信するには、「通知」アクティビティをコピーして独自バージョンを作成する必要があります。独自の「通知」アクティビティでは、複数のメッセージ（応答属性付き）のどれでも送信できるため、プロセスには、戻される可能性のあるすべての通知結果を用意する必要があります。

---

---

---

---

**注意：** 常に同じメッセージを送信するアクティビティを定義する場合には、この「通知」関数アクティビティを使用せずに、通知アクティビティを定義する必要があります。

---

---

この「通知」アクティビティは、WF\_STANDARD.NOTIFY という PL/SQL プロシージャをコールします。

## アクティビティ属性

「通知」アクティビティには、次の2つのアクティビティ属性があります。

- メッセージ名： 送信する事前定義済メッセージの名前。送信するメッセージを判断する前提条件となる関数アクティビティには、項目属性にこの名前が格納されている必要があります。その項目属性が「メッセージ名」アクティビティ属性で参照され、送信するメッセージ名が判断されます。
- 実行者： 通知メッセージの送信先となるロールの名前。データベースからロールをロードした場合、実行者として固定ロールを選択できます。または、実行時にロール名を戻す項目属性に実行者を設定することもできます。
- ロールの拡張： 値として、選択肢コード「Yes」または「No」を取ります。ロール内の各ユーザーに通知メッセージの個別コピーを送信する場合は、「ロールの拡張」を「Yes」に設定します。5-17 ページの「アクティビティ属性値の定義」を参照してください。

## 「はい/いいえ投票」アクティビティ

「はい/いいえ投票」アクティビティでは、ロール内のユーザーのグループに通知を送信し、ユーザーからのはい/いいえ応答を集計できます。集計結果によって、次のプロセスに進むアクティビティが決まります。

「はい/いいえ投票」アクティビティは通知アクティビティの一種です。最初にユーザー・グループに通知メッセージを送信し、次に PL/SQL の通知後関数を実行してユーザーの応答（投票）を集計します。

### アクティビティ属性

「はい/いいえ投票」アクティビティには、次の3つの属性があります。

- 「はい」の割合：「はい」の投票の割合を計算します。対象アクティビティの投票結果がすべて「はい」になると完了します。
- 「いいえ」の割合：「いいえ」の投票の割合を計算します。対象アクティビティの投票結果がすべて「いいえ」になると完了します。

---

**注意：** 最高得票の応答が結果となるように「人気投票」投票方法を使用するには、「はい」の割合および「いいえ」の割合属性の値をどちらも NULL として定義します。4-64 ページの「投票方法の例」を参照してください。

---

- 投票オプション： 次の3つの値のいずれかを選択し、投票の集計方法を指定します。
  - － 「参加者全員の投票を待つ」： ワークフロー・エンジンは、すべての投票が行われるまで待ってから、通知を受けた全ユーザーに対する割合として結果を集計します。タイムアウト条件が発生すると、タイムアウト発生前に出された投票の合計に対する割合として投票結果が計算されます。
  - － 「投票のつど集計する」： ワークフロー・エンジンは、通知を受けたすべてのユーザーに対する累積割合を、応答があるたびに集計します。タイムアウト条件が発生すると、応答は合計投票数に対する割合として集計されます。カスタム「応答」アクティビティ属性のどれかの値が空白値の場合、このオプションは意味がなくなるため注意してください。
  - － 「参加者全員の投票が必要」： すべての投票が行われた後に限り、通知を受けた全ユーザーに対する割合で応答が評価されます。タイムアウト条件が発生すると、標準的なタイムアウト・トランジションに従って進むか、使用可能なトランジションが1つも存在しない場合はエラーが発生し、投票結果は集計されません。5-17 ページの「アクティビティ属性値の定義」を参照してください。

#### 関連項目：

4-58 ページ [「投票アクティビティ」](#)

## 「マスター / ディテール連携」アクティビティ

「マスター / ディテール連携」アクティビティでは、マスター・プロセスとディテール・プロセスのフローを連携できます。たとえば、マスター・プロセスは、各ディテール・プロセスがフローの特定の点に到達したときにのみ（またはその逆）マスター・プロセスを継続するように、連携を必要とする詳細プロセスを作成できます。

Oracle Workflow のマスター・プロセスからディテール・プロセスを作成すると、実際には、一意の項目タイプと項目キーを持つ別のプロセスが作成されます。ディテール・プロセスの作成時には、`CreateProcess` API をコールしてから `StartProcess` API をコールする前に、ワークフロー・エンジンの `SetItemParent` API をコールして、2つのプロセス間にマスター / ディテール関係を定義します。8-81 ページの「[SetItemParent](#)」を参照してください。

その後、後述する 2つのアクティビティを使用して、マスター・プロセスとディテール・プロセスのフローを連携できます。一方はプロセスを休止するアクティビティ、他方は停止中のプロセスに継続するようにシグナルを送信するアクティビティです。これらのアクティビティを使用するには、一方のアクティビティをマスター・プロセスに、他方を各ディテール・プロセスに配置します。

どちらのアクティビティにも、他のプロセスの連携アクティビティの識別に使用する 2つのアクティビティ属性が含まれます。

## 「フロー待ち」アクティビティ

このアクティビティをマスター・プロセスまたはディテール・プロセスに入れて、対応する他方のディテール・プロセスまたはマスター・プロセスで指定したアクティビティが完了するまでフローを休止します。このアクティビティは、`WF_STANDARD.WAITFORFLOW` という PL/SQL プロシージャをコールします。

### アクティビティ属性

「フロー待ち」アクティビティには、次の 2つの属性があります。

- 継続フロー： 対応するマスター・プロセスまたはディテール・プロセスが完了するまで、このアクティビティが待機するかどうかを指定します。
- 継続アクティビティ： 現行のプロセスを継続する前に、対応するプロセス内で完了させる必要のあるアクティビティ・ノードのラベルを指定します。デフォルト値は `CONTINUEFLOW` です。5-17 ページの「[アクティビティ属性値の定義](#)」を参照してください。

## 「継続フロー」アクティビティ

このアクティビティを使用して、停止しているプロセスにマークを付け、対応するディテール・プロセスまたはマスター・プロセスの完了時にその位置から処理を継続します。このアクティビティは、`WF_STANDARD.CONTINUEFLOW` という PL/SQL プロシージャをコールします。

## アクティビティ属性

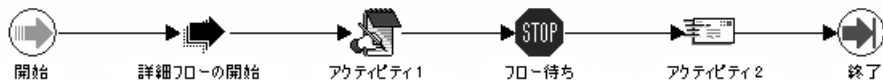
「継続フロー」アクティビティには、次の2つの属性があります。

- 待機中フロー： このアクティビティの完了を待機して停止しているプロセスが、「マスター」または「ディテール」フローのどちらであるかを指定します。
- 待機中アクティビティ： このアクティビティの完了を待機して停止しているプロセスの、アクティビティ・ノードのラベルを指定します。5-17 ページの「アクティビティ属性値の定義」を参照してください。

## 例

次の図は、これらの連携アクティビティがどのように使用されるかを示しています。マスター・プロセスの例では、マスター・プロセスが「開始」アクティビティから開始した後、「詳細フローの開始」アクティビティが複数のディテール・プロセスを開始します。その後、マスター・プロセスは「アクティビティ 1」を完了してから、「フロー待ち」アクティビティで休止します。「フロー待ち」は、すべてのディテール・プロセスが「継続フロー」アクティビティを完了するまで待ってから、マスター・プロセス「アクティビティ 2」に進み、最後に終了するように定義されています。次のディテール・プロセスの一例は、ディテール・プロセスが「開始」アクティビティから開始した後、「アクティビティ A」を完了することを示しています。「継続フロー」アクティビティに到達すると、マスター・プロセスを「フロー待ち」アクティビティから継続するように、ワークフロー・エンジンにシグナルが送信されます。その後、ディテール・プロセスは「アクティビティ B」に進み、最後に終了します。

### マスター・プロセス



### ディテール・プロセス





---

---

**注意：** 対応する 1 つ以上のディテール・プロセスに「継続フロー」アクティビティを使用せずに、マスター・プロセスに「フロー待ち」アクティビティを組み込むことができます。ワークフロー・エンジンは、「継続フロー」アクティビティを含む他のすべてのディテール・プロセスが「継続フロー」アクティビティを完了すると、即時にマスター・プロセスを継続します。

マスター・プロセスを継続する前にいつディテール・プロセスが完了してもかまわない（または、すべてのディテール・プロセスを継続する前に、いつマスター・プロセスが完了してもかまわない）場合は、単にマスター・プロセスまたはディテール・プロセスから連携する両方のアクティビティを除外します。

---

---

---

---

**注意：** プロセスに「継続フロー」アクティビティを含める場合は、「継続フロー」アクティビティのアクティビティ属性によって定義されるように、対応するマスターまたはディテール・プロセスに「フロー待ち」アクティビティも含める必要があります。

---

---

## 「割当」アクティビティ

「割当」アクティビティでは、項目属性に値を割り当てることができます。このアクティビティは、WF\_STANDARD.ASSIGN という PL/SQL プロシージャをコールします。

### アクティビティ属性

「割当」アクティビティには、「項目属性」というアクティビティ属性があります。「項目属性」を使用して、値を割り当てる項目属性を選択します。項目属性の書式タイプに応じて、「日付値」、「数値」または「テキスト値」アクティビティ属性を使用し、項目属性に割り当てる値を指定します。

## 「モニター URL」アクティビティ

「モニター URL」アクティビティでは、ワークフロー・モニターのダイアグラム・ウィンドウの URL が生成され、指定した項目属性に格納されます。このアクティビティは、WF\_STANDARD.GETURL という PL/SQL プロシージャをコールします。

### アクティビティ属性

「モニター URL」アクティビティには、次の 2 つのアクティビティ属性があります。

- 項目属性： ワークフロー・モニター・ウィンドウの URL を保存するための項目属性の名前を選択します。

- 管理モード： ワークフロー・モニター・ウィンドウで、URL をどのように表示するかを定義します。「管理モード」を「Yes」に設定すると、URL はワークフロー・モニターに ADMIN モードで表示されます。その他の設定の場合、URL はワークフロー・モニターに USER モードで表示されます。5-17 ページの「アクティビティ属性値の定義」を参照してください。

## 「イベント・プロパティの取得」アクティビティ

「イベント・プロパティの取得」アクティビティでは、ビジネス・イベント・システムからイベント・メッセージのプロパティを取り出し、そのプロパティ値を項目属性に格納します。このアクティビティは、WF\_STANDARD.GETEVENTPROPERTY という PL/SQL プロシージャをコールします。

### アクティビティ属性

「イベント・プロパティの取得」アクティビティには、次の 4 つのアクティビティ属性があります。

- イベント： イベント・タイプの項目属性を選択します。このイベント・メッセージから、プロパティを取り出します。
- プロパティ： 値を取り出すイベント・プロパティ。この属性の値は、「イベント・プロパティ」選択肢タイプの選択肢コードです。有効な値は、「優先度」、「送信日付」、「受信日付」、「関連 ID」、「イベント・パラメータ」、「イベント名」、「イベント・キー」、「送信元エージェント」、「送信元エージェント名」、「送信元エージェント・システム」、「宛先エージェント」、「宛先エージェント名」および「宛先エージェント・システム」です。8-242 ページの「イベント・メッセージ構造」を参照してください。
- イベント・パラメータ： 「プロパティ」属性で「イベント・パラメータ」プロパティを選択した場合は、値を取り出すパラメータの名前を入力します。Oracle Workflow では、この名前を使用して、イベント・メッセージのパラメータ・リスト内のパラメータを識別します。「イベント・パラメータ」以外のプロパティを選択した場合は、この属性を空白のままにします。
- 項目属性： イベント・プロパティ値を格納する項目属性。5-17 ページの「アクティビティ属性値の定義」を参照してください。

## 「イベント・プロパティの設定」アクティビティ

「イベント・プロパティの設定」アクティビティでは、ビジネス・イベント・システムのイベント・メッセージのプロパティ値を設定します。このアクティビティは、WF\_STANDARD.SETEVENTPROPERTY という PL/SQL プロシージャをコールします。

### アクティビティ属性

「イベント・プロパティの設定」アクティビティには、次の 6 つのアクティビティ属性があります。

- イベント： イベント・タイプの項目属性を選択します。このイベント・メッセージのプロパティを設定します。
- プロパティ： 値を設定するイベント・プロパティ。この属性の値は、「イベント・プロパティ」選択肢タイプの選択肢コードです。有効な値は、「優先度」、「送信日付」、「受信日付」、「関連 ID」、「イベント・パラメータ」、「イベント名」、「イベント・キー」、「送信元エージェント」、「送信元エージェント名」、「送信元エージェント・システム」、「宛先エージェント」、「宛先エージェント名」および「宛先エージェント・システム」です。8-242 ページの「イベント・メッセージ構造」を参照してください。
- イベント・パラメータ： 「プロパティ」属性の「イベント・パラメータ」プロパティを選択した場合は、値を設定するパラメータの名前を入力します。Oracle Workflow では、この名前を使用して、イベント・メッセージのパラメータ・リスト内のパラメータを識別します。「イベント・パラメータ」以外のプロパティを選択した場合は、この属性を空白のままにします。
- 日付値： イベント・プロパティに設定する日付タイプの値（「送信日付」または「受信日付」プロパティを選択した場合）
- 数値： イベント・プロパティに設定する数値タイプの値（「優先度」プロパティを選択した場合）
- テキスト値： イベント・プロパティに設定するテキスト・タイプの値（「関連 ID」、「イベント・パラメータ」、「イベント名」、「イベント・キー」、「送信元エージェント名」、「送信元エージェント・システム」、「宛先エージェント名」または「宛先エージェント・システム」プロパティを選択した場合）5-17 ページの「アクティビティ属性値の定義」を参照してください。

---

**注意：** アクティビティ属性に設定する値は、選択したイベント・プロパティのデータ型と一致していなければなりません。

---

## 「イベント・プロパティの比較」アクティビティ

「イベント・プロパティの比較」アクティビティでは、ビジネス・イベント・システムのイベント・メッセージのプロパティと、指定したテスト値を比較します。このアクティビティは、WF\_STANDARD.COMPAREEVENTPROPERTY という PL/SQL プロシージャをコールします。

### アクティビティ属性

「イベント・プロパティの比較」アクティビティには、次の 6 つのアクティビティ属性があります。

- イベント： イベント・タイプの項目属性を選択します。このイベント・メッセージのプロパティと、テスト値を比較します。

- プロパティ: テスト値と比較する値を持つイベント・プロパティ。この属性の値は、「イベント・プロパティ」選択肢タイプの選択肢コードです。有効な値は、「優先度」、「送信日付」、「受信日付」、「関連 ID」、「イベント・パラメータ」、「イベント名」、「イベント・キー」、「送信元エージェント」、「送信元エージェント名」、「送信元エージェント・システム」、「宛先エージェント」、「宛先エージェント名」および「宛先エージェント・システム」です。8-242 ページの「イベント・メッセージ構造」を参照してください。
- イベント・パラメータ: 「プロパティ」属性の「イベント・パラメータ」プロパティを選択した場合は、テスト値と比較する値を持つパラメータの名前を入力します。Oracle Workflow では、この名前を使用して、イベント・メッセージのパラメータ・リスト内のパラメータを識別します。「イベント・パラメータ」以外のプロパティを選択した場合は、この属性を空白のままにします。
- 日付値: イベント・プロパティ値と比較する日付タイプのテスト値（「送信日付」または「受信日付」プロパティを選択した場合）
- 数値: イベント・プロパティ値と比較する数値タイプのテスト値（「優先度」プロパティを選択した場合）
- テキスト値: イベント・プロパティ値と比較するテキスト・タイプのテスト値（「関連 ID」、「イベント・パラメータ」、「イベント名」、「イベント・キー」、「送信元エージェント名」、「送信元エージェント・システム」、「宛先エージェント名」または「宛先エージェント・システム」プロパティを選択した場合）

「イベント・プロパティの比較」アクティビティでは、結果コードに「比較」選択肢タイプを使用します。有効な値は、「より大きい」、「より小さい」、「等しい」、「NULL」（テスト・アクティビティ属性値が NULL の場合）です。イベント・プロパティ値とテスト値との比較方法に基づいて、ワークフロー・プロセスを進めることが可能です。5-17 ページの「アクティビティ属性値の定義」を参照してください。

---

---

**注意:** アクティビティ属性に入力するテスト値は、選択したイベント・プロパティのデータ型と一致していなければなりません。データ型が一致しないアクティビティ属性にテスト値を入力すると、「イベント・プロパティの比較」アクティビティから「NULL」結果コードが返されます。

---

---

## 「XML タグ値の取得」アクティビティ

「XML タグ値の取得」アクティビティでは、ビジネス・イベント・システムのイベント・メッセージの内容から、データを取り出します。現在、この機能を使用できるのは Oracle Workflow のスタンドアロン版のみです。このアクティビティでは、イベント・メッセージの特殊な XML タグ・セットに囲まれたデータを取り出し、指定した項目属性に格納します。「XML タグ値の取得」アクティビティは、oracle.apps.fnd.wf.XMLGetTagValue という外部 Java 関数をコールします。

---

**注意：** ワークフロー・エンジンでは、外部 Java 関数アクティビティを検出すると、エントリを「送信」キューに入れます。アクティビティの実行を続行するには、Java 関数アクティビティ・エージェントを実行して、該当する Java 関数をコールし、結果を「受信」キューに入れる必要があります。次に、バックグラウンド・エンジンを実行して「受信」キューを処理し、関数アクティビティを完了する必要があります。2-86 ページの「Java 関数アクティビティ・エージェントの設定」および 2-43 ページの「バックグラウンドのワークフロー・エンジンの設定」を参照してください。

---

### アクティビティ属性

「XML タグ値の取得」アクティビティには、次の 3 つのアクティビティ属性があります。

- イベント： イベント・タイプの項目属性を選択します。このイベント・メッセージから、データを取り出します。
- タグ： データを取り出すイベント・メッセージが囲まれているタグ・セット。タグ・セットは、XPath 表記で指定します。たとえば、発注を含む XML 文書の場合は、次の書式で発注番号タグの XML パスを指定できます。

```
/order/header/ordernumber
```

次のパスの例では、発注書の 3 行目に ITEMNO ノードがあります。

```
/order/orderlines/line[3]/itemno
```

次のパスの例では、通貨属性が「AUD」に設定されている COST ノードが、発注書の 2 行目にあります。「//」は、指定したノードがルート・ノードの下位にあることを示しています。

```
//line[2]/cost[@currency="AUD"]
```

詳細は、W3C 勧告の「XPath」を参照してください。

- 項目属性： データを格納する日付、数値またはテキスト・タイプの項目属性を選択します。項目属性と取り出すデータのデータ型は、一致していなければなりません。

**参照：** 5-17 ページの「アクティビティ属性値の定義」を参照してください。

## 「XML タグ値の比較」アクティビティ

「XML タグ値の比較」アクティビティでは、ビジネス・イベント・システムを介して受信したイベント・メッセージのデータを、テスト値と比較します。現在、この機能を使用できるのは Oracle Workflow のスタンドアロン版のみです。「XML タグ値の比較」アクティビティでは、イベント・メッセージ内で特殊な XML タグに囲まれているデータを、指定したテスト値と比較します。

「XML タグ値の比較（日付）」      このアクティビティでは、日付値が比較されます。

「XML タグ値の比較（数値）」      このアクティビティでは、数値が比較されます。

「XML タグ値の比較（テキスト）」      このアクティビティでは、テキスト値が比較されます。

すべての「XML タグ値の比較」アクティビティは、`oracle.apps.fnd.wf.XMLCompareTag` という外部 Java 関数をコールします。

---

**注意：** ワークフロー・エンジンでは、外部 Java 関数アクティビティを検出すると、エントリを「送信」キューに入れます。アクティビティの実行を続行するには、Java 関数アクティビティ・エージェントを実行して、該当する Java 関数をコールし、結果を「受信」キューに入れる必要があります。次に、バックグラウンド・エンジンを実行して「受信」キューを処理し、関数アクティビティを完了する必要があります。2-86 ページの「Java 関数アクティビティ・エージェントの設定」および 2-43 ページの「バックグラウンドのワークフロー・エンジンの設定」を参照してください。

---

### アクティビティ属性

各「XML タグ値の比較」アクティビティには、次の 3 つのアクティビティ属性があります。

- イベント： イベント・タイプの項目属性を選択します。このイベント・メッセージのデータと比較します。
- タグ： テスト値と比較するデータが含まれる、イベント・メッセージを囲むタグ・セット。タグ・セットは、XPath 表記で指定します。たとえば、発注を含む XML 文書の場合、次の書式で発注番号タグの XML パスを指定できます。

```
/order/header/ordernumber
```

次のパスの例では、発注書の 3 行目に ITEMNO ノードがあります。

```
/order/orderlines/line[3]/itemno
```

次のパスの例では、通貨属性が「AUD」に設定されている COST ノードが、発注書の 2 行目にあります。「//」は、指定したノードがルート・ノードの下位にあることを示しています。

```
//line[2]/cost[@currency="AUD"]
```

詳細は、W3C 勧告の「XML パス言語 (XPath)」を参照してください。

- 値: イベント・メッセージのデータと比較する日付、数値またはテキスト・タイプのテスト値。

「XML タグ値の比較」アクティビティでは、結果コードに「比較」選択肢タイプを使用します。有効な値は、「より大きい」、「より小さい」、「等しい」、「NULL」(テスト・アクティビティ属性値が NULL の場合) です。イベント・メッセージのデータとテスト値との比較方法に基づいて、ワークフロー・プロセスを進めることが可能です。5-17 ページの「アクティビティ属性値の定義」を参照してください。

## 「XML 変換」アクティビティ

「XML 変換」アクティビティでは、ビジネス・イベント・システムのイベント・メッセージのペイロードに対して、XML スタイル・シートを適用します。現在、この機能を使用できるのは Oracle Workflow のスタンドアロン版のみです。適用した文書は、イベント・タイプの項目属性に格納されます。このアクティビティは、`oracle.apps.fnd.wf.XSLTTransform` という外部 Java 関数をコールします。

---

**注意:** ワークフロー・エンジンでは、外部 Java 関数アクティビティを検出すると、エントリを「送信」キューに入れます。アクティビティの実行を続行するには、Java 関数アクティビティ・エージェントを実行して、該当する Java 関数をコールし、結果を「受信」キューに入れる必要があります。次に、バックグラウンド・エンジンを実行して「受信」キューを処理し、関数アクティビティを完了する必要があります。2-86 ページの「Java 関数アクティビティ・エージェントの設定」および 2-43 ページの「バックグラウンドのワークフロー・エンジンの設定」を参照してください。

---

## アクティビティ属性

「XML 変換」アクティビティには、次の 3 つのアクティビティ属性があります。

- イベント: イベント・タイプの項目属性を選択します。このイベント・メッセージを変換します。
- スタイルシート: 適用するスタイル・シートの格納場所への参照。この参照は URL として指定します。
- 新規文書: イベント・タイプの項目属性を選択します。ここに、スタイル・シートを適用して作成された新しい文書を格納します。5-17 ページの「アクティビティ属性値の定義」を参照してください。

## コンカレント・マネージャの標準アクティビティ

Oracle Applications には、Oracle Applications embedded Workflow を使用する場合に、プロセス制御に使用できるように、汎用アクティビティが用意されています。これらのアクティビティは、「コンカレント・マネージャ機能」項目タイプに関連付けられていますが、定義する任意のプロセスで使用できます。

- 「コンカレント・プログラムの実行」アクティビティ
- 「コンカレント・プログラムの発行」アクティビティ
- 「コンカレント・プログラム待ち」アクティビティ

「コンカレント・マネージャ機能」項目タイプは、Oracle Applications Workflow Server に自動的にインストールされます。この項目タイプには、\$FND\_TOP/admin/import サブディレクトリにあるファイル `fndwfaol.wft` からアクセスできます。

### 「コンカレント・プログラムの実行」アクティビティ

「コンカレント・プログラムの実行」アクティビティを使用できるのは、Oracle Applications embedded Workflow のみです。このアクティビティでは、ワークフロー・プロセスから Oracle Applications のコンカレント・プログラムを発行してその完了を待ち、完了時にアクティビティのステータスを更新し、ワークフロー・プロセスの実行をバックグラウンド・エンジンに戻します。コンカレント・プログラムは、「コンカレント・プログラムのステータス」選択枝タイプで定義されたとおり、NORMAL、ERROR、WARNING、CANCELLED または TERMINATED のうちの結果で完了してもかまいません。これらの結果はすべて、あらかじめプロセス・ダイアグラムに用意する必要があります。

---

---

**注意：**「コンカレント・プログラムの実行」アクティビティを使用するには、バックグラウンド・エンジンが実行用に設定されていることを確認する必要があります。

---

---

---

---

**注意：** 通常、Oracle Applications フォームからセッションを開始すると、プロセスの項目タイプのコンテキストが常に設定されます。ただし、通知やアクティビティのブロックなどによってセッションが中断された場合は、セレクト関数またはコールバック関数内で、`FND_GLOBAL.APPS_INITIALIZE(user_id,resp_id,resp_appl_id)` を SET\_CTX モードでコールし、コンテキストが設定されていることを確認する必要があります。7-13 ページの「項目タイプのセレクト関数またはコールバック関数の標準 API」および『Oracle Applications Developer's Guide』の「FNDSQF Routine APIs」を参照してください。

---

---



「コンカレント・プログラムの実行」アクティビティでは、標準の Oracle Application Object Library API である FND\_WF\_STANDARD.EXECUTECONCPROGRAM がコールされます。

## アクティビティ属性

「コンカレント・プログラムの実行」アクティビティには、次のアクティビティ属性があります。

- アプリケーション短縮名： コンカレント・プログラムが登録されているアプリケーションの短縮名。
- プログラム短縮名： 実行するコンカレント・プログラムの短縮名。
- 引数の数： コンカレント・プログラムの必須の引数の数。
- 項目属性名： コンカレント・プログラムの要求 ID を格納する、オプションの項目属性名。
- 引数 1、引数 2、... 引数 100： 各コンカレント・プログラムの引数の値が、コンカレント・プログラムの正しい構文に従った順序で示されます。ここでは最大 100 個の引数を使用できますが、このアクティビティの「引数の数」属性で定義したのと同じ数の引数値を指定する必要があります。5-17 ページの「アクティビティ属性値の定義」を参照してください。

## 「コンカレント・プログラムの発行」アクティビティ

「コンカレント・プログラムの発行」アクティビティを使用できるのは、Oracle Applications embedded Workflow のみです。このアクティビティでは、ワークフロー・プロセスから Oracle Applications のコンカレント・プログラムを発行しますが、このプログラムが実行または完了するまで待機するわけではありません。このアクティビティで、コンカレント要求を発行すると、ワークフロー・エンジンでは、プロセス内の次のアクティビティが続けて処理されます。

---

**注意：** 通常、Oracle Applications フォームからセッションを開始すると、プロセスの項目タイプのコンテキストが常に設定されます。ただし、通知やアクティビティのブロックなどによってセッションが中断された場合は、セレクト関数またはコールバック関数内で、  
`FND_GLOBAL.APPS_INITIALIZE(user_id, resp_id, resp_appl_id)` を SET\_CTX モードでコールし、コンテキストが設定されていることを確認する必要があります。7-13 ページの「項目タイプのセレクト関数またはコールバック関数の標準 API」を参照してください。

---

「コンカレント・プログラムの発行」アクティビティでは、標準の Oracle Application Object Library API である FND\_WF\_STANDARD.SUBMITCONCPROGRAM をコールします。

## アクティビティ属性

「コンカレント・プログラムの発行」アクティビティには、次のアクティビティ属性があります。

- アプリケーション短縮名： コンカレント・プログラムが登録されているアプリケーションの短縮名。
- プログラム短縮名： 実行するコンカレント・プログラムの短縮名。
- 引数の数： コンカレント・プログラムの必須の引数の数。
- 項目属性名： コンカレント・プログラムの要求 ID を格納する項目属性の名前。
- 引数 1、引数 2、... 引数 100: 各コンカレント・プログラムの引数の値が、コンカレント・プログラムの正しい構文に従った順序で示されます。ここでは最大 100 個の引数を使用できますが、このアクティビティの「引数の数」属性で定義したのと同じ数の引数値を指定する必要があります。5-17 ページの「アクティビティ属性値の定義」を参照してください。

## 「コンカレント・プログラム待ち」アクティビティ

「コンカレント・プログラム待ち」アクティビティを使用できるのは、Oracle Applications embedded Workflow のみです。ワークフロー・プロセスからコンカレント・プログラムを発行する場合は、「コンカレント・プログラム待ち」アクティビティを使用して、コンカレント・プログラムが完了するまで、それ以降のプロセスが実行されないようにできます。コンカレント・プログラムが完了すると、このアクティビティでは、アクティビティのステータスを更新してブロックを消去し、ワークフロー・プロセスの実行の制御をバックグラウンド・エンジンに戻します。コンカレント・プログラムは、「コンカレント・プログラムのステータス」選択肢タイプで定義されたとおり、NORMAL、ERROR、WARNING、CANCELLED または TERMINATED のうちのどれかの結果で完了してもかまいません。これらの結果はすべて、あらかじめプロセス・ダイアグラムに用意する必要があります。

---

---

**注意：**「コンカレント・プログラム待ち」アクティビティを使用するには、バックグラウンド・エンジンが実行用に設定されていることを確認する必要があります。

---

---

「コンカレント・プログラム待ち」アクティビティでは、標準の Oracle Application Object Library API である FND\_WF\_STANDARD.WAITFORCONCPROGRAM がコールされます。

## アクティビティ属性

「コンカレント・プログラム待ち」アクティビティには、「要求 ID」アクティビティ属性があります。この属性には、完了を待機しているコンカレント・プログラムの要求 ID を設定する必要があります。5-17 ページの「アクティビティ属性値の定義」を参照してください。

## デフォルト・エラー・プロセス

Oracle Workflow での設計時には、現行のプロセスでエラーが検出された場合に実行するエラー処理プロセスを指定できます。エラー処理プロセスは、プロセス、関数またはイベント・アクティビティの「詳細」プロパティ画面で指定します。エラー処理プロセスが定義されている項目タイプとエラー処理プロセスについて、内部名を指定します。

ナビゲータ・コントロールのプロパティ

アクティビティ **詳細** ロール アクセス

エラー項目タイプ① WFERROR

エラー・プロセス(P)

有効日(E) 2002/06/19

再到達時(R) リセット

バージョン(V) 0

OK キャンセル 適用(A) ヘルプ

Oracle Workflow には、「システム：エラー」と呼ばれる特別な項目タイプが用意されています。この項目タイプには、任意のプロセスで汎用的なエラー処理に使用できる 3 つのエラー・プロセスが含まれています。ただし、「システム：エラー」項目タイプのエラー・プロセスはカスタマイズできないため注意してください。これらのエラー・プロセスで使用できない機能を組み込むには、各自の項目タイプにカスタム・エラー処理プロセスを作成する必要があります。

**注意：** 特定のビジネス・ルールの非互換性によるエラーを処理する場合は、エラー・プロセスを利用するのではなく、エラーの状況をワークフロー・プロセス定義にモデル化する必要があります。たとえば、ビジネス・プロセスの前提条件が満たされないために関数アクティビティにエラーが発生する可能性がある場合は、そのような状況が発生したときに該当するルールに通知を送信して状況を修正するプロセスをモデル化して、ワークフロー・プロセスが先に進めるようにします。この状況をワークフロー・プロセスにモデル化せず、エラー・プロセスをアクティブにするエラーを利用すると、ワークフロー・プロセス全体のステータスが「エラー」になり、ワークフロー管理者がそのエラーを処理するまで停止します。

## 「システム:エラー」項目タイプと項目属性

「システム:エラー」項目タイプの詳細を参照するには、「ファイル」メニューから「オープン」を選択してからデータベースに接続し、「システム:エラー」項目タイプを選択するか、`<drive>:\<ORACLE_HOME>\wf\<Data>\<Language>` サブディレクトリにあるファイル `wferror.wft` に接続します。

「システム:エラー」項目タイプには、次の項目属性があります。

- エラー・アクティビティ ID
- エラー・アクティビティ・ラベル
- エラー割当済ユーザー
- エラー項目タイプ
- エラー項目キー
- エラー・ユーザー・キー
- エラー・メッセージ
- エラー名
- エラー通知 ID
- エラー結果コード
- エラー・スタック
- エラー・モニター URL
- タイムアウト値
- イベント名
- イベントの詳細
- イベント・メッセージ
- イベント・キー
- イベント・データの URL
- イベント・サブスクリプション
- エラー・タイプ

これらの項目属性は、「デフォルト・エラー・プロセス」、「再試行のみ」および「デフォルト・イベント・エラー・プロセス」というエラー・プロセスを構成する関数、通知およびイベント・アクティビティによって参照されます。

---

---

**注意：** 独自の項目タイプにカスタムのエラー処理プロセスを作成すると、Oracle Workflow では、エラー処理プロセスをコールするときに、前述の項目属性が自動的に設定されます。これらの項目属性は、プロセスになれば Oracle Workflow によって作成されます。ただし、メッセージなど独自のエラー処理プロセスでこれらの項目属性を参照する場合は、最初に Oracle Workflow Builder を使用して、これらの項目属性をプロセスの項目タイプの項目属性として作成する必要があります。

---

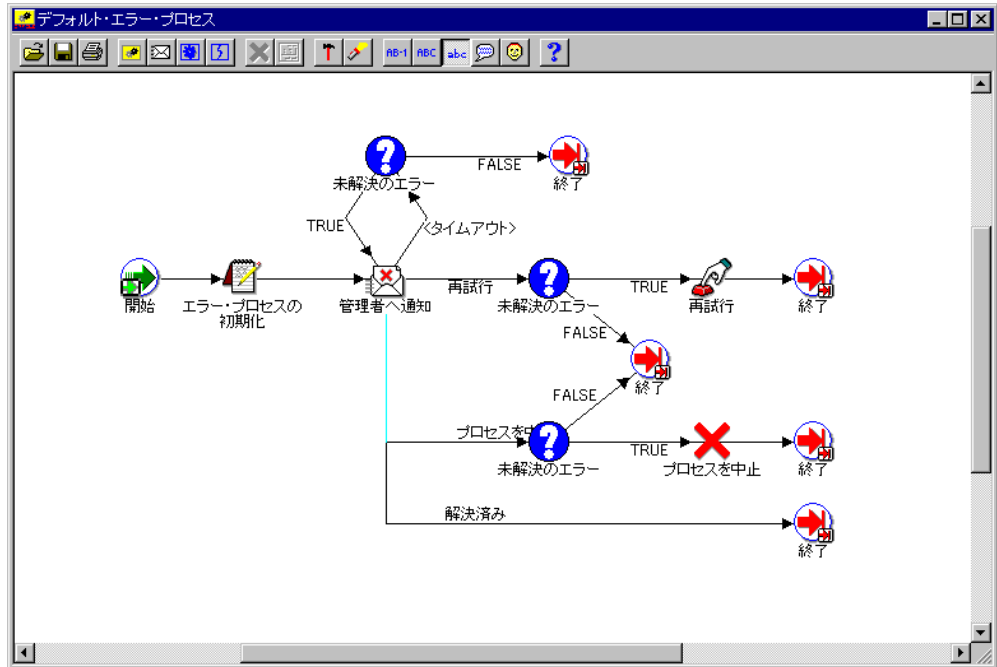
---

## デフォルト・エラー・プロセス

DEFAULT\_ERROR は、「デフォルト・エラー・プロセス」の内部名です。このエラー処理プロセスの目的は、次のとおりです。

- プロセスにエラーが発生した場合に、管理者に通知を送ります。
- 管理者にエラーに関する情報を提供します。
- 管理者によるプロセスの中止、エラーになったアクティビティの再試行、またはエラーの原因となった問題の解決を可能にします。

「デフォルト・エラー・プロセス」はカスタマイズできませんが、このプロセスには柔軟性があり、その動作はカスタマイズできます。「デフォルト・エラー・プロセス」をコールする項目タイプに、WF\_ADMINISTRATOR および ERROR\_TIMEOUT と呼ばれる 2 つの項目タイプ属性を定義することにより、それぞれに、エラー・プロセスで通知を送る宛先とエラー通知がタイムアウトになるかどうかを定義できます。



## 「エラー・プロセスの初期化」関数アクティビティ

「エラー・プロセスの初期化」アクティビティは、WF\_STANDARD.INITIALIZEERRORS という名前の PL/SQL プロシージャをコールします。このプロシージャは、エラーになったプロセスの項目タイプに、内部名 WF\_ADMINISTRATOR で項目タイプ属性の定義があるかどうかを判別します。定義があれば、後続の通知アクティビティの実行者である「管理者へ通知」を、WF\_ADMINISTRATOR に格納されているロールに設定します。定義がなければ、後続の通知アクティビティは、デフォルト実行者である「システム管理者」に設定されたままになります。

「エラー・プロセスの初期化」アクティビティでは、エラーになったプロセスの項目タイプの項目属性 WF\_ADMINISTRATOR がチェックされるため、エラー・プロセスを変更せずに、特定のプロセスでエラーが起きた場合の通知の送信先を指定できます。

たとえば、購買承認申請ワークフローがあり、このワークフローで起きた問題の解決は、システム管理者ではなく、購買管理者が行うとします。購買承認申請ワークフローを所有する項目タイプに、項目属性 WF\_ADMINISTRATOR を定義し、WF\_ADMINISTRATOR を PO\_ADMIN などの購買管理者ロールに設定できます。

## 「管理者へ通知」通知アクティビティ

「管理者へ通知」アクティビティは、「デフォルトの再試行エラー」メッセージを、実行者（システム管理者または項目タイプの WF\_ADMINISTRATOR 項目属性に格納されているロール）に送ります。このメッセージは、指定のプロセスでエラーが起きたことと、応答が必要であることを示します。応答オプションとその結果の処理は、次のとおりです。

- プロセスの異常終了：「未解決のエラー」アクティビティを実行し、エラーがまだ存在するかをチェックします。存在する場合は、「プロセスを中止」関数アクティビティをコールして、「デフォルト」エラー・プロセスは終了します。
- プロセスの再試行：「未解決のエラー」アクティビティを実行し、エラーがまだ存在するかを検証します。存在する場合は、「再試行」関数アクティビティをコールして、「デフォルト」エラー・プロセスを終了します。
- プロセスの解決： なんらかの外部的手段またはワークフロー・モニターへの埋込み URL リンクを使用して、エラーになったプロセスが直接処理されたため、「デフォルト」エラー・プロセスを終了します。

---

**注意：** 通知メッセージに埋め込まれたモニター URL により、エラーのプロセスがすべての管理者権限付きでワークフロー・モニターに表示されます。プロセスの一部の再試行、スキップまたはロールバックなどの処理を行って、エラーを解決できます。

---

「デフォルトの再試行エラー」メッセージの件名と本文は次のとおりです。

**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

```

「管理者へ通知」通知アクティビティには、動的タイムアウト値が割り当てられています。エラーの起きたプロセスの項目タイプをチェックして、内部名が `ERROR_TIMEOUT` である項目タイプ属性を検索します。`ERROR_TIMEOUT` は、数値型の属性です。ワークフロー・エンジンは、この属性値をアクティビティの開始日からの相対オフセットとして解釈し、「管理者へ通知」のタイムアウト値を分単位で決定します。`ERROR_TIMEOUT` に、`NULL` 値または値 0（ゼロ）が入っている場合や、まったく定義されていない場合は、「管理者へ通知」にタイムアウトはありません。

## 「未解決のエラー」関数アクティビティ

「管理者へ通知」アクティビティがタイムアウトになった場合や、結果として異常終了または再試行を戻した場合、ワークフロー・エンジンは「未解決のエラー」関数アクティビティを開始します。

「未解決のエラー」アクティビティは、`WF_STANDARD.CHECKERRORACTIVE` と呼ばれる PL/SQL プロシージャをコールします。「未解決のエラー」アクティビティの目的は、エラー処理を続ける前に、エラーの起きたプロセスが、まだエラー状態にあるかどうかを判別することです。エラー状態にある場合、「未解決のエラー」は `TRUE` を返し、ワークフロー・エンジンは、他の通知の送信、エラーの起きたプロセスの中止または再試行のうち、いずれか適切なトランジションを取ります。エラーの起きたプロセスがエラー状態になれば、このアクティビティは `FALSE` を返し、プロセス・ダイアグラムで示されたとおり、エラー処理プロセスは終了します。

## 「再試行」関数アクティビティ

「再試行」関数アクティビティは、`WF_STANDARD.RESETERERROR` と呼ばれる PL/SQL プロシージャを実行し、エラーになったアクティビティを消去して再実行します。このプロシージャは、`WF_ENGINE.HandleError` API をコールして、アクティビティを再実行します。

## 「プロセスを中止」関数アクティビティ

「プロセスを中止」関数アクティビティは、PL/SQL プロシージャ `WF_STANDARD.ABORTPROCESS` を実行し、続いて `WF_ENGINE.AbortProcess` API をコールして、エラーの起きたプロセスを中止します。

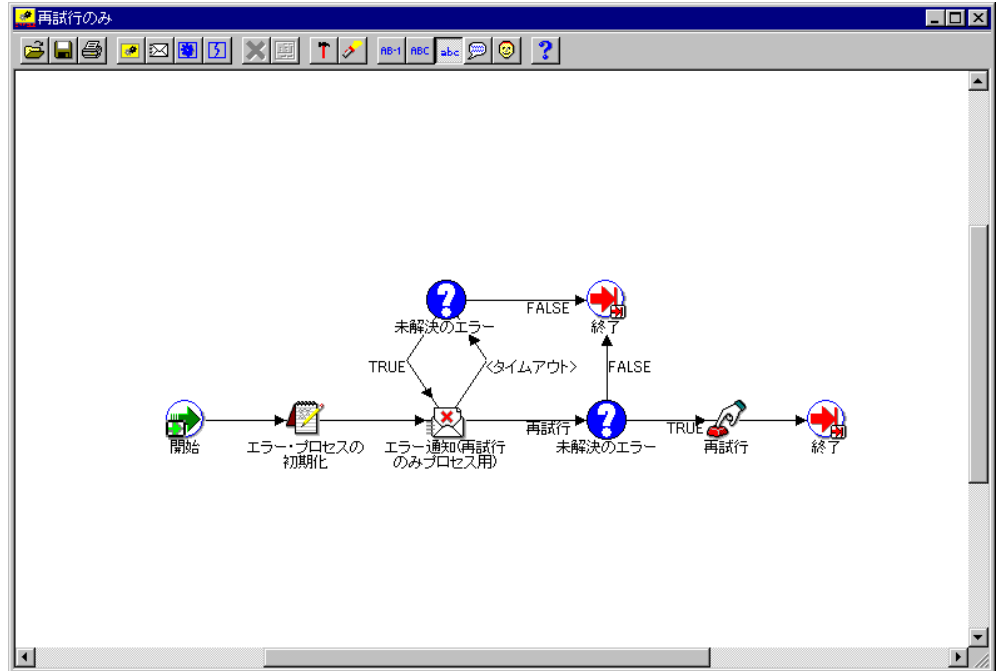
### 関連項目：

8-104 ページ [「Workflow CORE API」](#)



## 「再試行のみ」プロセス

「再試行のみ」エラー・プロセスの内部名は、RETRY\_ONLY です。このエラー処理プロセスの目的は、プロセスにエラーが発生した場合に管理者に警告することと、エラーになったプロセスを再試行するよう管理者に要求することです。



## 「エラー・プロセスの初期化」関数アクティビティ

「エラー・プロセスの初期化」アクティビティは、WF\_STANDARD.INITIALIZEERRORS という名前の PL/SQL プロシージャをコールします。このプロシージャは、エラーになったプロセスの項目タイプに、内部名 WF\_ADMINISTRATOR で項目タイプ属性の定義があるかどうかを判別します。定義があれば、後続の通知アクティビティである「エラー通知（再試行のみプロセス用）」の実行者を、WF\_ADMINISTRATOR に格納されているロールに設定します。定義がなければ、後続の通知アクティビティは、デフォルト実行者である「システム管理者」に設定されたままになります。

「エラー・プロセスの初期化」アクティビティでは、エラーになったプロセスの項目タイプの項目属性 WF\_ADMINISTRATOR がチェックされるため、エラー・プロセスを変更せずに、特定のプロセスでエラーが起きた場合の通知の送信先を指定できます。

たとえば、購買承認申請ワークフローがあり、このワークフローで起きた問題の解決は、システム管理者ではなく、購買管理者が行うとします。購買承認申請ワークフローを所有する

項目タイプに、項目属性 WF\_ADMINISTRATOR を定義し、WF\_ADMINISTRATOR を PO\_ADMIN などの購買管理者ロールに設定できます。

## 「エラー通知（再試行のみプロセス用）」通知アクティビティ

「エラー通知（再試行のみプロセス用）」アクティビティは、「再試行（オプション）」メッセージを、実行者（システム管理者または項目タイプの WF\_ADMINISTRATOR 項目属性に格納されているロール）に送ります。このメッセージは、指定のプロセスでエラーが起きたことを示し、管理者にエラーとなったアクティビティを再試行するように要求します。その後、エラー・プロセスは、「再試行」関数アクティビティに移り、「再試行のみ」エラー・プロセスを終了します。

---

---

**注意：** 通知メッセージに埋め込まれた URL リンクにより、エラーとなったプロセスがすべての管理者権限付きでワークフロー・モニターに表示されます。ワークフロー・モニターでプロセスの一部の再試行、スキップまたはロールバックなどの処理を行って、エラーを解決できます。

---

---

「再試行（オプション）」メッセージの件名と本文は次のとおりです。

**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
```

「エラー通知（再試行のみプロセス用）」通知アクティビティには、動的タイムアウト値が割り当てられています。エラーの起きたプロセスの項目タイプをチェックして、内部名が ERROR\_TIMEOUT である項目属性を検索します。ERROR\_TIMEOUT は、数値型の属性です。ワークフロー・エンジンは、この属性のタイムアウト値を、アクティビティの開始日からの分単位の相対オフセットとして解釈します。ERROR\_TIMEOUT に、NULL 値または値

0（ゼロ）が入っている場合や、まったく定義されていない場合、「エラー通知（再試行のみプロセス用）」にタイムアウトはありません。

### 「未解決のエラー」関数アクティビティ

「エラー通知（再試行のみプロセス用）」アクティビティがタイムアウトになると、ワークフロー・エンジンは「未解決のエラー」関数アクティビティを開始します。

「未解決のエラー」アクティビティは、WF\_STANDARD.CHECKERRORACTIVE と呼ばれる PL/SQL プロシージャをコールします。「未解決のエラー」アクティビティの目的は、エラー処理を続ける前に、エラーの起きたプロセスが、まだエラー状態にあるかどうかを判別することです。エラー状態にある場合、「未解決のエラー」は TRUE を戻し、ワークフロー・エンジンは「エラー通知（再試行のみプロセス用）」通知アクティビティに戻って、管理者に別の通知を送信します。エラーの起きたプロセスがエラー状態になれば、このアクティビティは FALSE を戻し、プロセス・ダイアグラムで示されたとおり、エラー処理プロセスは終了します。

### 「再試行」関数アクティビティ

「再試行」関数アクティビティは、WF\_STANDARD.RESETERROR と呼ばれる PL/SQL プロシージャを実行し、エラーになったアクティビティを消去して再実行します。このプロシージャは、WF\_ENGINE.HandleError API をコールして、アクティビティを再実行します。

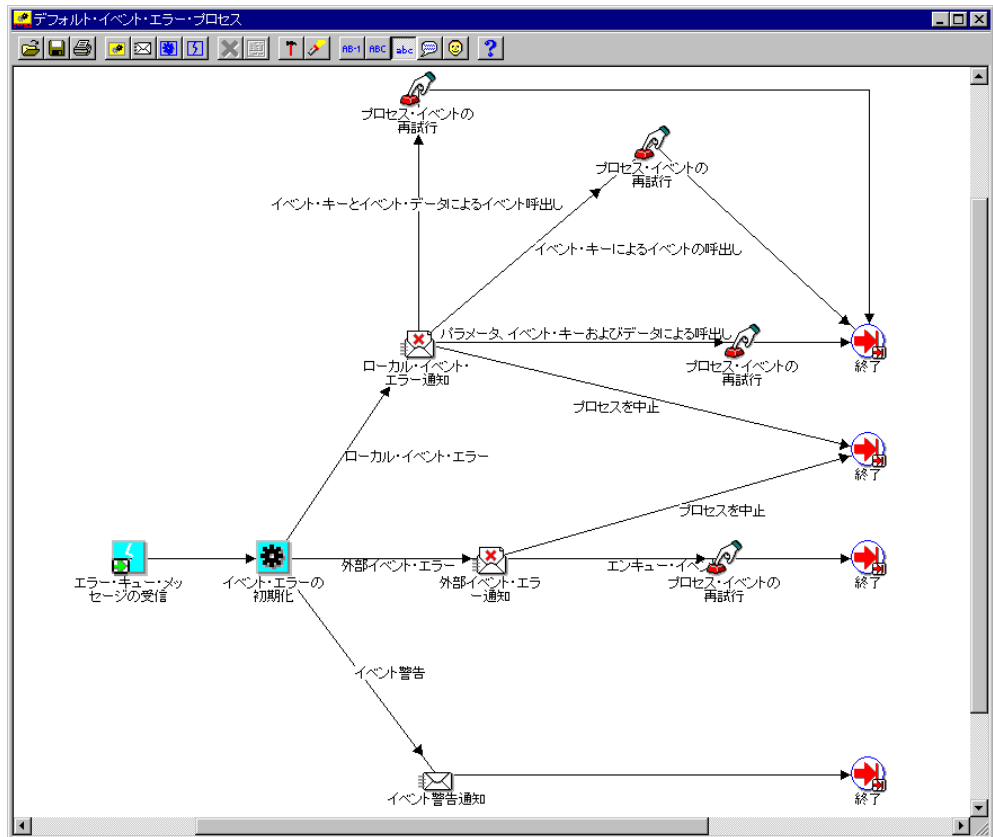
#### 関連項目：

8-104 ページ [「Workflow CORE API」](#)

## デフォルト・イベント・エラー・プロセス

DEFAULT\_EVENT\_ERROR は、ビジネス・イベント・システムの「デフォルト・イベント・エラー・プロセス」の内部名です。このエラー処理プロセスの目的は、次のとおりです。

- イベント・サブスクリプションを処理しているときにエラーまたは警告状態が発生すると、管理者に通知を送信します。
- 管理者にエラーに関する情報を提供します。
- 管理者がイベント・サブスクリプション処理を中止または再試行できるようにします。



### エラー・キュー・メッセージの受信イベント・アクティビティ

エラー・キュー・メッセージの受信イベント・アクティビティでは、サブスクリプションを処理しているときに、エラーまたは警告状態が発生したイベント・メッセージを受信しま

す。たとえば、サブスクリプションのルール関数から **ERROR** または **WARNING** ステータス・コードが返された場合、あるいは予期しないイベントが着信した場合は、イベント・マネージャによってデフォルトのサブスクリプションが実行され、イベント・メッセージが「デフォルト・イベント・エラー・プロセス」に送信されます。14-2 ページの「事前定義済ワークフロー・イベント」を参照してください。

エラー・キュー・メッセージの受信イベント・アクティビティでは、イベント名、イベント・キーおよびすべてのイベント・メッセージを項目タイプ属性に格納します。

## 「イベント・エラーの初期化」関数アクティビティ

「エラー・プロセスの初期化」アクティビティは、**WF\_STANDARD.INITIALIZEEVENTERROR** という PL/SQL プロシージャをコールします。このプロシージャによってエラー・タイプが決まります。

- イベント警告： 警告状態が発生しましたが、サブスクリプション処理は続行しました。このエラー・タイプでは、ワークフロー・エンジンによってイベント警告通知が送信されます。
- 外部イベント・エラー： エラーが発生して、外部ソースから着信したイベントのサブスクリプション処理が中止されました。このエラー・タイプでは、ワークフロー・エンジンによって「外部イベント・エラー通知」が送信されます。
- ローカル・イベント・エラー： エラーが発生して、ローカル・システムで発生したイベントのサブスクリプション処理が中止されました。このエラー・タイプでは、ワークフロー・エンジンによって「ローカル・イベント・エラー通知」が送信されます。

## 「イベント警告通知」アクティビティ

ワークフロー・エンジンでは、エラーが発生したイベントのエラー・タイプが「イベント警告」であるときに、「イベント警告通知」アクティビティを開始します。このアクティビティは、「デフォルト・イベント警告」メッセージをシステム管理者に送信して、サブスクリプションを処理しているときに警告状態が発生したことを通知します。このメッセージは FYI メッセージであるため、応答する必要はありません。

「デフォルト・イベント警告」メッセージの件名と本文は次のとおりです。

**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

## 「外部イベント・エラー通知」アクティビティ

ワークフロー・エンジンでは、エラーが発生したイベントのエラー・タイプが「外部イベント・エラー」であるときに「外部イベント・エラー通知」アクティビティを開始します。このアクティビティは、「デフォルトの外部イベント・エラー」メッセージをシステム管理者に送信します。このメッセージは、外部ソースから着信したイベントのサブスクリプションを処理しているときにエラーが発生したことで、応答が必要であることを通知します。応答オプションとその結果の処理は、次のとおりです。

- プロセスを中止： サブスクリプション処理を中止し、「デフォルト・イベント・エラー・プロセス」を終了します。たとえば、イベント・メッセージに含まれているイベント・データが破損した場合、システム管理者はそのイベント・メッセージに対するサブスクリプション処理を中止できます。
- エンキュー・イベント： 「プロセス・イベントの再試行」アクティビティを実行して、イベント・メッセージを送信したキューに戻し、「デフォルト・イベント・エラー・プロセス」を終了します。このイベント・メッセージは、-1の優先度でエンキューされるため、次にリスナーが動作するときに最初にデキューされます。

システム管理者は、エラーを訂正してからイベントを再度エンキューすることができます。たとえば、予期しないイベントを処理するサブスクリプションを作成してから、イベント・メッセージを再度エンキューすると、新しいサブスクリプションが処理されます。

「デフォルトの外部イベント・エラー」メッセージの件名と本文は次のとおりです。

**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

## 「ローカル・イベント・エラー通知」アクティビティ

ワークフロー・エンジンでは、エラーが発生したイベントのエラー・タイプが「ローカル・イベント・エラー」であるときに、「ローカル・イベント・エラー通知」アクティビティを開始します。このアクティビティは、「デフォルトのローカル・イベント・エラー」メッセージをシステム管理者に送信します。このメッセージは、ローカル・システムで発生したイベントのサブスクリプションを処理しているときにエラーが発生したことで、応答が必要であることを通知します。応答オプションとその結果の処理は、次のとおりです。

- プロセスを中止： サブスクリプション処理を中止し、「デフォルト・イベント・エラー・プロセス」を終了します。

- イベント・キーによるイベントの呼出し：「プロセス・イベントの再試行」アクティビティを実行して、イベント名とイベント・キーだけでイベントを再度呼び出し、「デフォルト・イベント・エラー・プロセス」を終了します。
- イベント・キーとイベント・データによるイベントの呼出し：「プロセス・イベントの再試行」アクティビティを実行して、イベント名、イベント・キーおよびイベント・データだけでイベントを再度呼び出し、「デフォルト・イベント・エラー・プロセス」を終了します。
- パラメータ、イベント・キーおよびイベント・データによるイベントの呼出し：「プロセス・イベントの再試行」アクティビティを実行して、イベント名、イベント・キー、イベント・データおよびパラメータでイベントを再度呼び出し、「デフォルト・イベント・エラー・プロセス」を終了します。

システム管理者は、イベントを再度呼び出すときに、イベント・マネージャに提供する情報のレベルを選択できます。たとえば、最初に渡したイベント・データにエラーがある場合は、イベント名とイベント・キーだけでイベントを再度呼び出します。このとき、イベントのジェネレート関数を使用して、イベント・データを再生成することができます。

システム管理者は、エラーを訂正してからイベントを再度呼び出すこともできます。

「デフォルトのローカル・イベント・エラー」メッセージの件名と本文は次のとおりです。

**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

## 「プロセス・イベントの再試行」関数アクティビティ

「プロセス・イベントの再試行」アクティビティは、PL/SQL プロシージャ `WF_STANDARD.RETRYRAISE` を実行します。このプロシージャでは、システム管理者が選択した通知応答に応じて、エラーが発生したイベントのエンキューまたは呼出しを再度行います。「プロセス・イベントの再試行」アクティビティを開始できる応答とそれに応じた処理は、次のとおりです。

- エンキュー・イベント：エラーが発生した外部イベント・メッセージを、送信元のキューに戻します。このイベント・メッセージは、-1 の優先度でエンキューされるため、次にリスナーが動作するときに最初にデキューされます。
- イベント・キーによるイベントの呼出し：イベント名とイベント・キーだけで、エラーが発生したローカル・イベントを再度呼び出します。

- イベント・キーとイベント・データによるイベントの呼出し： イベント名、イベント・キーおよびイベント・データだけで、エラーが発生したローカル・イベントを再度呼び出します。
- パラメータ、イベント・キーおよびイベント・データによるイベントの呼出し： イベント名、イベント・キー、イベント・データおよびパラメータで、エラーが発生したローカル・イベントを再度呼び出します。

### 関連項目：

8-104 ページ [「Workflow CORE API」](#)

13-2 ページ [「ビジネス・イベントの管理」](#)



---

# Oracle Workflow のプロシージャ および関数の定義

この章では、Oracle Workflow に対して PL/SQL および Java のプロシージャと関数を使用するときの標準 API について説明します。

## Oracle Workflow のプロシージャおよび関数の定義

Oracle Workflow では、ワークフロー・プロセスやビジネス・イベント・システムの特定の場所に、PL/SQL および Java のカスタム・プロシージャと関数を取り込むことができます。カスタム・コードを Oracle Workflow で正しく動作させるには、次の標準 API に従ってプロシージャおよび関数を作成します。

- 7-3 ページ「[関数アクティビティがコールする PL/SQL プロシージャの標準 API](#)」
- 7-8 ページ「[関数アクティビティがコールする Java プロシージャの標準 API](#)」
- 7-12 ページ「[項目タイプのセクタ関数またはコールバック関数の標準 API](#)」
- 7-15 ページ「[PL/SQL および PL/SQL CLOB 文書の標準 API](#)」
- 7-19 ページ「[イベント・データ・ジェネレート関数の標準 API](#)」
- 7-21 ページ「[キュー・ハンドラの標準 API](#)」
- 7-24 ページ「[イベント・サブスクリプションのルール関数の標準 API](#)」

## 関数アクティビティがコールする PL/SQL プロシージャの標準 API

Oracle ワークフロー・プロセスの関数アクティビティまたは通知アクティビティでコールされる PL/SQL ストアド・プロシージャはすべて、ワークフロー・エンジンでアクティビティを正しく実行できるように、次の標準 API の書式に従う必要があります。

---

---

**注意：** ワークフロー・エンジンでは、各関数アクティビティの前にセーブポイントを設定し、関数アクティビティによって生成されるエラーが検出されるようにしています。アクティビティで、処理されない例外が生成された場合、ワークフロー・エンジンはセーブポイントまでロールバックし、アクティビティのステータスを **ERROR** に設定します。このため、関数アクティビティの PL/SQL プロシージャでは、ユーザーがコミットすることはありません。コミットは、コール側のアプリケーションが行うため、ワークフロー・エンジンではコミットを発行しません。

データベース・トリガーや分散トランザクションなど、セーブポイントを使用できない環境では、ワークフロー・エンジンは自動的に「セーブポイントの使用不可」エラーを検出し、バックグラウンド・エンジンに対してアクティビティの実行を遅延させます。

---

---

この項の例には、参照しやすいように「1⇒」のように番号が付いています。この番号と矢印自体はプロシージャの一部ではありません。

```

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⇒ ワークフロー・エンジンは、関数アクティビティのストアド・プロシージャをコールするときに、プロシージャに 4 つのパラメータを渡し、プロシージャの完了時に結果を求める場合があります。次のパラメータを定義します。

<b>itemtype</b>	項目タイプの内部名。項目タイプは、Oracle Workflow Builder で定義します。
<b>itemkey</b>	項目タイプの主キーを表す文字列。主キーは、ワークフローで使用可能なアプリケーションによって生成されます。この文字列により、項目タイプの項目が一意に識別されます。
<b>actid</b>	このプロシージャのコール元となるアクティビティの ID 番号。

**funcmode**

アクティビティの実行モード。アクティビティが関数アクティビティの場合は、RUN または CANCEL モードとします。アクティビティが通知アクティビティの場合で、通知後関数を使用している場合、モードは RESPOND、FORWARD、TRANSFER、TIMEOUT または RUN のいずれかです。今後、これ以外の実行モードが追加されることもあります。

**resultout**

Oracle Workflow Builder の「アクティビティ」プロパティ画面で、アクティビティの結果タイプが指定されている場合、このパラメータはプロシージャの完了時に戻される結果を表します。戻される結果は、次のとおりです。

**COMPLETE:<result\_code>:** アクティビティは指定された結果コードを戻して完了します。結果コードは、関数アクティビティの結果タイプで指定されている結果コードの 1 つと一致する必要があります。

**WAITING:** アクティビティが保留中、つまり他のアクティビティが完了するまで待機している状態です。たとえば、標準の「And」アクティビティなどです。

**DEFERRED:<date>:** バックグラウンド・エンジンによるアクティビティの実行は、指定した日付まで延期されます。<date> は、次の書式で指定する必要があります。  
to\_char(<date\_string>, wf\_engine.date\_format)

**NOTIFIED:<notification\_id>:<assigned\_user>:** 処理の実行が必要なことが、外部エンティティに通知されます。必要であれば、この結果で通知 ID と割当て済のユーザーを戻すことができます。外部エンティティは、処理が完了した時点で CompleteActivity() をコールして、ワークフロー・エンジンに知らせる必要があることに注意してください。

**ERROR:<error\_code>:** アクティビティにエラーが発生し、指定のエラー・コードが戻されます。

2⇒ このセクションでは、このプロシージャで使用されるローカル引数を宣言します。

3⇒ プロシージャ本体は、このセクションの if 文から始まります。このセクションには、funcmode の値が RUN の場合に実行される 1 つ以上の実行文が含まれます。実行文の 1 つは、プロシージャの結果を戻すことができます。たとえば、結果として「COMPLETE:APPROVED」を戻すことがあります。

---

**注意：** 通知システムが RESPOND モードで通知後関数の実行を完了すると、ワークフロー・エンジンではその通知後関数が自動的に RUN モードで実行されます。RUN モードの実行文では、投票集計などの処理を実行し、通知アクティビティに戻す結果を決定できます。

---

4⇒ このセクションでは、アクティビティを消去します。funcmode の値が CANCEL の場合に実行される 1 つ以上の実行文を含めることができます。通常、このセクションには実行文

が含まれず、NULL 値が戻されるのみですが、必要であれば処理を取り消すことも可能です。そのアクティビティがループの一部として再実行される場合は、funcmode に CANCEL を指定できます。

ループ内の最初のアクティビティの場合は、「アクティビティ」プロパティの「詳細」ページで「ループ・リセット」フラグを常にオンにする必要があります。ワークフロー・エンジンが実行済のアクティビティに到達すると、そのアクティビティに「ループ・リセット」フラグが設定されているかどうかを検証されます。このフラグが設定されている場合は、そのループに属するアクティビティが識別され、そのアクティビティの funcmode が CANCEL に設定されます。次に、エンジンはループ内をさかのぼり、CANCEL モードの各アクティビティを実行し、そのアクティビティの以前の結果をすべて消去して、再度実行できるようにします。8-10 ページの「ループ」および 6-7 ページの「ループ・カウンタ」アクティビティ」を参照してください。

5⇒ このセクションは、通知後関数の場合にのみ必要です。このセクションを使用して、funcmode が RESPOND の場合、つまり RESPOND の処理が行われた場合に処理される実行文を指定します。たとえば、通知の応答を検証する実行文を指定できます。通知システムが RESPOND モードで通知後関数を実行した後、その通知後関数はワークフロー・エンジンにより RUN モードで再実行されます。8-13 ページの「通知後関数」を参照してください。

6⇒ このセクションは、通知後関数の場合にのみ必要です。このセクションを使用して、funcmode が FORWARD の場合、つまり通知のステータスが FORWARD に変更された場合に処理される実行文を指定します。たとえば、通知の転送先のロールを検証する実行文を指定できます。

7⇒ このセクションは、通知後関数の場合にのみ必要です。このセクションを使用して、funcmode が TRANSFER の場合、つまり通知のステータスが TRANSFER に変更された場合に処理される実行文を指定します。たとえば、通知の譲渡先のロールを検証する実行文を指定できます。

---

---

**注意：** funcmode が RESPOND、FORWARD または TRANSFER の場合は、戻り値が「ERROR%」の場合を除いて、resultout パラメータは無視されます。したがって、通知後関数を実行した後で、Respond、Forward または Transfer の処理を行わない場合は、次のどちらかの方法で対処します。

- resultout パラメータに ERROR:<errcode> を戻して、このメッセージの errcode を一般的な例外に変換します。
  - プロシージャ内で、詳細なエラー・メッセージを付加した例外を直接発生させます。8-13 ページの「通知後関数」および 8-192 ページの「通知モデル」を参照してください。
- 
- 

8⇒ このセクションは、通知後関数の場合にのみ必要です。このセクションを使用して、通知アクティビティがタイムアウト値に達したときに処理される実行文を指定します。ワークフローが正常に進行できるかどうかをテストし、問題がなければワークフローが次に進めるようにアクティビティを完了するロジックを指定できます。たとえば、投票アクティビティ

で、すべての宛先が応答する前にタイムアウト値に達した場合に、現行の応答プールに基づいて応答の解釈方法を決定し、適切な結果でアクティビティを完了するロジックを指定できます。

ワークフローが正常に進行できない場合に、`wf_engine.eng_timeout` の結果を戻すロジックも指定する必要があります。別のアクティビティへのタイムアウト・トランジションを使用し、プロセス・ダイアグラムの後続の動作をモデル化してください。ワークフロー・エンジンは、結果 `wf_engine.eng_timeout` が戻されると、タイムアウト・トランジションに従います。

9⇒ このセクションでは、`RUN`、`CANCEL`、`RESPOND`、`FORWARD`、`TRANSFER` または `TIMEOUT` 以外の実行モードを処理します。今後、これ以外の実行モードが追加されることもあります。アクティビティでは、これらのモードを実装する必要がないため、`NULL` が戻されます。

10⇒ このセクションでは、エラー・スタックにコンテキスト情報を含めてエラーの原因箇所を見つけやすいように、例外が発生した場合に `WF_CORE.CONTEXT()` をコールします。  
8-108 ページの「`CONTEXT`」を参照してください。

## 関数アクティビティがコールする Java プロシージャの標準 API

Oracle Workflow プロセスでは、カスタム Java クラスを作成し、外部 Java 関数アクティビティからコールすることができます。現在、この機能を使用できるのは Oracle Workflow のスタンドアロン版のみです。関数アクティビティからコールする Java プロシージャは、WFFunctionAPI クラスの拡張クラスとして実装します。カスタム Java クラスは、Oracle Workflow の Java 関数アクティビティ・エージェントで正しく実行できるように、標準 API の書式に従う必要があります。

---

**注意：** ワークフロー・エンジンでは、各関数アクティビティの前にセーブポイントを設定し、関数アクティビティによって生成されるエラーが検出されるようにしています。アクティビティで、処理されない例外が生成された場合、ワークフロー・エンジンはセーブポイントまでロールバックし、アクティビティのステータスを **ERROR** に設定します。このため、PL/SQL プロシージャと同様に、関数アクティビティの Java プロシージャ内でコミットしないでください。コミットは、コール側のアプリケーションが行うため、ワークフロー・エンジンではコミットを発行しません。

---

ワークフロー・エンジンおよび Workflow 通知システムのほとんどの API には、Java メソッドが対応付けられています。Java プログラムからこれらの Java メソッドをコールすれば、Oracle Workflow と通信できます。WFFunctionAPI および WFAttribute クラスには、Java プログラムからコールして項目タイプとアクティビティ属性にアクセスできるメソッドも含まれています。8-5 ページの「Oracle Workflow Java インタフェース」、8-82 ページの「Workflow Function API」および 8-90 ページの「Workflow Attribute API」を参照してください。

ワークフロー・プロセス内からカスタム Java クラスを呼び出すには、そのクラスをコールする外部 Java 関数アクティビティを作成します。4-50 ページの「関数アクティビティの作成」を参照してください。

Java 関数アクティビティは、外部プロシージャとして実装されます。ワークフロー・エンジンでは、外部 Java 関数アクティビティを検出すると、メッセージを Workflow の「送信」キューに入れます。Java 関数アクティビティ・エージェントは、このキューを監視し、関数アクティビティの「機能名」プロパティに指定されたクラスをコールします。Java プロシージャが完了すると、Java 関数アクティビティ・エージェントは結果を「受信」キューにエンキューします。2-86 ページの「Java 関数アクティビティ・エージェントの設定」を参照してください。

---

**注意：** これらの「送信」キューおよび「受信」キューは、ビジネス・イベント・システムに使用するキューとは異なります。今後のリリースでは、この関数処理はビジネス・イベント・システム内に実装され、専用のキュー・ハンドラを使用してデキュー / エンキュー操作が行われる予定です。8-162 ページの「Workflow キュー API」を参照してください。

---



Java プロシージャが完了したら、バックグラウンド・エンジンを実行して受信キューを処理し、関数アクティビティを実行する必要があります。実行しない場合は、関数アクティビティのステータスが **DEFERRED** のままになります。2-43 ページの「バックグラウンドのワークフロー・エンジンの設定」を参照してください。

カスタム・クラスを Java 関数アクティビティ・エージェントで利用するには、カスタム・クラスを格納した JAR ファイルを **CLASSPATH** に追加する必要があります。カスタム・クラス・ファイルは、Java 関数アクティビティ・エージェントが実行されるのと同じプラットフォームに常駐させる必要があります。ただし、Java 関数アクティビティ・エージェントをデータベースと同じ層に常駐させる必要はありません。

この項の例には、参照しやすいように「1⇒」のように番号が付いています。この番号と矢印自体はプロシージャの一部ではありません。

```

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.getWFEErrorStack();
           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⇒ デフォルトでは、Oracle Workflow が提供する Java クラスは `oracle.apps.fnd.wf` パッケージに入っています。このセクションはオプションです。

2⇒ 正しく操作するには、これらのパッケージを組み込む必要があります。

3⇒ カスタム Java クラスは、WFFunctionAPI クラスの拡張でなければなりません。このクラスには、関数アクティビティの操作に必要なクラス変数およびメソッドが用意されています。

通常、PL/SQL 関数アクティビティに渡されるパラメータは、カスタム・クラスのクラス変数として利用できます。それらのパラメータは、`boolean execute()` メソッドをコールする前に初期化されます。次に、`resultOut` および `errorStack` が Oracle Workflow Engine に返されます。

完了したアクティビティのステータスは、`errorStack` 変数に値がない場合、COMPLETE に設定されます。この変数に値が入っている場合、アクティビティのステータスは ERROR に設定されます。`errorStack` 変数の内容は、WFContext クラスの ErrorStack クラスを使用して設定できます。例外の取込みについては、この API のセクション 5 および 11 も参照してください。

次のクラス変数が、事前に定義されています。

<b>itemType</b>	項目タイプの内部名。項目タイプは、Oracle Workflow Builder で定義します。
<b>itemKey</b>	項目タイプの主キーを表す文字列。主キーは、ワークフローで使用する可能なアプリケーションによって生成されます。この文字列により、項目タイプの項目が一意に識別されます。
<b>ActID</b>	このプロシージャのコール元となるアクティビティの ID 番号。
<b>funcMode</b>	アクティビティの実行モード。外部 Java 関数アクティビティでは現在、「RUN」モードのみがサポートされています。
<b>resultOut</b>	Oracle Workflow Builder の「アクティビティ」プロパティ画面で、アクティビティの結果タイプが指定されている場合、このパラメータはプロシージャの完了時に戻される結果を表します。

---

**注意：** 関数アクティビティからコールされる PL/SQL プロシージャの `resultout` とは異なり、Java プロシージャの `resultOut` にはステータス・コードがありません。Java API では、結果タイプ値のみが必須です。アクティビティのステータスは、`errorStack` 変数に値が入っているかどうかに応じて、ワークフロー・エンジンが自動的に設定します。

---

4⇒ カスタム Java クラスには、`boolean execute()` メソッドを実装する必要があります。このメソッドは、Java クラスのメイン・エントリ・ポイントとなります。このメソッドは、正常に完了すると `TRUE` を返します。

5⇒ カスタム Java クラスの例外を取り込み、`ErrorStack` クラスを介してワークフロー・エンジンに返します。例外の取込みについては、この API のセクション 11 も参照してください。

6⇒ 項目属性およびアクティビティ属性を利用できるように、`WFAttribute` クラスが用意されています。

7⇒ 項目属性の値を Java クラスで利用するには、明示的または暗黙的にロードする必要があります。項目属性の値は、要求時にロードされます。明示的にロードするには、`void loadItemAttributes(WFContext)` または `void loadActivityAttributes(WFContext)` メソッドを使用します。`WFAttribute getItemAttr(String)` または `WFAttribute getActivityAttr(String)` メソッドをコールした場合は、暗黙的にロードされます。このセクションはオプションです。

8⇒ 項目属性およびアクティビティ属性の実際の値は、`WFAttribute getItemAttr(String)` および `WFAttribute getActivityAttr(String)` メソッドを介して利用します。これらの属性の値を明示的にロードしなかった場合は、この時点で自動的にロードされます。

9⇒ このセクションには、任意の実行文を記述します。通常は、必要な項目属性およびアクティビティ属性の詳細を取り出してから（セクション 8）、項目属性の値を設定する（セクション 10）までに追加します。

10⇒ `void setItemAttrValue(WFContext, WFAttribute)` メソッドを使用して項目属性の値を設定すると、ローカルの `WFAttribute` の値がデータベースに書き込まれます。`WFAttribute` クラスの値を設定するには、`WFAttribute.value(Object)` メソッドを使用する必要があります。

11⇒ カスタム Java クラス内の例外を取り込み、`ErrorStack` クラスを介してワークフロー・エンジンに返します。

外部 Java 関数アクティビティが正常に実行されなかった場合は、`FALSE` を返します。

`WFContext.wErrorStack` クラス変数のメッセージは、すべてワークフロー・エンジンに返されます。外部 Java 関数アクティビティの完了ステータスは、この時点で `ERROR` になります。

12⇒ 外部 Java 関数アクティビティが正常に実行された場合は、`TRUE` を返します。

## 項目タイプのセレクト関数またはコールバック関数の標準 API

特定の項目タイプについて、セレクト関数とコールバック関数の両方として機能する 1 つの関数を定義できます。セレクト関数は PL/SQL プロシージャの一種で、特定の項目タイプでプロセス名が指定されていないワークフローが開始されると、実行対象の特定のプロセス定義を自動的に識別します。Oracle Workflow では、コールバック関数を使用して、項目タイプのコンテキスト情報をリセットまたはテストする機能もサポートしています。次の標準 API を使用して、1 つの PL/SQL プロシージャに、セレクト関数とコールバック関数の機能の両方を定義できます。

Oracle Workflow では、次のコマンドでセレクト関数またはコールバック関数をコールできます。

- **RUN:** 次の 2 つの条件のどちらかが発生したときに開始する適切なプロセスを選択します。
  - プロセスが WF\_ENGINE.CreateProcess に明示的に渡されない場合
  - 前に WF\_ENGINE.CreateProcess がコールされず、WF\_ENGINE.CompleteActivity によってプロセスが暗黙的に開始される場合
- **SET\_CTX:** 項目タイプの関数アクティビティを実行するために必要な項目タイプと項目キーの組合せについて、コンテキスト情報を設定します。ワークフロー・エンジンでは、新規の項目タイプと項目キーの組合せが見つかるたびに、このコマンドでセレクトまたはコールバック関数がコールされ、適切なコンテキスト情報が常に設定されていることが確認されます。
- **TEST\_CTX:** 関数を実行する前に、現行の項目タイプのコンテキスト情報が正しいかどうかを判断します。たとえば、セレクトまたはコールバック関数を TEST\_CTX モードで使用すると、「通知の詳細」Web 画面で参照フォームを起動する直前に、現行のコンテキスト情報でフォームを起動できるかどうかをチェックできます。コンテキストが正しくなければ、フォームは起動されず、その旨を示すメッセージが表示されます。10-19 ページの「通知の詳細の表示」を参照してください。

セレクト関数またはコールバック関数の標準 API は、次のとおりです。この項の例には、参照しやすいように「1⇒」のように番号が付いています。この番号と矢印自体はプロシージャの一部ではありません。

```

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⇒ ワークフロー・エンジンは、セレクト関数またはコールバック関数をコールするときに、プロシージャに 4 つのパラメータを渡し、プロシージャの完了時に結果を求める場合があります。次のパラメータを定義します。

<b>itemtype</b>	項目タイプの内部名。項目タイプは、Oracle Workflow Builder で定義します。
<b>itemKey</b>	項目タイプの主キーを表す文字列。主キーは、ワークフローで使用可能なアプリケーションによって生成されます。この文字列により、項目タイプの項目が一意に識別されます。
<b>actid</b>	このプロシージャのコール元となるアクティビティの ID 番号。プロシージャが RUN コマンドを使用してコールされ、セレクト関数を実行する場合、このパラメータは常に NULL になることに注意してください。
<b>command</b>	セレクト関数またはコールバック関数の実行方法を決定するコマンド。「RUN」、「SET_CTX」または「TEST_CTX」のいずれかです。今後、他のコマンドが追加される可能性があります。

**resultout**

セレクト関数またはコールバック関数のコールに使用するコマンドによっては、結果が戻されることがあります。

関数が「RUN」でコールされる場合は、実行されるプロセスの名前を **resultout** パラメータを介して戻す必要があります。関数が「SET\_CTX」でコールされる場合、戻り値は期待されません。関数が「TEST\_CTX」でコールされる場合は、コンテキストが正しければコードで「TRUE」を戻し、正しくなければ「FALSE」を戻す必要があります。それ以外の値が戻されると、Oracle Workflow では、このコマンドがコールバックによって実装されていないものと見なされます。

2⇒ このセクションでは、このプロシージャで使用されるローカル引数を宣言します。

3⇒ プロシージャ本体は、このセクションの **if** 文から始まります。このセクションには、セレクト関数を構成する 1 つ以上の実行文が含まれます。実行文は、**command** の値が「RUN」のときに実行されます。実行文の 1 つは、実行されるプロセスを反映するプロシージャの結果を戻す必要があります。たとえば、プロセス・アクティビティ名である「REQUISITION\_APPROVAL」という結果を戻すことがあります。

4⇒ このセクションには、**command** の値が「SET\_CTX」のときに項目タイプのコンテキスト情報を設定する、1 つ以上の実行文が含まれます。ワークフロー・エンジンでは、新規の項目タイプと項目キーの組合せが見つかるたびに、このコマンドでセレクトまたはコールバック関数がコールされてから、その組合せの関数アクティビティが実行されます。このコマンドは、データベース・セッションのアクティビティを意図したとおりに実行する前に、そのセッションで項目タイプのコンテキスト情報を設定する必要がある場合に役立ちます。たとえば、複数組織データによって左右される関数アクティビティの場合は、職責と組織のコンテキストを設定する必要があります。

5⇒ このセクションには、**command** の値が「TEST\_CTX」のときに項目タイプのコンテキスト情報を検証する、1 つ以上の実行文が含まれます。ワークフロー・エンジンでは、アクティビティが実行される前に、このコマンドを使用してセレクト関数またはコールバック関数がコールされ、現行のデータベース・セッションのコンテキストを許容できるかどうかを検証されます。たとえば、このコールバック機能は、「通知の詳細」Web 画面で参照フォームが起動される直前に実行されます。このセクションのコードは、コンテキストが正しければ「TRUE」を、コンテキストが正しくなければ「FALSE」を戻す必要があります。コンテキストが正しくない場合は、例外を発生させ、WF\_CORE エラー・システムにメッセージを出して、コンテキストが無効な理由を示すことができます。発生した例外は、フォームにもエラー・メッセージとして出力されます。

6⇒ このセクションでは、RUN、SET\_CTX または TEST\_CTX 以外の実行モードを処理します。今後、他のモードも追加される可能性があります。関数では、これらのコマンドを実装する必要がないため、NULL が戻されます。

7⇒ このセクションでは、エラー・スタックにコンテキスト情報を含めてエラーの原因箇所を見つけやすいように、例外が発生した場合に WF\_CORE.CONTEXT() をコールします。

8-108 ページの「CONTEXT」を参照してください。

## PL/SQL および PL/SQL CLOB 文書の標準 API

項目タイプ、メッセージまたはアクティビティの属性として「文書」タイプを定義すると、ワークフロー・プロセスに文書を統合できます。Oracle Workflow は、「PL/SQL」文書および「PL/SQL CLOB」文書と呼ばれる文書タイプに対応しています。PL/SQL 文書はデータを文字列で表現し、PL/SQL CLOB 文書はデータをキャラクタ・ラージ・オブジェクト (CLOB) で表現します。

Oracle Workflow では、作成する文書タイプの属性によって、文書を生成する PL/SQL プロシージャの動的コールをどのように作成するかが決定されます。メッセージ本文に、PL/SQL 文書タイプのメッセージ属性を埋め込んで、通知に文書を表示できます。

PL/SQL および PL/SQL CLOB 文書を生成する PL/SQL プロシージャは、標準 API の書式に従う必要があります。

### PL/SQL 文書

PL/SQL 文書を生成する PL/SQL プロシージャには、次の標準 API を定義する必要があります。

```
procedure <procedure name> (document_id in varchar2,
                             display_type in varchar2,
                             document in out varchar2,
                             document_type in out varchar2)
```

このプロシージャの引数は、次のとおりです。

<b>document_id</b>	<p>文書を一意に識別する文字列。これは、PL/SQL 文書 (plsql:&lt;procedure&gt;/&lt;document_identifier&gt;) の「属性」プロパティ画面のデフォルト値フィールドに指定する値と同じ文字列です。&lt;procedure&gt; は、「package.procedure」の形式で PL/SQL パッケージおよびプロシージャの名前に置き換えます。&lt;document_identifier&gt; は、プロシージャに直接渡される PL/SQL 引数文字列に置き換えます。引数文字列では、文書を識別する必要があります。たとえば、plsql:po_wf.show_req/2034 となります。PL/SQL 引数文字列の値を動的に生成する場合は、PL/SQL 引数の文字列を使用するかわりに、別の項目属性を作成し、そこから対象の項目属性を「&amp;ITEM_ATTRIBUTE」として参照します。次に、別に作成した項目属性を参照するアクティビティが実行される前に、WF_ENGINE.SetItemAttribute API をコールして、PL/SQL 引数の文字列値を動的に設定します。たとえば、plsql:po_wf.show_req/&amp;POREQ_NUMBER のように指定します。</p>
--------------------	--

**display\_type**

通知表示に使用されるコンテンツ・タイプを表す 3 つの値のうちの 1 つ。要求タイプとも呼ばれます。

**text/plain:** 文書は、電子メール・メッセージから表示される通知のプレーン・テキスト表現に埋め込まれます。メール・メッセージ全体を 32KB 以内に必要があるため、電子メール・テンプレートのサイズによっては、プロシージャによって生成されるプレーン・テキスト文書の一部が切り捨てられることがあります。2-69 ページの「メッセージ・テンプレートの変更」を参照してください。

**text/html:** 文書は、「通知」 Web 画面または電子メール・メッセージの HTML 添付ファイルから表示される通知の HTML 表現に埋め込まれます。プロシージャでは、文書の HTML 表現を 32KB 以内に生成する必要があります。ただし、文書の挿入先となる HTML ページには、すでに <HTML> や <BODY> などの最上位レベルの HTML タグが含まれているため、これらのタグは除きます。最上位レベルの HTML タグが誤って入力されている場合は、文書属性がメッセージ本文内で参照されるときに、これらのタグが Oracle Workflow によって削除されます。また、通知システムがプレーン・テキストの前後を適切な HTML タグで自動的に囲んで書式を保持するため、プロシージャではプレーン・テキスト文書も生成できます。

空白: 文書は、通知の別の添付ファイルとして表示されます。任意のコンテンツ・タイプを戻すことができます。

**document**

32KB 以内の文書テキストが戻されるアウトバウンド・テキスト・バッファ。

**document\_type**

文書のコンテンツ・タイプが戻されるアウトバウンド・テキスト・バッファ。戻りタイプとも呼ばれます。タイプを指定しない場合は、「text/plain」と見なされます。

**関連項目:**

4-13 ページ [「文書属性の定義」](#)



## PL/SQL CLOB 文書

PL/SQL CLOB 文書を生成する PL/SQL プロシージャには、次の標準 API を定義する必要があります。

```
procedure <procedure name> (document_id in varchar2,
                             display_type in varchar2,
                             document in out clob,
                             document_type in out varchar2)
```

このプロシージャの引数は、次のとおりです。

### document\_id

文書を一意に識別する文字列。これは、PL/SQL CLOB 文書 (plsqlob:<procedure>/<document\_identifier>) の「属性」プロパティ画面のデフォルト値フィールドに指定する値と同じ文字列です。<procedure> は、「package.procedure」の形式で PL/SQL パッケージおよびプロシージャの名前に置き換えます。<document\_identifier> は、プロシージャに直接渡される PL/SQL 引数文字列に置き換えます。引数文字列では、文書を識別する必要があります。たとえば、plsqlob:po\_wf.show\_req\_clob/2036 となります。PL/SQL 引数文字列の値を動的に生成する場合は、PL/SQL 引数の文字列を使用するかわりに、別の項目属性を作成し、そこから対象の項目属性を「&ITEM\_ATTRIBUTE」として参照します。次に、別に作成した項目属性を参照するアクティビティが実行される前に、WF\_ENGINE.SetItemAttribute API をコールして、PL/SQL 引数の文字列値を動的に設定します。たとえば、plsqlob:po\_wf.show\_req\_clob/&POREQ\_NUMBER のように指定します。

### display\_type

通知表示に使用されるコンテンツ・タイプを表す 3 つの値のうちの 1 つ。要求タイプとも呼ばれます。

**text/plain:** 文書は、通知のプレーン・テキスト表現に埋め込まれます。

**text/html:** 文書は、「通知」Web 画面から表示される通知の HTML 表現に埋め込まれます。プロシージャでは、文書の HTML 表現を生成する必要があります。ただし、文書の挿入先となる HTML ページには、すでに <HTML> や <BODY> などの最上位レベルの HTML タグが含まれているため、これらのタグは除きます。最上位レベルの HTML タグが誤って入力されている場合は、文書属性がメッセージ本文内で参照されるときに、これらのタグが Oracle Workflow によって削除されます。また、通知システムがプレーン・テキストの前後を適切な HTML タグで自動的に囲んで書式を保持するため、プロシージャではプレーン・テキスト文書も生成できます。

**空白:** 文書は、通知の別の添付ファイルとして表示されます。任意のコンテンツ・タイプに戻すことができます。

<b>document</b>	文書テキストが戻されるアウトバウンド・テキスト・バッファ。
<b>document_type</b>	文書のコンテンツ・タイプが戻されるアウトバウンド・テキスト・バッファ。戻りタイプとも呼ばれます。タイプを指定しない場合は、「text/plain」と見なされます。

---

---

**注意：** WF\_NOTIFICATION.WriteToClob() をコールして、文字データの文字列を CLOB に追加することにより、CLOB を簡単に作成できます。8-234 ページの「WriteToClob」を参照してください。

---

---

**関連項目：**

4-13 ページ [「文書属性の定義」](#)

## イベント・データ・ジェネレート関数の標準 API

ビジネス・イベント・システムにイベントを定義するときに、ジェネレート関数をイベントに割り当てることにより、イベント名、イベント・キーおよびオプションのパラメータ・リストから完全なイベント・データを生成することができます。イベント・データには、発生するイベントおよび作成される XML 文書の詳細が追加されます。イベントを呼び出すアプリケーションがイベント・データを生成しない場合は、ジェネレート関数を指定する必要があります。

イベントがローカルで呼び出されると、イベント・マネージャはサブスクリプションを実行する前に、各サブスクリプションにイベント・データが必要かどうかをチェックします。必要なイベント・データが渡されていない場合、イベント・マネージャはそのイベントに対してジェネレート関数をコールし、イベント・データを生成します。ジェネレート関数は、キャラクタ・ラージ・オブジェクト（CLOB）形式のイベント・データを返します。

---

---

**注意：** イベント・データを必要とするイベントにジェネレート関数が定義されていない場合、イベント名およびイベント・キーを使用してデフォルトのイベント・データが作成されます。

---

---



---

---

**注意：** ジェネレート関数の負荷が大きいため、イベントの呼び出し後に制御をできるだけ早くコール側アプリケーションに戻したい場合は、完全なイベント・データを必要とするすべてのサブスクリプションを遅延させることができます。後でそれらのサブスクリプションを実行するまで、イベント・マネージャはジェネレート関数を実行しません。13-41 ページの「遅延サブスクリプション処理」を参照してください。

---

---

イベント・データを生成する PL/SQL 関数には、次の標準 API を定義する必要があります。

```
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;
```

この関数の引数は、次のとおりです。

<b>p_event_name</b>	イベントの内部名。
<b>p_event_key</b>	プログラムまたはアプリケーション内でイベントが発生したときに、生成される文字列。このイベント・キーにより、イベントの特定のインスタンスが一意に識別されます。
<b>p_parameter_list</b>	(オプション) イベントの追加したパラメータの、名前と値の組合せのリスト。

### 関連項目：

13-6 ページ [「イベントの定義」](#)

13-48 ページ [「イベント・サブスクリプションの定義」](#)

8-247 ページ [「パラメータ・リスト構造」](#)

## キュー・ハンドラの標準 API

ビジネス・イベント・システムでエージェントを定義したときは、そのエージェントにキュー・ハンドラを割り当てる必要があります。キュー・ハンドラとは、WF\_EVENT\_T データ型によって定義される標準の Workflow イベント・メッセージ形式と、エージェントに関連付けられたキューに必要なメッセージ形式との間で、変換を行うパッケージのことです。

Oracle Workflow には、WF\_EVENT\_T 形式を使用するキューに対して、WF\_EVENT\_QH (通常の処理用) および WF\_ERROR\_QH (エラー処理用) という、2つの標準キュー・ハンドラが用意されています。また、WF\_EVENT\_OMB\_QH という標準キュー・ハンドラも提供されています。Oracle8i に Oracle Message Broker を実装してシステム間でイベント・メッセージを伝播する場合には、このキュー・ハンドラを設定して使用できます。

他の形式を使用するキューに対して、独自のカスタム・キュー・ハンドラを作成することもできます。カスタム・キュー・ハンドラを作成する場合は、パッケージ内で次の標準 API を定義する必要があります。

- Enqueue()
- Dequeue()

### 関連項目：

8-248 ページ [「イベント・メッセージ構造」](#)

13-24 ページ [「エージェント」](#)

2-94 ページ [「手順 WF-15 WF\\_EVENT\\_OMB\\_QH キュー・ハンドラの設定」](#)

## エンキュー

キュー・ハンドラ・パッケージのエンキュー・プロシージャでは、送信エージェントに関連付けられたキューにイベント・メッセージをエンキューする必要があります。オーバーライド・エージェントを指定して、そこにイベント・メッセージをエンキューすることもできます。指定しない場合、イベント・メッセージはメッセージ内に指定されている送信元エージェントにエンキューされます。エンキュー・プロシージャでは、必要に応じてイベント・メッセージのヘッダー情報を変換し、キューが必要とする形式でメッセージをエンキューします。

イベント・メッセージが送信されると、WF\_EVENT.Enqueue 汎用プロシージャは指定されている送信エージェントから対応するキュー・ハンドラを判断し、そのキュー・ハンドラのエンキュー・プロシージャをコールしてメッセージをエンキューします。

PL/SQL エンキュー・プロシージャには、次の標準 API を定義する必要があります。

```
procedure enqueue (p_event in WF_EVENT_T,  
                  p_out_agent_override in WF_AGENT_T);
```

このプロシージャの引数は、次のとおりです。

<b>p_event</b>	イベント・メッセージ。
<b>p_out_agent_override</b>	送信エージェント。ここに関連付けられているキューに対して、イベント・メッセージをエンキューします。

## デキュー

キュー・ハンドラ・パッケージのデキュー・プロシージャでは、デキューするメッセージをメッセージ優先度の順に選択して、指定されている受信エージェントに関連付けられたキューからイベント・メッセージをデキューする必要があります。このプロシージャでは、必要に応じてイベント・メッセージのヘッダー情報を変換し、標準の WF\_EVENT\_T 構造でイベント・メッセージを返します。また、デキュー・プロシージャでは、メッセージがイベント・メッセージの RECEIVE\_DATE 属性にデキューされる日時を指定できます。

イベント・メッセージが着信すると、WF\_EVENT.Listen プロシージャは指定されている受信エージェントから使用するキュー・ハンドラを判断し、そのキュー・ハンドラのデキュー・プロシージャをコールしてメッセージをデキューします。

PL/SQL Dequeue プロシージャには、次の標準 API を定義する必要があります。

```
procedure dequeue (p_agent_guid in raw,  
                  p_event out WF_EVENT_T);
```

このプロシージャの引数は、次のとおりです。

<b>p_agent_guid</b>	受信エージェントのグローバル意識別子。この受信エージェントに、イベント・メッセージをデキューするキューが関連付けられています。
<b>p_event</b>	イベント・メッセージ。

## イベント・サブスクリプションのルール関数の標準 API

イベント・サブスクリプションを定義すると、イベント・メッセージに対してルール関数と呼ばれる PL/SQL 関数を実行できます。Oracle Workflow には、基本的なサブスクリプション処理を実行するための標準のデフォルト・ルール関数が用意されています。この関数は、サブスクリプションに対してルール関数が指定されていない場合に、デフォルトで実行されます。デフォルトのルール関数には次のアクションが組み込まれています。

- ワークフロー・プロセスへのイベント・メッセージの送信（サブスクリプション定義で指定されている場合）
- エージェントへのイベント・メッセージの送信（サブスクリプション定義で指定されている場合）

**参照：** 8-281 ページの「Default\_Rule」を参照してください。

Oracle Workflow には、テストおよびデバッグ、またはその他の目的で使用可能な標準のルール関数もいくつか用意されています。6-279 ページの「イベント・サブスクリプションのルール関数の API」を参照してください。

サブスクリプション処理は、カスタム・ルール関数を作成することで拡張できます。カスタム・ルール関数は、標準 API に従って定義する必要があります。

ルール関数は、イベント・メッセージに対して読取りまたは書込みを行ったり、任意のデータベース処理を実行したりします。ただし、ルール関数内ではコミットしないでください。コミットは、コール側のアプリケーションが行うため、イベント・マネージャではコミットを発行しません。また、ルール関数を使用して、セキュリティや NLS の設定などの接続コンテキストを変更しないでください。

---

**注意：** ルール関数を使用してイベント・メッセージに書き込んだ場合、そのイベントに対して実行した後続のサブスクリプションは、変更したメッセージにアクセスします。

---

特定のイベントに対してサブスクリプション処理を実行するときに、連続したステップがいくつか含まれている場合は、再利用できない複雑で特別なルール関数を作成するのではなく、再利用できる単純なルール関数を作成して複数のサブスクリプションをイベントに定義することをお勧めします。複数のサブスクリプションの実行順序を指定するには、サブスクリプションに対してフェーズ値を入力します。

デフォルトでは、イベント・マネージャではイベント・メッセージの相関 ID としてイベント・キーが使用されます（他の相関 ID が指定されていない場合）。別の相関 ID を指定する場合は、WF\_EVENT\_FUNCTIONS\_PKG.AddCorrelation を使用して相関 ID をイベント・メッセージに追加します。カスタム・ルール関数からこの関数をコールするか、WF\_EVENT\_FUNCTIONS\_PKG.AddCorrelation をルール関数として使用する別のサブスクリプションを定義してください。8-294 ページの「AddCorrelation」を参照してください。



イベント・メッセージに対してカスタム・コードを実行してから、そのメッセージをワークフロー・プロセスまたはエージェントに送信する場合は、ルール関数に送信処理を組み込むか、デフォルト・ルール関数を使用して送信処理を実行するサブスクリプションを個別に定義する必要があります。

- `WF_ENGINE.Event()` をコールすると、イベント・メッセージがワークフロー・プロセスに送信されます。
- `WF_EVENT.Send()` をコールすると、イベント・メッセージがエージェントに送信されます。
- `WF_RULE.Default_Rule()` をコールすると、ワークフロー・プロセスおよびエージェントにイベント・メッセージを送信できる、デフォルトのサブスクリプション処理が組み込まれます。

---

**注意：** イベント・マネージャにサブスクリプションを定義するときに、ルール関数を定義する以外に、ワークフロー項目タイプ、ワークフロー・プロセス名、送信エージェント、宛先エージェント、優先度および送信処理のパラメータを定義できます。すべてのルール関数でこれらの送信属性を利用できますが、デフォルト・ルール関数を使用しない場合は、カスタム・ルール関数に送信処理を明示的に組み込む必要があります（同一のサブスクリプションからイベントを送信する場合）。

---

ルール関数の標準 API は、次のとおりです。この項の例には、参照しやすいように「1⇒」のように番号が付いています。この番号と矢印自体はプロシージャの一部ではありません。

```

1⇒  function <function_name> (p_subscription_guid in raw,
                                p_event in out WF_EVENT_T) return varchar2 is
2⇒  <local declarations>
3⇒  begin
4⇒    <your executable statements>
5⇒    <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';
6⇒  return 'SUCCESS';
7⇒  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';
8⇒  end;
```

1⇒ イベント・マネージャがルール関数をコールすると、2つのパラメータがルール関数に渡されます。関数が完了すると、リターン・コードが返されます。次のパラメータを定義します。

**p\_subscription\_guid**      サブスクリプションのグローバル意識別子。  
**p\_event**                      イベント・メッセージ。

この関数は、次のステータス・コードのいずれかを返す必要があります。

- **SUCCESS:** ルール関数が正常に完了しました。
- **WARNING:** 警告状態が発生しました。ルール関数は、エラー処理用の **Workflow** コア API を使用して警告メッセージを報告し、警告情報をイベント・メッセージに設定します。イベント・マネージャは、イベント・メッセージのコピーを **WF\_ERROR** キューに格納し、サブスクリプション処理を続行します。
- **ERROR:** エラーが発生しました。ルール関数は、エラー処理用の **Workflow** コア API を使用してエラー・メッセージを報告し、エラー情報をイベント・メッセージに設定します。イベント・マネージャは、このイベントのサブスクリプション処理を停止し、イベントに対して実行済のサブスクリプションをすべてロールバックし、イベント・メッセージを **WF\_ERROR** キューに格納します。

2⇒ このセクションでは、この関数で使用するローカル引数を宣言します。

3⇒ プロシージャ本体は、このセクションから始まります。このセクションには、ルール関数を構成する 1 つ以上の実行文を記述します。

4⇒ このセクション（オプション）では、エラー・スタックにコンテキスト情報を含めてエラーの原因箇所を見つけやすいように、警告状態が発生した場合に **WF\_CORE.CONTEXT()** をコールします。また、警告情報をイベント・メッセージに設定して、ステータス・コード「**WARNING**」を返します。8-108 ページの「**CONTEXT**」を参照してください。

5⇒ このセクションでは、ルール関数の通常処理が正常に完了したときに、ステータス・コード「**SUCCESS**」を返します。

6⇒ このセクションでは、エラー・スタックにコンテキスト情報を含めてエラーの原因箇所を見つけやすいように、例外が発生した場合に **WF\_CORE.CONTEXT()** をコールします。また、エラー情報をイベント・メッセージに設定して、ステータス・コード「**ERROR**」を返します。8-108 ページの「**CONTEXT**」を参照してください。

---

**注意：** ルール関数で例外が発生した場合は、イベント・マネージャによってそのイベントのサブスクリプション処理がすべてロールバックされ、コール側のアプリケーションにエラーが発生します。この場合、イベント・メッセージは **WF\_ERROR** キューに格納されません。

---

**関連項目：**

13-48 ページ [「イベント・サブスクリプションの定義」](#)

8-248 ページ [「イベント・メッセージ構造」](#)

8-104 ページ [「Workflow CORE API」](#)

8-286 ページ [「イベント・サブスクリプションのルール関数の API」](#)

8-77 ページ [「Event」](#)

8-272 ページ [「Send」](#)

8-287 ページ [「Default\\_Rule\(\)」](#)

8-280 ページ [「SetErrorInfo」](#)



---

## Oracle Workflow の API

この章では、Oracle Workflow の API について説明します。API は、ワークフロー・エンジン、通知システム、ビジネス・イベント・システムおよびワークフロー・データへのアクセスに使用できるビュー、PL/SQL および Java の関数とプロシージャで構成されています。

## Oracle Workflow のプロシージャと関数

Oracle Workflow には、ワークフロー・プロセスを設定するためのパブリック PL/SQL および Java のプロシージャと関数のリストが用意されています。これらは、次のパッケージおよびクラスにまとめられています。

- 8-20 ページ [「Workflow Engine API」](#)
- 8-84 ページ [「Workflow Function API」](#)
- 8-92 ページ [「Workflow Attribute API」](#)
- 8-104 ページ [「Workflow CORE API」](#)
- 8-114 ページ [「Workflow PURGE API」](#)
- 8-124 ページ [「Workflow ディレクトリ・サービス API」](#)
- 8-148 ページ [「Workflow LDAP API」](#)
- 8-152 ページ [「Workflow Preference API」](#)
- 8-153 ページ [「Workflow Monitor API」](#)
- 8-161 ページ [「Oracle Workflow のビュー」](#)
- 8-167 ページ [「Workflow QUEUE API」](#)
- 8-191 ページ [「文書管理 API」](#)
- 8-202 ページ [「通知 API」](#)
- 8-267 ページ [「イベントの API」](#)
- 8-286 ページ [「イベント・サブスクリプションのルール関数の API」](#)
- 8-296 ページ [「イベント関数の API」](#)
- 8-306 ページ [「WF\\_EVENTS\\_PKG」](#)
- 8-306 ページ [「WF\\_EVENT\\_GROUPS\\_PKG」](#)
- 8-306 ページ [「WF\\_SYSTEMS\\_PKG」](#)
- 8-306 ページ [「WF\\_AGENTS\\_PKG」](#)
- 8-306 ページ [「WF\\_EVENT\\_SUBSCRIPTIONS\\_PKG」](#)

## ワークフロー・エンジンの概要

ワークフロー・エンジンは、各項目のワークフロー・プロセスで自動化されている処理をすべて管理します。ワークフロー・エンジンはサーバー側の PL/SQL に実装されており、ワークフローのプロシージャまたは関数がコールされると起動されます。このエンジンは Oracle データベース・サーバーに埋め込まれているため、ワークフローのサーバーがなんらかの原因で停止しても、Oracle データベース・サーバーを使用して、障害の発生時に実行されていたワークフローのトランザクションのリカバリと整合性を管理できます。

また、ワークフロー・エンジンをバックグラウンド・タスクとして設定し、リアル・タイムで実行するにはコストがかかりすぎるアクティビティを実行できます。

ワークフロー・エンジンでは、クライアント・アプリケーションに対する次のサービスが実行されます。

- 項目の全アクティビティのステータスを管理し、特に、前提条件アクティビティが完了するたびに、どの新規アクティビティに進む（遷移する）かを決定します。
- 関数アクティビティを自動的に実行し、通知を送信します（アクティビティは即時に実行されるか、バックグラウンド・エンジンで遅延処理されます）。
- アクティビティのステータスの履歴を保存します。
- エラー条件を検出し、エラー・プロセスを実行します。

ワークフロー項目のステータスは、その項目のプロセスの一部であるすべてのアクティビティの様々なステータスによって定義されます。エンジンは、API コールに応じてアクティビティのステータスを変更し、アクティビティを更新します。アクティビティのステータスを更新する API コールは、次のとおりです。

- 8-22 ページ [「CreateProcess」](#)
- 8-30 ページ [「StartProcess」](#)
- 8-71 ページ [「CompleteActivity」](#)
- 8-74 ページ [「CompleteActivityInternalName」](#)
- 8-76 ページ [「AssignActivity」](#)
- 8-79 ページ [「HandleError」](#)
- 8-35 ページ [「SuspendProcess」](#)
- 8-37 ページ [「ResumeProcess」](#)
- 8-39 ページ [「AbortProcess」](#)

エンジンは前のアクティビティの結果に基づいて、次のアクティビティを直接実行しようとします。アクティビティのステータスは、次のとおりです。

- Active（アクティブ）：アクティビティは実行中です。
- Complete（完了）：アクティビティは正常に終了しました。

- Waiting（待機）： アクティビティは実行待ちの状態です。
- Notified（通知済）： 通知アクティビティは配信済でオープンされています。
- Deferred（遅延）： アクティビティは延期されています。
- Error（エラー）： アクティビティはエラーを起こして完了しました。
- Suspended（中断）： アクティビティは中断されています。

---

**注意：** ワークフロー・エンジンでは、各関数アクティビティの前にセーブポイントを設定し、関数アクティビティによって生成されるエラーが検出されるようにしています。アクティビティで、処理されない例外が生成された場合、ワークフロー・エンジンはセーブポイントまでロールバックし、アクティビティのステータスを **ERROR** に設定します。このため、関数アクティビティの PL/SQL プロシージャでは、ユーザーがコミットすることはありません。コミットは、コール側のアプリケーションが行うため、ワークフロー・エンジンではコミットを発行しません。

データベース・トリガーや分散トランザクションなど、セーブポイントを使用できない環境では、ワークフロー・エンジンは自動的に「セーブポイントの使用不可」エラーを検出し、バックグラウンド・エンジンに対してアクティビティの実行を遅延させます。

---



---

**注意：** Oracle データベース・サーバー リリース 8i 以上では、自律型トランザクションを利用できます。プラグマ

AUTONOMOUS\_TRANSACTION をプロシージャに埋め込むことで、メイン・トランザクションから独立して確定およびロールバックを実行できます。Oracle データベースでは自律型トランザクションは独立したセッションとして処理されます。つまり、メイン・セッションで加えたデータベースの変更がまだ確定されていない場合は、その変更にはアクセスできません。その結果、たとえば、項目属性を設定できないなど、自律型トランザクションに含まれるワークフロー固有のデータの更新が制限されます。このデータにアクセスできないのは、項目自体がまだ確定されていないうえ、メイン・セッションでロックの競合が発生する可能性があるためです。

Oracle Workflow では、プロシージャ内で直接コールされる自律型コミットはサポートされません。コミットを実行する必要がある場合は、SQL をサブプロシージャ内に埋め込み、自律型ブロックとして宣言します。このサブプロシージャは、再実行可能でなければなりません。また、Oracle Workflow では、プロシージャ全体をロールバックし、ステータスを ERROR に設定することによって、エラーが処理されます。自律型の確定によって実行されるデータベースの更新はロールバックできないため、エラー処理に関する独自の補正ロジックを作成する必要があります。詳細は、『Oracle9i データベース概要』の「自律型トランザクション」を参照してください。

---

## Oracle Workflow Java インタフェース

Oracle Workflow Java インタフェースには、あらゆる Java プログラムを Oracle Workflow と統合するための手段が用意されています。Oracle Workflow Engine および通知の API には、パブリック・サーバーの PL/SQL パッケージおよび公開されているビューを介してアクセスできます。Oracle Workflow Java インタフェースでは、これらの API が、Oracle Workflow とやりとりするために任意の Java プログラムからコールできる Java メソッドとして公開されます。Java メソッドは、WF\_ENGINE および WF\_NOTIFICATION PL/SQL パッケージ・プロシージャおよびビューを直接参照し、JDBC を介して Oracle Workflow データベースとやりとりします。

これらのメソッドは、Java パッケージ「oracle.apps.fnd.wf.engine」内の EngineAPI クラスと NotificationAPI クラスに定義されています。ワークフロー・エンジンまたは通知の API に対応する Java メソッドがある場合は、ドキュメント内で PL/SQL 構文に続いて対象の Java メソッドの構文が表示されます。8-20 ページの「[Workflow Engine API](#)」および 8-202 ページの「[通知 API](#)」を参照してください。

また、Java 関数を外部 Java 関数アクティビティとしてワークフロー・プロセス内に組み込むことができます。現在、この機能を使用できるのは Oracle Workflow のスタンドアロン版のみです。これらのアクティビティのカスタム Java クラスは、WFFunctionAPI クラスの拡張クラスとして実装します。カスタム・クラスは、Oracle Workflow の Java 関数アクティビティ・エージェントで正しく実行できるように、標準 API の書式に従う必要があります。7-8 ページの「[関数アクティビティがコールする Java プロシージャの標準 API](#)」および 4-42 ページの「[関数アクティビティ](#)」を参照してください。

WFFunctionAPI クラスと WFAttribute クラスには、Oracle Workflow と通信するときにコールできるメソッドも含まれています。これらのクラスは、Java パッケージ「oracle.apps.fnd.wf」に定義されています。8-84 ページの「[Workflow Function API](#)」および 8-92 ページの「[Workflow Attribute API](#)」を参照してください。

Oracle Workflow と統合する Java プログラムには、次の import 文を含めて、Oracle Workflow に必要なクラスへのアクセス権を付与する必要があります。

```
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 のコンテキスト

データベースにアクセスする Oracle Workflow Java メソッドでは、WFContext オブジェクトの入力が必要になります。WFContext オブジェクトは、ユーザーによってインスタンス化されるデータベース接続情報と、WFContext クラスによってインスタンス化されるリソース・コンテキスト情報から構成されます。Java プログラムからいずれかの Workflow Engine

Java API をコールするには、最初にクラス WFDB のデータベース変数を、データベースのユーザー名、パスワードおよび別名でインスタンス化する必要があります。また、オプションで JDBC 文字列を指定することもできます。次に、データベース変数で WFContext オブジェクトをインスタンス化する必要があります。システム・プロパティ CHARSET を取得して、データベース・セッションのキャラクタ・セットを指定することができます。次のコードは、これらのオブジェクトをインスタンス化する方法の例です。

```
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
}
```

JDBC 接続をすでに確立している場合は、次の例で示すように、その接続を WFContext オブジェクトに設定します。

```
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
```

## Java プログラムのサンプル

Oracle Workflow には、ほとんどのワークフロー・エンジンおよび通知の Java API のコール方法を示す Java プログラムのサンプルが用意されています。この Java プログラム名は WFTest です。WFTest は、様々な Java API をコールして WFDEMO プロセスを起動し、属性を設定または取得し、プロセスを一時停止、再開および中止します。また、通知を送信し、通知属性を設定および取得し、通知を委任および譲渡するときも、Java API をコールします。WFTest Java プログラムを実行する前に、Oracle JDBC 実装と Oracle のサポートされるバージョンの CLASSPATH および LD\_LIBRARY\_PATH が定義されていることを確認する必要があります。たとえば、UNIX では次のコマンドを使用します。

```
setenv CLASSPATH  
<Workflow_JAR_file_directory>/wfapi.jar:${ORACLE_HOME}/jdbc/lib/classes111.zip  
  
setenv LD_LIBRARY_PATH ${ORACLE_HOME}/lib:${LD_LIBRARY_PATH}
```

---

**注意：** Oracle9i で Oracle Workflow スタンドアロン版を使用している場合、Workflow JAR ファイルは <ORACLE\_HOME>/jlib ディレクトリに格納されています。Oracle Applications embedded Workflow を使用している場合、Workflow JAR ファイルは <ORACLE\_HOME>/wf/java/oracle/apps/fnd/wf/jar/ ディレクトリに格納されています。

---

WFTest プログラムを開始するには、oracle.apps.fnd.wf.WFTest に対して Java コマンドを実行します。たとえば、UNIX の場合は、コマンドラインに次の文を指定します。

```
$java oracle.apps.fnd.wf.WFTest
```

このプログラムのソース・ファイルは、Oracle Workflow のインストールにも含まれているため、サンプル・コードを表示できます。ソース・ファイル名は WFTest.java で、<ORACLE\_HOME>/wf/java/oracle/apps/fnd/wf/ ディレクトリにあります。

## ワークフロー・エンジンのその他の機能

ワークフロー・エンジンでは、プロセスの管理の他に、次の機能もサポートしています。

- 8-9 ページ「完了処理」
- 8-9 ページ「遅延処理」
- 8-11 ページ「エラー処理」
- 8-11 ページ「ループ」
- 8-13 ページ「バージョン / 有効日付」
- 8-14 ページ「項目タイプ属性」
- 8-14 ページ「通知後関数」
- 8-15 ページ「同期プロセス、非同期プロセスおよび強制同期プロセス」
- 8-18 ページ「ビジネス・イベント」

### 完了処理

プロセス・アクティビティが完了するとエンジン処理が起動され、Workflow Engine API をコールします。次に、エンジンは完了したアクティビティに依存するすべてのアクティビティを実行（または遅延実行を示すマークを設定）しようとします。

---

**注意：** プロセス全体が完了しても、到達したのに未完了のアクティビティが含まれている場合があります。たとえば、完了したプロセスに、指定した待機時間が経過しなかったために完了していない、標準の「待機」アクティビティが含まれている場合があります。プロセス全体が完了すると、ワークフロー・エンジンは、これらの未完了アクティビティのステータスを「完了」に設定し、結果を「#FORCE」にしてマークします。この区別は、ワークフロー・モニターを介してプロセスのステータスを見直すときに重要になります。

---

### 遅延処理

エンジンには、長時間かかるタスクをリアル・タイムではなくバックグラウンド・エンジンで処理する遅延処理機能があります。アクティビティ関数の実行をバックグラウンド・エンジンでの処理に遅延させると、ワークフロー・エンジンは現在有効になっている他のアクティビティを処理できます。エンジンは、連続した処理で常に適格の作業を即時に実行するか、何も処理しないですべてのトランジションに処理の遅延マークを付けるように設定できます。

それぞれのアクティビティには、ユーザー定義の処理コストが含まれています。アクティビティで、項目属性のみを設定するとコストが低減され、多量のリソースを使用する操作を実行すると、コストが上がります。完了したアクティビティの結果によって、コストの高い関

数の実行がトリガーされる場合は、このような関数をバックグラウンド・エンジンに対して延期する場合があります。

ワークフロー・エンジンは、**Oracle Advanced Queuing** と組み合わせて、処理を延期できます。関数アクティビティのコストがメインのしきい値コストを超えている場合は、ワークフローのステータス表で、このアクティビティのステータスが「DEFERRED」に設定され、「遅延」アクティビティ用の特殊なキューに挿入されます。バックグラウンド・エンジンと呼ばれる特別なキュー・プロセッサが、「遅延」キューのアクティビティをチェックして処理します。「遅延」アクティビティの処理は、アクティビティがキューに挿入された順序で実行されます。1 つ以上のバックグラウンド・エンジンを常に有効に設定する必要があります。サイトによっては、個別のしきい値や項目タイプを指定して複数のバックグラウンド・エンジンを稼働し、どのバックグラウンド・エンジンでも、稼働しない時間が短くなるようにすることができます。

### 関連項目：

2-40 ページ [「手順 WF-8 バックグラウンドのワークフロー・エンジンの設定」](#)

C-6 ページ [「アクティビティの遅延」](#)

## エラー処理

通常、コール元ではエラーに応答する方法がわからないため、ワークフロー実行中に発生したエラーをコール元に直接戻すことはできません（実際には、コール元は担当のオペレータがいないバックグラウンド・エンジンである場合があります）。Oracle Workflow Builder を使用すると、エラー発生時に発生させる処理を定義できます。Oracle Workflow Builder を使用して、「システム:エラー」項目タイプに関連したデフォルト・エラー・プロセスを変更する方法と、独自のカスタム・エラー・プロセスを作成する方法があります。6-23 ページの「[デフォルト・エラー・プロセス](#)」を参照してください。

エラー・プロセスに、エラー・コードに基づく分岐を含めたり、通知の送信、失敗したアクティビティのリセット、再試行またはスキップなどを自動化するルールを使用したエラー処理が可能です。エラー・プロセスを定義した後で、アクティビティに関連付けることができます。これにより、そのアクティビティにエラーが発生するたびに、エラー・プロセスが実行されます 4-56 ページの「[オプションのアクティビティ詳細の定義](#)」を参照してください。

ワークフロー・エンジンでは、各関数アクティビティの前にセーブポイントを設定し、関数アクティビティによって生成されるエラーが検出されるようにしています。アクティビティで、処理されない例外が生成された場合、ワークフロー・エンジンはセーブポイントまでロールバックし、アクティビティのステータスを ERROR に設定します。

---

**注意：** このため、関数アクティビティの PL/SQL プロシージャでは、ユーザーがコミットすることはありません。コミットは、コール側のアプリケーションが行うため、ワークフロー・エンジンではコミットを発行しません。

---

その後、ワークフロー・エンジンは、エラーが発生したアクティビティを実行してエラー・プロセスに進み、対応するエラー・プロセスが見つかるまで各親プロセス・アクティビティをチェックします。

## ループ

ループは、あるアクティビティが完了した結果、すでに完了している別のアクティビティに進んだ場合に発生します。最初のアクティビティは、再実行アクティビティとして検出されるもので、ループ・ポイントまたはピボット・アクティビティとも呼ばれます。ワークフロー・エンジンには、再実行アクティビティについて次の 3 通りの処理方法が用意されています。

- 再実行アクティビティを無視し、スレッドのそれ以降の処理を中止して、実際にはこのアクティビティが 1 回しか実行されないようにします。
- 再実行する前に、最初に実行したロジックを使用して対象ループをループ・ポイントにリセットし、ループ内のアクティビティを取り消します。
- 対象ループ内で、ロジックを実行せずにループ・ポイントおよびすべてのアクティビティを再実行します。

それぞれのアクティビティには、Oracle Workflow Builder の「詳細」プロパティ画面に「再開封時」ドロップ・ダウン・フィールドがあります。「再開封時」ドロップ・ダウン・リストを使用すると、ワークフロー・エンジンがワークフロー・プロセスのアクティビティを再実行するときのように機能するかを指定できます。このフィールドには、「無視」、「リセット」または「ループ」を設定できます。

「再開封時」を「無視」に設定すると、複数の場所からトランジションが発生しても、アクティビティを 1 回しか実行しないようにする場合に役立ちます。たとえば、このモードを使用して「論理 OR」タイプのアクティビティを実装し、トランジションが何回行われても最初のトランジションの後で終了できます。

「再開封時」を「リセット」に設定すると、ループ内でアクティビティのステータスをリセットしてから、アクティビティを再実行する場合に役立ちます。リセットすると、ワークフロー・エンジンでは、次の処理が実行されます。

- ピボット・アクティビティ以降に通過した全アクティビティのリストを作成します。
- アクティビティのリストを逆にたどって、各アクティビティを取り消し、そのステータスをリセットします。

アクティビティの取消しは、アクティビティの実行に似ていますが、アクティビティは RUN モードではなく CANCEL モードで実行されます。CANCEL モードには、安全のためのロジックを定義して、それまでに RUN モードで実行されたすべての操作を取り消すことができます。

FYI 通知アクティビティを含むループのピボット・アクティビティの場合は、「再開封時」を「リセット」に設定すると、以前の通知が取り消されてから、ループが再実行され、通知アクティビティの現行の実行者に新規通知が送信されます。

アクティビティの「再開封時」を「ループ」に設定すると、ループ内でアクティビティのステータスをリセットせずに、アクティビティを再実行するのみの場合に役立ちます。「ループ」に設定すると、アクティビティに対して CANCEL モードのロジックは実行されず、アクティビティが RUN モードで再実行されます。

FYI 通知アクティビティを含むループのピボット・アクティビティの場合は、「再開封時」を「ループ」に設定すると、以前の通知はオープンのままループが再実行され、通知アクティビティの現行の実行者に新規通知が送信されます。



## バージョン/有効日付

プロセス定義のいくつかのワークフロー・オブジェクトには、バージョン番号が付けられており、常にそのオブジェクトの複数バージョンを使用できるようになっています。この種のオブジェクトは、次のとおりです。

- アクティビティ： 通知、関数およびプロセス

---

**注意：** 関数アクティビティではバージョン管理をサポートしていますが、基礎となる PL/SQL コードでは、開発者がこの機能を実装しない限りサポートしていません。PL/SQL コード内の既存のアクティビティでサポートしていない新しいアクティビティ属性を参照することや、結果の選択肢コードを戻すことはできません。

---

- アクティビティ属性
- プロセス・アクティビティ・ノード
- アクティビティ属性値
- アクティビティ・トランジション

前述のオブジェクトのいずれかを **Oracle Workflow Builder** で編集し、データベースに保存すると、**Oracle Workflow** では、バージョン番号が 1 だけ増やされ、そのオブジェクトまたは所有オブジェクトの新バージョンが自動的に作成されます。前述のオブジェクトの編集結果を既存のファイルに保存すると、元のオブジェクトが上書きされます。まだ実行中のプロセス・インスタンスがあるときに、ワークフロー・サーバー内で基礎となるワークフロー定義をアップグレードすると、そのプロセス・インスタンスは、最初に行われたときのバージョンのワークフロー・オブジェクト定義を使用して引き続き実行されます。

プロセスの実行時に、ワークフロー・エンジンで定義のどのバージョンを使用するかは、有効日付によって制御されます。プロセスの編集時には、「即時」または将来の有効日付を定義して保存できます。新しいプロセス・インスタンスが開始されると、常にこの時点で有効に設定されているバージョンが使用されます。3-13 ページの「[項目タイプのオープンと保存](#)」を参照してください。

**Oracle Workflow** では、他のワークフロー・オブジェクトのバージョンは保守されないため注意してください。次のオブジェクトに関する変更を保存すると、そのオブジェクトの既存の定義が上書きされます。

- 項目属性
- メッセージ
- 選択肢タイプ

## 項目タイプ属性

項目ごとに、設計時と実行時の両方の時点で一連の項目タイプ属性が定義されます。これらの属性は、その項目タイプに関連付けられているプロセスで使用する関数アクティビティと通知アクティビティに情報を提供します。

項目タイプ属性を実行時に定義する場合は、適切な **Workflow Engine API** を使用して、個々の属性、または同じタイプの複数の属性を含む配列を追加できます。同様に、既存の属性の値を個別に設定する方法と、同じタイプの複数の属性を含む配列の値を設定する方法があります。

一度に多数の項目タイプ属性の値を追加または設定する場合は、配列 API を使用します。これらの API では、**Oracle8i** 以降の一括バインド機能によりデータベース操作数が減少し、パフォーマンスが改善されます。**8-49** ページの「[AddItemAttributeArray](#)」および **8-56** ページの「[SetItemAttributeArray](#)」を参照してください。

---

**注意：** これらの配列 API では、タイプによりグループ化された複数の項目タイプ属性からなる配列が処理されます。**Oracle Workflow** では、配列自体からなる個別の項目タイプ属性はサポートされません。

---

## 通知後関数

通知アクティビティに対して、1 つの通知後関数を関連付けることができます。ワークフロー・エンジンでは、通知の配信後に、通知のステータス更新に応じて通知後関数が実行されます。たとえば、通知の受信者が通知を転送して譲渡したときに、通知後関数が実行されるように指定できます。通知後関数をバックエンド・ロジックで実行し、転送や譲渡が適切かどうかを検証したり、他にサポートしているロジックを実行することもできます。

通知後関数は PL/SQL プロシージャの一種で、関数アクティビティに必要な標準 API と同じものを使用して記述されています。**7-3** ページの「[関数アクティビティがコールする PL/SQL プロシージャの標準 API](#)」を参照してください。

通知後関数を指定すると、ワークフロー・エンジンでは、最初にコンテキスト情報が設定されます。この情報は、次の 2 種類のグローバル・エンジン変数を介して、関数で使用されます。

- `WF_ENGINE.context_nid = notification_ID`。
- `WF_ENGINE.context_text = new_recipient_role`（通知後関数が **FORWARD** または **TRANSFER** モードでコールされた場合）。この変数は、通知の譲渡先となる新しいロールを表します。

または

`WF_ENGINE.context_text = responder`（通知後関数が **RESPOND** モードでコールされた場合）。

---

**注意：** *responder* の値は、受信者が応答で使用する通知インタフェースによって異なります。受信者が「通知」 Web 画面を使用して応答した場合、*responder* には受信者のロール名が設定されます。受信者が電子メールで応答した場合、*responder* は「`email:responder_email_address`」に設定されます。

---

これらのグローバル・エンジン変数は、PL/SQL 関数内で参照できます。

通知のステータスが変わると、通知のコールバック関数は、通知のステータス（RESPOND、FORWARD または TRANSFER）と一致するモードで通知後関数を実行します。

通知システムが RESPOND モードで通知後関数の実行を終了すると、ワークフロー・エンジンは自動的にその通知後関数を RUN モードで再実行します。このモードでは、通知後関数で投票集計のような追加の処理を実行できます。

通知アクティビティがタイムアウトになると、ワークフロー・エンジンはそのアクティビティの通知後関数を TIMEOUT モードで実行します。投票アクティビティの場合、TIMEOUT モードのロジックでは、タイムアウトになるまでに受け取った投票の集計方法を識別する必要があります。

通知後関数が完了すると、2 つのグローバル・エンジン変数は無効になります。

最終手順として、通知後関数が TRANSFER モードで実行され、通知アクティビティの「拡張ロール」がオフになっている場合は、その通知に割り当てられているユーザーが指定の新規ロール名に設定されます。

---

**注意：** 通知後関数で `ERROR:<errcode>` が戻されるか、例外が発生すると、ワークフロー・エンジンは応答、転送または譲渡の処理を中止します。たとえば、通知後関数を FORWARD モードで実行した場合に、転送先のロールが不正なために例外が発生すると、エラーが表示され、転送処理を実行できなくなります。通知の受信者は、もう一度なんらかの処理を行うように要求されます。

---

#### 関連項目：

8-197 ページ「[通知モデル](#)」

## 同期プロセス、非同期プロセスおよび強制同期プロセス

ワークフロー・プロセスには、同期プロセスまたは非同期のプロセスがあります。同期プロセスは、開始から終了まで中断しないで実行されるプロセスです。バックグラウンド・エンジンにすぐに渡される関数アクティビティなど、プロセスに含まれるアクティビティが遅延アクティビティでない場合、ワークフロー・エンジンはプロセスを同期的に実行します。ワークフロー・エンジンは、プロセスが完了するまで、ワークフローを開始したコール元の

アプリケーションに制御を返しませんが、同期プロセスの場合、項目属性に書き込まれたプロセスの結果またはデータベースに直接書き込まれたプロセスの結果をすぐにチェックできます。ただし、ユーザーはプロセスが完了するまで待機する必要があります。

非同期プロセスには、フローを中断するアクティビティが含まれます。このため、ワークフロー・エンジンは非同期プロセスの実行を待機します。非同期プロセスを要求するアクティビティには、遅延アクティビティ、応答を必要とする通知、ブロック・アクティビティ、待機アクティビティなどがあります。ワークフロー・エンジンがこれらのアクティビティを検出すると、無期限に待機するのでなく、監査表を適切に設定したうえでコール元のアプリケーションに制御を返します。ワークフロー・プロセスは、再開されるまで未完了の状態になります。このプロセスは、通知システム（ユーザーが通知に応答したときなど）、バックグラウンド・エンジン（遅延アクティビティが実行されたときなど）、またはビジネス・イベント・システム（イベント・メッセージが受信キューからデキューされ、ワークフロー・プロセスに送信されたときなど）によって再開されます。非同期プロセスの場合、ユーザーはプロセスが完了するまで待機する必要がなく、アプリケーションの使用を続行できます。ただし、プロセスの結果は、プロセスが完了するまで確認できません。

ワークフロー・エンジンでは、通常の同期プロセスと非同期プロセスの他に、強制同期プロセスと呼ばれる、特別な種類の同期プロセスも使用できます。強制同期プロセスは、開始から終了まで1つのSQLセッションで完了し、データベース表に対して挿入または更新を行います。そのため、強制同期プロセスの実行速度は、通常の同期プロセスよりもかなり速くなります。プロセスの結果は、完了時にすぐに確認できます。ただし、監査証跡は記録されません。

強制同期プロセスは、監査証跡を記録する必要がなく、特定の結果をすばやく出力したいときに使用します。たとえば、**Oracle Applications** で、複数の製品で勘定科目生成処理ワークフローを使用して、様々な表から導出された一連の連結セグメントによって、有効なフレックスフィールド・コードを生成する場合です。この場合は、勘定科目生成処理ワークフローが強制同期プロセスとなり、ここで計算が実行され、完了したフレックスフィールド・コードが、コールしたアプリケーションに即時に戻されます。

強制同期プロセスを作成するには、プロセスの `itemkey` を `#SYNCH` または `wf_engine.eng_synch` に設定する必要があります。これにより、必要な `WF_ENGINE` API をコールすると、`#SYNCH` 定数が戻されます。強制同期プロセスでは、データベースに対して書込みを行わないため、`#SYNCH` などの重複した `itemkey` を使用しても問題ありません。ただし、プロセス定義は次の制限事項に従う必要があります。

- 通知アクティビティは使用できません。
- 制限付きのブロック・タイプ・アクティビティは使用できません。1つのプロセスが `WF_ENGINE.CompleteActivity` をコールしてブロックおよび再開できるのは、ブロックされるアクティビティと再開されるアクティビティが次の場合のみです。
  - アクティビティが同じデータベース・セッションで発生する場合
  - アクティビティ内で `Oracle Workflow` をコールしない場合
  - アクティビティ内でコミットしない場合

- エラー・プロセスは、プロセスに割り当てることも、そのプロセスのアクティビティに割り当てることもできません。
- それぞれの関数アクティビティは、「再開封時」が「ループ」に設定されている場合と同様に動作し、実際の「再開封時」の設定に関係なく、**CANCEL** 以外のモードで実行されます。このプロセスでは、ループを使用することもできます。
- 「マスター / ディテール連携」アクティビティは使用できません。
- 各アクティビティからのトランジションの結果が異なるため、このプロセスでは並列フローは使用できません。これは、並列フローが生成されるため、任意トランジションも使用できないことを表します。
- 次の標準アクティビティは、いずれも使用できません。
  - And
  - ブロック（前述の制限付きのブロックの箇所に説明されているように、制約があります。）
  - スレッド遅延
  - 待機
  - 継続フロー / フロー待ち
  - ロール解決
  - 投票
  - 実行時間の比較
  - 通知
- バックグラウンド・エンジンは使用できません。つまり、アクティビティで遅延は発生しません。
- データは、Oracle Workflow の表に書き込まれないため、次のようになります。
  - このプロセスはワークフロー・モニターで参照できません。
  - このプロセスには監査機能を使用できません。
- 次の WF\_ENGINE API コールのみが作成されます。いずれの場合も、これらの API で提供される itemkey は、#SYNCH または wf\_engine.eng\_synch として指定する必要があります。
  - WF\_ENGINE.CreateProcess
  - WF\_ENGINE.StartProcess
  - WF\_ENGINE.GetItemAttribute
  - WF\_ENGINE.SetItemAttribute

- WF\_ENGINE.GetActivityAttribute
- WF\_ENGINE.CompleteActivity (ブロック・タイプ・アクティビティの制限使用の場合)
- WF\_ENGINE API は、現行の同期項目に対する項目以外の項目には使用できません。

---

**注意：** 強制同期プロセスでエラーが見つかった場合は、同期モードで一意の項目キーを使用してプロセスに戻り、ワークフロー・モニターまたはスクリプト `wfstat.sql` を使用してエラー・スタックをチェックする必要があります。同期プロセスが正常終了しているのに、強制同期プロセスでエラーが見つかった場合は、前述の制約に違反していることが原因と考えられます。16-16 ページの「[wfstat.sql](#)」を参照してください。

---

**関連項目：**

8-15 ページ「[同期プロセス、非同期プロセスおよび強制同期プロセス](#)」

## ビジネス・イベント

ビジネス・イベント・システムからのイベントは、ワークフロー・プロセス内でイベント・アクティビティとして表されます。イベント・アクティビティは、ビジネス・イベントを発生、送信または受信できます。

「呼出し」 イベント・アクティビティは、イベントをイベント・マネージャに呼び出して、そのイベントへのサブスクリプションをトリガーします。ワークフロー・エンジンは、WF\_EVENT.Raise API をコールしてイベントを呼び出します。8-268 ページの「[Raise](#)」を参照してください。

「送信」 イベント・アクティビティは、イベント・マネージャにイベントを呼び出さずに、ビジネス・イベント・システムのエージェントにイベントを直接送信します。ワークフロー・エンジンは、WF\_EVENT.Send API をコールしてイベントを送信します。8-272 ページの「[Send](#)」を参照してください。

「受信」 イベント・アクティビティは、イベント・マネージャからイベントを受信してワークフロー・プロセスに格納し、そのアクティビティからスレッドの実行を続行します。ワークフロー・エンジンは、イベントを受信すると、イベント・メッセージの関連 ID を使用して、イベントと対応するプロセスとを照合します。照合が完了すると、イベントを待機している既存のプロセス・インスタンスのアクティビティに渡します。また、ワークフロー・エンジンは、受信したイベントを「開始」アクティビティとしてマークされた「受信」イベント・アクティビティに渡し、新しいワークフロー・プロセスを起動することもできます。ワークフロー・エンジンは、受信したイベントをワークフロー・プロセスに渡すときに、WF\_ENGINE.Event API を使用します。8-77 ページの「[Event](#)」を参照してください。

---

**注意：**「受信」イベント・アクティビティが受信したイベントが別のワークフロー・プロセスの「呼出し」イベント・アクティビティによって発生した場合は、そのプロセスの項目タイプと項目キーがイベント・メッセージ内のパラメータ・リストに追加されます。ワークフロー・エンジンは、イベントを受信するプロセスの親として、指定されたプロセスを自動的に設定し、既存の親の設定を上書きします。8-81 ページの「[SetItemParent](#)」を参照してください。

---

**関連項目：**

13-2 ページ「[ビジネス・イベントの管理](#)」

4-42 ページ「[イベント・アクティビティ](#)」

## Workflow Engine API

Workflow Engine API は、ランタイム・フェーズでアプリケーション・プログラムやワークフロー関数によってコールされ、エンジンとの通信や各アクティビティのステータス変更を行います。これらの API は、WF\_ENGINE という PL/SQL パッケージで定義されています。

Workflow Engine API の多くには、対応する Java メソッドも定義されています。これらの Java メソッドは、任意の Java プログラムからコールして、Oracle Workflow に取り込むことができます。次のリストは、Workflow Engine API をどのように使用できるか（つまり PL/SQL 関数 / プロシージャ、Java メソッドのいずれか、またはその両方として使用できるか）を表しています。

---

---

**注意：** Java では大文字 / 小文字が区別されます。Java のネーミング規則に従って、すべての Java メソッド名の先頭文字は小文字となります。

---

---

- 8-22 ページ [「CreateProcess」](#) : PL/SQL および Java
- 8-25 ページ [「SetItemUserKey」](#) : PL/SQL
- 8-26 ページ [「GetItemUserKey」](#) : PL/SQL
- 8-27 ページ [「GetActivityLabel」](#) : PL/SQL
- 8-28 ページ [「SetItemOwner」](#) : PL/SQL および Java
- 8-30 ページ [「StartProcess」](#) : PL/SQL および Java
- 8-33 ページ [「LaunchProcess」](#) : PL/SQL および Java
- 8-35 ページ [「SuspendProcess」](#) : PL/SQL および Java
- 8-37 ページ [「ResumeProcess」](#) : PL/SQL および Java
- 8-39 ページ [「AbortProcess」](#) : PL/SQL および Java
- 8-41 ページ [「CreateForkProcess」](#) : PL/SQL
- 8-43 ページ [「StartForkProcess」](#) : PL/SQL
- 8-44 ページ [「Background」](#) : PL/SQL
- 8-46 ページ [「AddItemAttribute」](#) : PL/SQL および Java
- 8-49 ページ [「AddItemAttributeArray」](#) : PL/SQL
- 8-51 ページ [「SetItemAttribute」](#) : PL/SQL および Java
- 8-54 ページ [「SetItemAttrDocument」](#) : PL/SQL および Java
- 8-56 ページ [「SetItemAttributeArray」](#) : PL/SQL
- 8-58 ページ [「getItemTypes」](#) : Java



- 8-59 ページ [「GetItemAttribute」](#) : PL/SQL
- 8-61 ページ [「GetItemAttrDocument」](#) : PL/SQL
- 8-62 ページ [「GetItemAttrClob」](#) : PL/SQL
- 8-63 ページ [「getItemAttributes」](#) : Java
- 8-64 ページ [「GetItemAttrInfo」](#) : PL/SQL
- 8-64 ページ [「GetItemAttrInfo」](#) : PL/SQL
- 8-66 ページ [「GetActivityAttribute」](#) : PL/SQL
- 8-68 ページ [「GetActivityAttrClob」](#) : PL/SQL
- 8-69 ページ [「BeginActivity」](#) : PL/SQL
- 8-71 ページ [「CompleteActivity」](#) : PL/SQL
- 8-74 ページ [「CompleteActivityInternalName」](#) : PL/SQL
- 8-76 ページ [「AssignActivity」](#) : PL/SQL
- 8-77 ページ [「Event」](#) : PL/SQL
- 8-79 ページ [「HandleError」](#) : PL/SQL
- 8-81 ページ [「SetItemParent」](#) : PL/SQL
- 8-82 ページ [「ItemStatus」](#) : PL/SQL および Java
- 8-83 ページ [「getProcessStatus」](#) : Java

**関連項目：**

7-3 ページ [「関数アクティビティがコールする PL/SQL プロシージャの標準 API」](#)

## CreateProcess

### PL/SQL 構文

```
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 構文

```
public static boolean createProcess
(
  WfContext wCtx,
  String itemType,
  String itemKey,
  String process)
```

### 説明

アプリケーション項目に対して新しいランタイム・プロセスを作成します。

たとえば、「購買申請」項目タイプには、最上位レベルのプロセスとして「購買承認申請」プロセスがあります。特定の購買申請が作成されると、アプリケーションは **CreateProcess** をコールして、定義されたプロセスの開始に必要な情報を設定します。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>itemtype</b>	有効な項目タイプ。項目タイプは、Workflow Builder で定義します。
<b>itemkey</b>	通常アプリケーション・オブジェクトの主キーから導出される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーによって新しいプロセスが識別されるため、そのプロセスの後続のすべての API コールにそれらを渡す必要があります。

---

---

**注意：** itemkey として #SYNCH を渡して、強制同期プロセスを作成できません。8-15 ページの「[同期プロセス、非同期プロセスおよび強制同期プロセス](#)」を参照してください。

---

---

<b>process</b>	その項目の特定のプロセスを選択するためのオプションの引数。プロセスの内部名を指定します。プロセスが NULL の場合は、項目タイプのセレクト関数によって、実行する最上位レベルのプロセスが判別されます。セレクト関数およびこの引数を指定しない場合は、エラーが発生します。
<b>user_key</b>	指定した項目タイプと項目キーで識別される項目に割り当てるキー。わかりやすい名前を付けます。この引数はオプションです。
<b>owner_role</b>	有効なロール。項目の所有者として設定します。この引数はオプションです。

---

**注意：** データベース・トリガーから `CreateProcess()` および `StartProcess()` をコールしてワークフロー・プロセスを開始できますが、状況によってはこの方法を使用しないでください。たとえば、データベース・エンティティにヘッダー、行および詳細があり、そのエンティティのヘッダー・レベルで `AFTER INSERT` トリガーからワークフロー・プロセスを実行すると、ワークフロー・プロセスが失敗することがあります。これは、プロセス内の後続のアクティビティによって、まだ値が挿入されていないエンティティの行レベルや詳細レベルの情報が要求されることがあるためです。

---



---

**注意：** ワークフロー・エンジンは、エラー発生時に直前のアクティビティにロールバックできるように、各アクティビティを実行する前に常にセーブポイントを発行します。データベース・トリガーや分散トランザクションなど、セーブポイントを使用できない環境では、ワークフロー・エンジンは自動的に「セーブポイントの使用不可」エラーを検出し、アクティビティの実行を遅延させます。ワークフロー・プロセスをデータベース・トリガーから開始する必要がある場合は、初期の「開始」アクティビティを即時にバックグラウンド・エンジンに遅延して、データベース・トリガーから実行されないようにする必要があります。

---

## 例

次のコードは、Java プログラムで `createProcess()` をコールする方法の例です。このコード例は、`WFTTest.java` プログラムからの引用です。

```
// 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 構文

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

### 説明

最初は項目タイプと項目キーで識別されるプロセス内の項目について、ユーザーが親しみやすい識別子を設定できます。ユーザー・キーは、ワークフロー・モニターや、**Oracle Workflow** の他のユーザー・インタフェースのコンポーネント内で項目を検索するための、ユーザーが親しみやすい識別子として使用します。

### 引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	通常、アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセス内の項目が識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。
<b>userkey</b>	指定した項目タイプと項目キーで識別される項目に割り当てるユーザー・キー。

## GetItemUserKey

### PL/SQL 構文

```
function GetItemUserKey  
    (itemtype in varchar2,  
     itemkey in varchar2)  
    return varchar2;
```

### 説明

項目タイプと項目キーで識別されるプロセス内の項目に割り当てられている、ユーザーが親しみやすいキーを戻します。ユーザー・キーは、ワークフロー・モニターや、Oracle Workflow の他のユーザー・インタフェースのコンポーネント内で項目を検索するための、ユーザーが親しみやすい識別子です。

### 引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	通常、アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセス内の項目が識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。

## GetActivityLabel

### PL/SQL 構文

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

### 説明

内部アクティビティ・インスタンス ID を指定すると、アクティビティのインスタンス・ラベルを戻します。戻されるラベルの書式は、次のとおりです。これは、**CompleteActivity** や **HandleError** など、アクティビティ・ラベルを引数として受け入れる他の Workflow Engine API に渡すのに適した書式です。

`<process_name>:<instance_label>`

### 引数（入力）

**actid**                      アクティビティ・インスタンス ID。

## SetItemOwner

### PL/SQL 構文

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

### Java 構文

```
public static boolean setItemOwner
  (WFContext wCtx,
   String itemType,
   String itemKey,
   String owner)
```

### 説明

既存項目の所有者を設定します。所有者には、有効なロールを指定する必要があります。通常、トランザクションを開始するロールは、プロセス所有者として割り当てられます。これにより、ロールの関係者は、ワークフロー・モニターでプロセス・インスタンスのステータスを検索および参照できます。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>itemtype</b>	有効な項目タイプ。項目タイプは、Workflow Builder で定義します。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから導出される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーによって新しいプロセスが識別されるため、そのプロセスの後続のすべての API コールにそれらを渡す必要があります。
<b>owner</b>	有効なロール。

### 例

次のコードは、Java プログラムで `setItemOwner()` をコールする方法の例です。このコード例は、`WFTest.java` プログラムからの引用です。

```
// set item owner
```



```
if (WFEEngineAPI.setItemOwner(ctx, itemType, iKey, owner))
    System.out.println("Set Item Owner:"+owner);
else
{
    System.out.println("Cannot set owner.");
    WFEEngineAPI.showError(ctx);
}
```

## StartProcess

### PL/SQL 構文

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

### Java 構文

```
public static boolean startProcess
  (WFContext wCtx,
   String itemType,
   String itemKey)
```

### 説明

指定したプロセスの実行を開始します。エンジンは **START** とマークされたアクティビティを検出し、それを実行します。最初に **CreateProcess()** がコールされ、**StartProcess()** をコールする前に **itemtype** と **itemkey** を定義します。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから導出される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。

---

---

**注意：** トリガーから **CreateProcess()** および **StartProcess()** をコールしてワークフロー・プロセスを開始できますが、状況によってはこの方法を使用しないでください。たとえば、データベース・エンティティにヘッダー、行および詳細があり、そのエンティティのヘッダー・レベルで **AFTER INSERT** トリガーからワークフロー・プロセスを実行すると、ワークフロー・プロセスが失敗することがあります。これは、プロセス内の後続のアクティビティによって、まだ値が挿入されていないエンティティの行レベルや詳細レベルの情報が要求されることがあるためです。

---

---

---

**注意：** ワークフロー・エンジンは、エラー発生時に直前のアクティビティにロールバックできるように、各アクティビティを実行する前に常にセーブポイントを発行します。この機能では、セーブポイントとロールバックをデータベース・トリガーに含めることはできないため、ワークフロー・プロセスをデータベース・トリガーから実行することは避ける必要があります。

ワークフロー・プロセスをデータベース・トリガーから開始する必要がある場合は、初期の開始アクティビティを即時にバックグラウンド・エンジンに遅延して、データベース・トリガーから実行されないようにする必要があります。そのためには、次の方法があります。

---

- 「プロセス開始」アクティビティのコストを、ワークフロー・エンジンのしきい値より大きい値に設定します（デフォルト値は 0.5）。
- または
- プロセスを開始する前に、ワークフロー・エンジンのしきい値を 0 より小さい値に設定します。

```
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
```

（この方法では、最初の方法と同じ効果が得られますが、初期の「開始」アクティビティはコストが変化しても常に遅延されるため、より安全です。）

---

**注意：** itemkey として #SYNCH を渡して、強制同期プロセスを作成できます。8-15 ページの「同期プロセス、非同期プロセスおよび強制同期プロセス」を参照してください。

---

## 例

次のコードは、Java プログラムで `startProcess()` をコールする方法の例です。このコード例は、`WFTest.java` プログラムからの引用です。

```
// start a process
if (WFEEngineAPI.startProcess(ctx, iType, iKey))
    System.out.println("Process Started successfully");
```

```
    else
    {
        System.out.println("launch failed");
        WFEEngineAPI.showError(ctx);
    }
```

## LaunchProcess

### PL/SQL 構文

```
procedure LaunchProcess
  (itemtype in varchar2,
   itemkey in varchar2,
   process in varchar2 default '',
   userkey in varchar2 default '',
   owner in varchar2 default '');
```

### Java 構文

```
public static boolean LaunchProcess
  (WFContext wCtx,
   String itemType,
   String itemKey,
   String process,
   String userKey,
   String owner)
```

### 説明

新しいランタイム・プロセスを作成し、指定したプロセスを開始して実行します。このプロセスージャは、CreateProcess と StartProcess を組み合わせたラッパーです。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから導出される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーによって新しいプロセスが識別されるため、そのプロセスの後続のすべての API コールにそれらを渡す必要があります。

---

---

**注意：** itemkey として #SYNCH を渡して、強制同期プロセスを作成できます。8-15 ページの「[同期プロセス、非同期プロセスおよび強制同期プロセス](#)」を参照してください。

---

---

<b>process</b>	その項目タイプの特定のプロセスを選択するためのオプションの引数。プロセスの内部名を指定します。プロセスが <b>NULL</b> の場合は、項目タイプのセレクト関数によって、実行する最上位レベルのプロセスが判別されます。この引数のデフォルト値は <b>NULL</b> です。
<b>userkey</b>	指定した項目タイプと項目キーで識別される項目に割り当てるユーザー・キー。ユーザー・キーが <b>NULL</b> の場合は、項目インスタンスにユーザー・キーは割り当てられません。
<b>owner</b>	有効なロール。項目の所有者として指定します。所有者が <b>NULL</b> の場合は、プロセスに所有者は割り当てられず、プロセスを監視できるのはワークフローの管理者ロールのみとなります。

---

**注意：** データベース・トリガーから `CreateProcess()` および `StartProcess()` をコールしてワークフロー・プロセスを開始できますが、状況によってはこの方法を使用しないでください。たとえば、データベース・エンティティにヘッダー、行および詳細があり、そのエンティティのヘッダー・レベルで **AFTER INSERT** トリガーからワークフロー・プロセスを実行すると、ワークフロー・プロセスが失敗することがあります。これは、プロセス内の後続のアクティビティによって、まだ値が挿入されていないエンティティの行レベルや詳細レベルの情報が要求されることがあるためです。

---

---

**注意：** ワークフロー・エンジンは、エラー発生時に直前のアクティビティにロールバックできるように、各アクティビティを実行する前に常にセーブポイントを発行します。データベース・トリガーや分散トランザクションなど、セーブポイントを使用できない環境では、ワークフロー・エンジンは自動的に「セーブポイントの使用不可」エラーを検出し、アクティビティの実行を遅延させます。ワークフロー・プロセスをデータベース・トリガーから開始する必要がある場合は、初期の「開始」アクティビティを即時にバックグラウンド・エンジンに遅延して、データベース・トリガーから実行されないようにする必要があります。

---

## SuspendProcess

### PL/SQL 構文

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

### Java 構文

```
public static boolean suspendProcess
(WFContext wCtx,
String itemType,
String itemKey,
String process)
```

### 説明

新規のトランジションが発生しないように、プロセスの実行を保留します。  
CompleteActivity() をコールすれば処理中の通知を完了できますが、ワークフローでは次のアクティビティには進みません。ResumeProcess() をコールして、保留中のプロセスを再開します。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。

**process**

その項目タイプの特定のサブプロセスを選択するためのオプションの引数。プロセス・アクティビティのラベル名を指定します。プロセス・アクティビティのラベル名で特定のサブプロセスを識別できない場合は、ラベル名の前に親プロセスの内部名を追加できます。たとえば、  
<parent\_process\_internal\_name>:<label\_name> のように指定します。この引数が NULL の場合は、項目の最上位レベルのプロセスが中止されます。この引数のデフォルト値は NULL です。

**例**

次のコードは、Java プログラムで `suspendProcess()` をコールする方法の例です。このコード例は、`WFTest.java` プログラムからの引用です。

```
// 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 構文

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

### Java 構文

```
public static boolean resumeProcess
  (WFContext wCtx,
   String itemType,
   String itemKey,
   String process)
```

### 説明

保留中のプロセスを通常の実行状態に戻します。これにより、プロセスの保留中にトランジションが発生したアクティビティが実行されます。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。
<b>process</b>	その項目タイプの特定のサブプロセスを選択するためのオプションの引数。プロセス・アクティビティのラベル名を指定します。プロセス・アクティビティのラベル名で特定のサブプロセスを識別できない場合は、ラベル名の前に親プロセスの内部名を追加できます。たとえば、 <code>&lt;parent_process_internal_name&gt;:&lt;label_name&gt;</code> のように指定します。この引数が NULL の場合は、項目の最上位レベルのプロセスが中止されます。この引数のデフォルト値は NULL です。

## 例

次のコードは、Java プログラムで `resumeProcess()` をコールする方法の例です。このコード例は、`WFTest.java` プログラムからの引用です。

```
// 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 構文

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

### Java 構文

```
public static boolean abortProcess
(
  WfContext wCtx,
  String itemType,
  String itemKey,
  String process,
  String result)
```

### 説明

プロセスの実行を中止し、未完了の通知を取り消します。プロセスのステータスは、`result` 引数で指定した結果によって **COMPLETE** とみなされます。また、未完了の通知やサブプロセスは、`result` 引数に関係なく、結果によって強制的に **COMPLETE** のステータスが設定されます。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。

**process**

その項目タイプの特定のサブプロセスを選択するためのオプションの引数。プロセス・アクティビティのラベル名を指定します。プロセス・アクティビティのラベル名で特定のサブプロセスを識別できない場合は、ラベル名の前に親プロセスの内部名を追加できます。たとえば、

`<parent_process_internal_name>:<label_name>` のように指定します。この引数が `NULL` の場合は、項目の最上位レベルのプロセスが中止されます。この引数のデフォルト値は `NULL` です。

**result**

中止されたプロセスに割り当てるステータス。**result** は、プロセスの「結果タイプ」で定義された値の 1 つか、次の標準エンジンの値の 1 つである必要があります。

`eng_exception`

`eng_timeout`

`eng_force`

`eng_mail`

`eng_null`

この引数のデフォルトは「`eng_force`」です。

**例**

次のコードは、Java プログラムで `abortProcess()` をコールする方法の例です。このコード例は、`WFTest.java` プログラムからの引用です。

```
// abort process, should see status COMPLETE with result
// code force
System.out.println("Abort Process ..." + iType + "/" + iKey);
if (!WFEEngineAPI.abortProcess(ctx, iType, iKey, pr, null))
{
    System.out.println("Seemed to have problem aborting...");
    WFEEngineAPI.showError(ctx);
}
```

## CreateForkProcess

### PL/SQL 構文

```
procedure CreateForkProcess
  (copy_itemtype in varchar2,
   copy_itemkey in varchar2,
   new_itemkey in varchar2,
   same_version in boolean default TRUE);
```

### 説明

オリジナルのコピーである新規プロセスを作成し、ランタイム・プロセスをフォークします。CreateForkProcess() をコールした後で SetItemOwner()、SetItemUserKey() または SetItemAttribute などの API をコールし、新規プロセスに必要な項目プロパティをリセットしたり項目属性を変更できます。その後、StartForkProcess() をコールして新規プロセスを開始する必要があります。

プロセスの途中で項目固有の属性を変更する必要がある場合は、CreateForkProcess() を使用します。たとえば、在庫不足のために受注を満たせない場合は、CreateForkProcess() を使用して、受注残数量に対する新規トランザクションをフォークできます。承認通知はすべてコピーされるため注意してください。結果は、このトランザクションに関して 2 つの項目が作成された場合と同じになります。

### 引数（入力）

<b>copy_itemtype</b>	コピーする元のプロセスの有効な項目タイプ。新規プロセスは、同じ項目タイプとなります。
<b>copy_itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。copy_itemtype と copy_itemkey により、コピー元となるプロセスが識別されます。
<b>new_itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと新規項目キーにより、新規プロセスが識別されます。
<b>same_version</b>	TRUE または FALSE を指定して、新規ランタイム・プロセスでオリジナルと同じバージョンを使用するか、最新バージョンを使用するかを指定します。TRUE を指定すると、CreateForkProcess() では、オリジナル・プロセスの項目属性とステータスが新規プロセスにコピーされます。FALSE を指定すると、CreateForkProcess() では、オリジナル・プロセスの項目属性は新規プロセスにコピーされますが、ステータスはコピーされません。デフォルト値は TRUE です。

---

---

**注意：** CreateForkProcess() と StartForkProcess() は、プロセスの並列分岐からはコールしないでください。この 2 つの API では、有効でない独自分岐に並列の分岐はコピーされません。

---

---

---

---

**注意：** 項目をフォークすると、Oracle Workflow により自動的に新規項目の項目属性 #FORKED\_FROM が作成され、この属性が元の項目の項目キーに設定されます。この属性は、フォークされた項目の実行情報を提供します。

---

---

## StartForkProcess

### PL/SQL 構文

```
procedure StartForkProcess
  (itemtype in varchar2,
   itemkey in varchar2);
```

### 説明

指定した新規のフォーク・プロセスの実行を開始します。StartForkProcess() をコールする前に、CreateForkProcess() をコールして新規プロセスを作成する必要があります。StartForkProcess() をコールする前に、新規プロセスの項目属性を変更できます。

新規プロセスでオリジナルと同じバージョンを使用する場合、StartForkProcess() では、フォークされるプロセス内の各アクティビティのステータスと履歴が個別にコピーされます。新規プロセスで最新バージョンを使用する場合、StartForkProcess() では StartProcess() が実行されます。

プロセス内で StartForkProcess() をコールすると、そのプロセス内の「有効」ステータスになっている関連アクティビティが「通知済」ステータスに更新されます。その後、CompleteActivity() をコールしてプロセスを継続する必要があります。

StartForkProcess() では、項目属性に基づく通知属性が自動的にリフレッシュされます。元のプロセスにあるオープン通知は新規プロセスにコピーされ、再度送信されます。クローズ済の通知はコピーされますが、再送はされず、ステータスは「完了」のまま変わりません。

新規プロセス内の「待機」アクティビティは、元のアクティビティと同時に有効化されます。たとえば、元のプロセス内の 24 時間の「待機」アクティビティが、2 時間で適格となる場合は、新規の「待機」アクティビティも 2 時間で適格となります。

### 引数（入力）

itemtype	有効な項目タイプ。
itemkey	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。

---

---

**注意：** CreateForkProcess() と StartForkProcess() は、プロセスの並列分岐からはコールしないでください。この 2 つの API では、自分の分岐と並行している有効でない分岐をコピーしません。

---

---

## Background

### PL/SQL 構文

```
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);
```

### 説明

遅延されたアクティビティ、タイムアウトになったアクティビティおよびスタック状態を処理するために、指定のパラメータを使用してバックグラウンド・エンジンを実行します。バックグラウンド・エンジンは、呼び出された時点で指定された引数を満たす全アクティビティを実行します。このプロシージャは、長時間は実行されないため、定期的に再起動する必要があります。現行のバックグラウンド・エンジンの起動後に新しく延期されたアクティビティやタイムアウトになったアクティビティ、またはスタック状態は、次に起動されるバックグラウンド・エンジンによって処理されます。スクリプト `wfbkgchk.sql` を実行すると、次のバックグラウンド・エンジンの実行による処理待ちのアクティビティのリストを取得できます。16-7 ページの「[wfbkgchk.sql](#)」を参照してください。

Oracle Workflow のスタンドアロン版を使用している場合は、後述するバックグラウンド・エンジンのループ・スクリプトのどちらかを使用するか、独自のスクリプトを作成して、バックグラウンド・エンジンのプロシージャを無限にループさせることができます。Oracle Applications embedded Workflow を使用している場合は、このプロシージャのコンカレント・プログラム版を使用し、コンカレント・マネージャを利用して、バックグラウンド・エンジンを定期的に実行するようにスケジュールを設定できます。2-42 ページの「[バックグラウンド・エンジンのスケジューリング](#)」を参照してください。

### 引数（入力）

<b>itemtype</b>	有効な項目タイプ。項目タイプが NULL の場合、ワークフロー・エンジンはすべての項目タイプで実行されます。
<b>minthreshold</b>	(オプション) このバックグラウンド・エンジンで処理されるアクティビティの、最小コストのしきい値。100 分の 1 秒単位で指定します。このパラメータが NULL の場合、最小コストのしきい値はありません。



<b>maxthreshold</b>	(オプション) このバックグラウンド・エンジンで処理されるアクティビティの、最大コストのしきい値。100 分の 1 秒単位で指定します。このパラメータが NULL の場合、最大コストのしきい値はありません。
<b>process_deferred</b>	遅延されたプロセスを実行するかどうかを示す TRUE または FALSE を指定します。デフォルト値は TRUE です。
<b>process_timeout</b>	タイムアウトになったプロセスを実行するかどうかを示す TRUE または FALSE を指定します。デフォルト値は FALSE です。
<b>process_stuck</b>	スタック状態のプロセスを実行するかどうかを示す TRUE または FALSE を指定します。デフォルト値は FALSE です。

## バックグラウンド・エンジンのループ・スクリプト例

Oracle Workflow のスタンドアロン版では、2 つのサンプル・スクリプトのどちらかを使用して、バックグラウンド・エンジンを定期的に繰り返して実行できます。

最初の例は、wfbkg.sql ファイルに保存されている sql スクリプトで、サーバーの Oracle Workflow の admin/sql サブディレクトリにあります。このスクリプトを実行するには、ファイルが入っているディレクトリに移動し、オペレーティング・システムのプロンプトから次のコマンドを入力します。

```
sqlplus <username/password> @wfbkg <min> <sec>
<username/password> を、バックグラウンド・エンジンを実行する Oracle データベース・アカウントのユーザー名とパスワードに置き換えます。<min> をバックグラウンド・エンジンを実行する時間 (分) に置き換え、<sec> をコール間でバックグラウンド・エンジンを休止させる秒数に置き換えます。
```

2 番目の例は、wfbkg.csh ファイルに保存されているシェル・スクリプトで、サーバーの Oracle\_Home の bin サブディレクトリにあります。このスクリプトを実行するには、ファイルが入っているディレクトリに移動し、オペレーティング・システムのプロンプトから次のコマンドを入力します。

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

```
<username/password> を、バックグラウンド・エンジンを実行する Oracle データベース・アカウントのユーザー名とパスワードに置き換えます。
```

## AddItemAttribute

### PL/SQL 構文

```
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 構文

```
public static boolean addItemAttr
(
  WFCContext wCtx,
  String itemType,
  String itemKey,
  String aName)
public static boolean addItemAttrText
(
  WFCContext wCtx,
  String itemType,
  String itemKey,
  String aName,
  String aValue)
public static boolean addItemAttrNumber
(
  WFCContext wCtx,
  String itemType,
  String itemKey,
  String aName,
  BigDecimal aValue)
public static boolean addItemAttrDate
(
  WFCContext wCtx,
  String itemType,
  String itemKey,
  String aName,
  String aValue)
```

### 説明

プロセスに新規の項目タイプ属性の変数を追加します。ほとんどの項目タイプ属性は設計時に定義されますが、特定のプロセスに対して実行時に新しい属性を作成できます。属性の作成時に、オプションで新しい項目タイプ属性のデフォルトのテキスト値、数値または日付値を設定できます。

---

---

**注意：** Java を使用している場合は、属性タイプに対して適切なメソッドを選択してください。空の項目タイプ属性を追加するには、`addItemAttr()` を使用します。デフォルト値を持つ項目タイプ属性を追加するには、数値と日付以外のすべての属性タイプに `addItemAttrText()` を使用します。

---

---

---

---

**注意：** 一度に多数の項目タイプ属性を追加する必要がある場合は、パフォーマンスを改善するために、API は `AddItemAttribute` ではなく `AddItemAttributeArray` を使用します。8-49 ページの「[AddItemAttributeArray](#)」を参照してください。

---

---

## 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。
<b>aname</b>	項目タイプ属性の内部名。
<b>text_value</b>	項目タイプ属性のデフォルトのテキスト値。PL/SQL プロシージャの場合にのみ必須です。デフォルト値は NULL です。
<b>number_value</b>	項目タイプ属性のデフォルトの数値。PL/SQL プロシージャの場合にのみ必須です。デフォルト値は NULL です。
<b>date_value</b>	項目タイプ属性のデフォルトの日付値。PL/SQL プロシージャの場合にのみ必須です。デフォルト値は NULL です。
<b>aValue</b>	項目タイプ属性のデフォルト値。Java メソッド <code>addItemAttrText()</code> 、 <code>addItemAttrNumber()</code> および <code>addItemAttrDate()</code> の場合にのみ必須です。

## 例

次の例は、`AddItemAttr()` を使用して新しい項目タイプ属性の作成時にデフォルト値を設定し、API コールを簡素化する方法を示しています。

`AddItemAttr()` を使用して新規属性を作成し、`SetItemAttrText()` を使用して属性の値を設定するには、次のコールが必須です。

```
AddItemAttr('ITYPE', 'IKEY', 'NEWCHAR_VAR');  
SetItemAttrText('ITYPE', 'IKEY', 'NEWCHAR_VAR',  
                'new text values');
```

新規属性の作成と属性値の設定の両方に `AddItemAttr()` を使用する場合は、次のコールのみですみます。

```
AddItemAttr('ITYPE', 'IKEY', 'NEWCHAR_VAR',  
            'new text values');
```

# AddItemAttributeArray

## PL/SQL 構文

```
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);
```

## 説明

プロセスに新しい項目タイプ属性の配列を追加します。ほとんどの項目タイプ属性は設計時に定義されますが、特定のプロセスに対して実行時に新しい属性を作成できます。一度に多数の項目タイプ属性を追加する必要がある場合は、パフォーマンスを改善するために、API は `AddItemAttribute` ではなく `AddItemAttributeArray` を使用します。

属性タイプに対して適切なプロシージャを使用してください。数値と日付を除き、すべての属性タイプには `AddItemAttrTextArray` を使用します。

**注意：** `AddItemAttributeArray` API では、`WF_ENGINE` パッケージに定義されている PL/SQL 表の複合データ型が使用されます。次の表に、列のデータ型定義を PL/SQL 表タイプごとに示します。

表 8-1

PL/SQL 表タイプ	列のデータ型定義
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

引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。 この文字列により、項目タイプの項目が一意に識別されます。項目 タイプと項目キーにより、プロセスが識別されます。8-22 ページ の「 <a href="#">CreateProcess</a> 」を参照してください。
<b>aname</b>	新しい項目タイプ属性の内部名の配列。
<b>avalue</b>	新しい項目タイプ属性の値の配列。

## SetItemAttribute

### PL/SQL 構文

```
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 構文

```
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)
```

説明

プロセスの項目タイプ属性の値を設定します。属性タイプに対して適切なプロシージャを使用してください。数値、日付およびイベントを除き、すべての属性タイプには `SetItemAttrText` を使用します。

**注意：** 一度に多数の項目タイプ属性の値を設定する必要がある場合は、パフォーマンスを改善するために、API は `SetItemAttribute` ではなく `SetItemAttributeArray` を使用します。8-56 ページの「[SetItemAttributeArray](#)」を参照してください。

引数（入力）

<code>wCtx</code>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<code>itemtype</code>	有効な項目タイプ。
<code>itemkey</code>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。

**注意：** `itemkey` として `#SYNCH` を渡して、強制同期プロセスを作成できません。8-15 ページの「[同期プロセス、非同期プロセスおよび強制同期プロセス](#)」を参照してください。

<code>aname または name</code>	項目タイプ属性の内部名。
<code>avalue または event</code>	項目タイプ属性の値。

例 1

次のコードは、Java プログラムで `setItemAttrText()` をコールする方法の例です。このコード例は、`WFTest.java` プログラムからの引用です。

```
if (WFEngineAPI.setItemAttrText(ctx, iType, iKey,  
"REQUESTOR_USERNAME", owner))
```



```

        System.out.println("Requestor:"+owner);
    else
    {
        WFEEngineAPI.showError(ctx);
    }
}

```

## 例 2

イベント・メッセージをイベント・タイプの項目属性に保存した場合、イベント・メッセージのイベント・データ（CLOB）にアクセスするには、URL タイプの項目属性を作成してイベント・データを参照します。次の PL/SQL コード例では、URL 属性の値を設定してイベント・データを参照しています。

```

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);

```

スタイルシートをイベント・データ（XML 文書）に適用して HTML を作成する場合は、URL の p\_mime\_type パラメータを text/html に設定します。

URL の p\_mime\_type パラメータを省略すると、MIME タイプはデフォルトの text/xml に設定されます。

### 関連項目：

8-248 ページ [「イベント・メッセージ構造」](#)

## SetItemAttrDocument

---

---

**注意：** 文書管理機能は今後使用する目的で確保されています。  
SetItemAttrDocument API に関する説明は、参考のために記載しています。

---

---

### PL/SQL 構文

```
procedure SetItemAttrDocument
  (itemtype in varchar2,
   itemkey in varchar2,
   aname in varchar2,
   documentid in varchar2);
```

### Java 構文

```
public static boolean setItemAttrDocument
  (WFContext wCtx,
   String itemType,
   String itemKey,
   String aName,
   String documentId)
```

### 説明

文書 ID に、文書タイプの項目属性の値を設定します。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。

---

---

**注意：** itemkey として #SYNCH を渡して、強制同期プロセスを作成できません。8-15 ページの「[同期プロセス、非同期プロセスおよび強制同期プロセス](#)」を参照してください。

---

---

<b>aname</b>	項目タイプ属性の内部名。
<b>documentid</b>	<p>項目タイプ属性の値で、次の値を完全に連結した文字列となります。</p> <p>DM:&lt;node_id&gt;:&lt;doc_id&gt;:&lt;version&gt;</p> <p>&lt;node_id&gt; は、「文書管理ノード」 Web 画面で定義されている、文書管理システム・ノードに割り当てられたノード ID です。</p> <p>&lt;doc_id&gt; は文書の文書 ID で、文書が保存されている文書管理システムによって割り当てられています。</p> <p>&lt;version&gt; は文書のバージョンです。バージョンを指定しなければ、最新バージョンとみなされます。</p>

SetItemAttributeArray

PL/SQL 構文

```
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);
```

説明

プロセスの項目タイプ属性の配列の値を設定します。一度に多数の項目タイプ属性の値を設定する必要がある場合は、パフォーマンスを改善するために、API は `SetItemAttribute` ではなく `SetItemAttributeArray` を使用します。

属性タイプに対して適切なプロシージャを使用してください。数値と日付を除き、すべての属性タイプには `SetItemAttrTextArray` を使用します。

**注意：** `SetItemAttributeArray` API では、WF\_ENGINE パッケージに定義されている PL/SQL 表の複合データ型が使用されます。次の表に、列のデータ型定義を PL/SQL 表タイプごとに示します。

表 8-2

PL/SQL 表タイプ	列のデータ型定義
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

## 引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。 この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。
<b>aname</b>	項目タイプ属性の内部名の配列。
<b>avalue</b>	項目タイプ属性の値の配列。

## 例

次の例は、`SetItemAttribute` ではなく `SetItemAttributeArray` API を使用して、データベースのコール数を減らす方法を示しています。

`SetItemAttrText()` を使用する場合：

```
SetItemAttrText('ITYPE', 'IKEY', 'VAR1', 'value1');
SetItemAttrText('ITYPE', 'IKEY', 'VAR2', 'value2');
SetItemAttrText('ITYPE', 'IKEY', 'VAR3', 'value3');
```

// Multiple calls to update the database.

`SetItemAttrTextArray()` を使用する場合：

```
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.
```

## getItemTypes

### Java 構文

```
public static WFTwoDDataSource getItemTypes  
    (WFContext wCtx)
```

### 説明

Oracle Workflow データベースで定義されているすべての項目タイプのリストを、2次元のデータ・オブジェクトとして戻します。

### 引数（入力）

**wCtx**

ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「[Oracle Workflow のコンテキスト](#)」を参照してください。

## GetItemAttribute

### PL/SQL 構文

```
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;
```

### 説明

プロセスの項目タイプ属性の値を戻します。属性タイプに対して適切な関数を使用してください。数値、日付およびイベントを除き、すべての属性タイプには **GetItemAttrText** を使用します。

### 引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。

---

---

**注意：** itemkey として #SYNCH を渡して、強制同期プロセスを作成します。8-15 ページの「[同期プロセス、非同期プロセスおよび強制同期プロセス](#)」を参照してください。

---

---

<b>aname</b>	項目タイプ属性の内部名（ <code>GetItemAttrText()</code> 、 <code>GetItemAttrNumber()</code> および <code>GetItemAttrDate()</code> の場合）。
<b>avalue</b>	項目タイプ属性の内部名（ <code>GetItemAttrEvent()</code> の場合）。

**関連項目：**

8-248 ページ [「イベント・メッセージ構造」](#)



## GetItemAttrDocument

---

**注意：** 文書管理機能は今後使用する目的で確保されています。  
GetItemAttrDocument API に関する説明は、参考のために記載しています。

---

### PL/SQL 構文

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

### 説明

DM 文書タイプの項目属性の文書識別子を戻します。文書 ID は、次の値を連結した文字列となります。

DM:<nodeid>:<documentid>:<version>

<nodeid> は、「文書管理ノード」Web 画面で定義されている、文書管理システム・ノードに割り当てられたノード ID です。

<documentid> は文書の文書 ID で、文書が保存されている文書管理システムによって割り当てられています。

<version> は文書のバージョンです。バージョンを指定しなければ、最新バージョンとみなされます。

### 引数（入力）

itemtype	有効な項目タイプ。
itemkey	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。

---

**注意：** itemkey として #SYNCH を渡して、強制同期プロセスを作成します。8-15 ページの「[同期プロセス](#)、[非同期プロセス](#)および[強制同期プロセス](#)」を参照してください。

---

aname	項目タイプ属性の内部名。
-------	--------------

## GetItemAttrClob

### PL/SQL 構文

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

### 説明

プロセスの項目タイプ属性の値をキャラクタ・ラージ・オブジェクト（CLOB）として戻します。

### 引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。
<b>aname</b>	項目タイプ属性の内部名。

## getItemAttributes

### Java 構文

```
public static WFTwoDDataSource getItemAttributes  
(WFContext wCtx,  
    String itemType,  
    String itemKey)
```

### 説明

指定された項目タイプ・インスタンスに対するすべての項目属性およびそのタイプと値のリストを、2次元のデータ・オブジェクトとして戻します。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。

## GetItemAttrInfo

### PL/SQL 構文

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

### 説明

タイプや書式など、項目タイプ属性に関して指定されている情報があれば、それを返します。現在、項目タイプ属性のサブタイプ情報は使用できません。

### 引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>aname</b>	項目タイプ属性の内部名。

## GetActivityAttrInfo

### PL/SQL 構文

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

### 説明

タイプや書式など、アクティビティ属性に関して指定されている情報があれば、それを返します。現在、このプロシージャは、アクティビティ属性のサブタイプ情報は返しません。

### 引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。
<b>actid</b>	アクティビティ ID。プロセス定義内のアクティビティに関する、特定の使用方法を示します。この ID は、ノードのアクティビティ ID とも呼ばれます。
<b>aname</b>	アクティビティ属性の内部名。

## GetActivityAttribute

### PL/SQL 構文

```
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;
```

### 説明

プロセスのアクティビティ属性の値を戻します。属性タイプに対して適切な関数を使用してください。属性が「数値」または「日付」タイプの場合は、該当する関数が属性の書式を使用して、その数値や日付の値をテキスト文字列表現に変換します。

---

---

**注意：**「フォーム」、「URL」、「選択肢」および「文書」属性タイプには、GetActivityAttrText() を使用します。

---

---

### 引数（入力）

**itemtype**                      有効な項目タイプ。

**itemkey**                      アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「[CreateProcess](#)」を参照してください。

---

---

**注意：** itemkey として #SYNCH を渡して、強制同期プロセスを作成します。8-15 ページの「[同期プロセス、非同期プロセスおよび強制同期プロセス](#)」を参照してください。

---

---

**actid**                      アクティビティ ID。プロセス定義内のアクティビティに関する、特定の使用方法を示します。この ID は、ノードのアクティビティ ID とも呼ばれます。

**aname**                      アクティビティ属性の内部名 (GetActivityAttrText()、GetActivityAttrNumber() および GetActivityAttrDate() の場合)。

**name**                      アクティビティ属性の内部名 (GetActivityAttrEvent() の場合)。

**関連項目：**

8-248 ページ「[イベント・メッセージ構造](#)」

## GetActivityAttrClob

### PL/SQL 構文

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

### 説明

プロセスのアクティビティ属性の値をキャラクタ・ラージ・オブジェクト（CLOB）として戻します。

### 引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。
<b>actid</b>	アクティビティ ID。プロセス定義内のアクティビティに関する、特定の使用方法を示します。この ID は、ノードのアクティビティ ID と呼ばれます。
<b>aname</b>	アクティビティ属性の内部名。



## BeginActivity

### PL/SQL 構文

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

### 説明

指定されたアクティビティをプロセス項目に対して実行可能かどうかを判断し、実行できない場合は例外を発行します。

`CompleteActivity()` プロシージャは、検証の一部として自動的にこの関数を実行します。ただし、`BeginActivity` を使用すると、実行するアクティビティを実際にコールする前に現在実行できるかどうかを検証できます。8-71 ページの「[CompleteActivity](#)」を参照してください。

### 引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。
<b>activity</b>	プロセスで実行されるアクティビティ・ノード。アクティビティ・ノードのラベル名を指定します。アクティビティ・ノードのラベル名で特定のアクティビティ・ノードを識別できない場合は、ラベル名の前に親プロセスの内部名を追加できます。たとえば、 <code>&lt;parent_process_internal_name&gt;:&lt;label_name&gt;</code> と指定します。

### 例

```
/*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 構文

```
procedure CompleteActivity
(itemtype in varchar2,
 itemkey in varchar2,
 activity in varchar2,
 result_code in varchar2);
```

### 説明

特定の項目について指定されたアクティビティが完了したことを、ワークフロー・エンジンに通知します。このプロシージャは、次の状況でコールできます。

- **完了したアクティビティとオプションの結果を通知する場合：** ワークフロー・エンジンに非同期アクティビティが完了したことを知らせます。このプロシージャを実行するには、現在、アクティビティのステータスが「通知済」になっている必要があります。オプションでアクティビティの完了結果も渡すことができます。この結果により、プロセスの次のトランジションを判別できます。
- **新規項目を作成して開始する場合：** 「開始」アクティビティで新規項目が暗黙的に作成されて開始されるように、**CompleteActivity()** をコールします。「開始」アクティビティは、ワークフロー・ビルダーでプロセスの始めとして指定されています。このコールで指定する項目タイプおよびキーは、この項目を処理する後続のすべてのコールに渡す必要があります。

**CreateProcess()** と **StartProcess()** を使用してプロセスを開始できない場合は、**CompleteActivity()** を使用してください。たとえば、プロセス・スレッドの始めではなく途中にあるアクティビティ・ノードでプロセスを開始する必要がある場合は、**CompleteActivity()** をコールします。プロセスの始めとして指定するアクティビティ・ノードは、そのプロパティ画面の「ノード」タブで「開始」に設定しないと、エラーが発生します。

**注意：** プロセスの開始に `CompleteActivity()` を使用方法と、`CreateProcess()` および `StartProcess()` を使用方法には、次のような違いがあります。

- `CompleteActivity()` でコールする「開始」アクティビティには、入力トランジションがあってもなくてもかまいません。`StartProcess()` では、入力トランジションのない「開始」アクティビティのみが実行されます。
- `CompleteActivity()` では、コールされた単一の「開始」アクティビティのみが完了します。プロセス内の他の「開始」アクティビティは完了しません。ただし、`StartProcess()` では、プロセス内で「開始」アクティビティとしてマーク付けされていて、入力トランジションを持たない各アクティビティが実行されます。
- `CompleteActivity()` では、コールされたアクティビティは実行されず、完了マークのみが設定されます。`StartProcess()` では、プロセスの開始時の「開始」アクティビティが実行されます。
- `CompleteActivity()` を使用して新規プロセスを開始する場合、完了するアクティビティの項目タイプには、ルート・プロセスを選択するセレクタ関数が定義されているか、「開始」アクティビティとしてマークされていて完了するアクティビティには、実行可能なプロセスが1つのみである必要があります。`StartProcess()` のように、ルート・プロセスを明示的に指定することはできません。

引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。
<b>activity</b>	完了したアクティビティ・ノードの名前。アクティビティ・ノードのラベル名を指定します。アクティビティ・ノードのラベル名で特定のサブプロセスを識別できない場合は、ラベル名の前に親プロセスの内部名を追加できます。たとえば、 <parent_process_internal_name>:<label_name> のように指定します。このアクティビティ・ノードは、「開始」アクティビティとしてマークされている必要があります。
<b>result_code</b>	オプションのアクティビティの完了結果。有効な値は、プロセス・アクティビティの「結果タイプ」、またはエンジンの標準の結果の1つによって決まります。8-39 ページの「 <a href="#">AbortProcess</a> 」を参照してください。

### 例 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);
```

### 例 2

/\*Complete the 'LEGAL REVIEW' activity with status 'APPROVED'.The item must already exist.\*/

```
wf_engine.CompleteActivity('ORDER', '1003', 'LEGAL_REVIEW',  
                           'APPROVED');
```

### 例 3

/\*Complete the BLOCK activity which is used in multiple subprocesses in parallel splits.\*/

```
wf_engine.CompleteActivity('ORDER', '1003', 'ORDER_PROCESS:BLOCK-3',  
                           'null');
```

## CompleteActivityInternalName

### PL/SQL 構文

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

### 説明

特定の項目について指定されたアクティビティが完了したことを、ワークフロー・エンジンに通知します。このプロシージャを実行するには、現在、アクティビティのステータスが「通知済」になっている必要があります。オプションでアクティビティの完了結果も渡すことができます。この結果により、プロセスの次のトランジションを判別できます。

CompleteActivityInternalName() は CompleteActivity() に似ています。ただし、CompleteActivityInternalName() では完了したアクティビティがその内部名で識別されるのに対して、CompleteActivity() ではアクティビティがアクティビティ・ノードのラベル名で識別されます。CompleteActivityInternalName() は、アクティビティ・ノードのラベル名が不明な場合にのみ使用してください。アクティビティ・ノードのラベル名がわかっている場合は、かわりに CompleteActivity() を使用します。8-71 ページの「[CompleteActivity](#)」を参照してください。

---

---

**注意：** CompleteActivity() とは異なり、CompleteActivityInternalName() はプロセスの開始には使用できません。また、CompleteActivityInternalName() は同期プロセスにも使用できません。

---

---

CompleteActivityInternalName() の実行時には、「通知済」ステータスになっている指定のアクティビティ・インスタンスが 1 つのみである必要があります。複数のアクティビティ・インスタンスが「通知済」ステータスになっていると、プロセスのステータスが「ERROR」になります。

### 引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。

<b>activity</b>	完了したアクティビティの内部名。アクティビティの内部名で特定のサブプロセスを識別できない場合は、アクティビティの内部名の前に親プロセスの内部名を追加できます。たとえば、 <code>&lt;parent_process_internal_name&gt;:&lt;activity_internal_name&gt;</code> のように指定します。
<b>result</b>	オプションのアクティビティの完了結果。有効な値は、プロセス・アクティビティの「結果タイプ」、またはエンジンの標準の結果の1つによって決まります。8-39 ページの「 <a href="#">AbortProcess</a> 」を参照してください。

## AssignActivity

### PL/SQL 構文

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

### 説明

アクティビティを別の実行者に割り当てるか、再割当てします。あるアクティビティに進む前に、このプロシージャをコールできます。たとえば、プロセス内の前の関数アクティビティで、後のアクティビティの実行者を決定できます。

すでに通知を処理中の通知アクティビティに新規ユーザーが割り当てられると、処理中の通知は取り消され、WF\_Notification.Transfer がコールされ、その新規ユーザー用に新しい通知が生成されます。

### 引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。
<b>activity</b>	アクティビティ・ノードのラベル名。アクティビティ・ノードのラベル名で特定のアクティビティ・ノードを識別できない場合は、ラベル名の前に親プロセスの内部名を追加できます。たとえば、 <code>&lt;parent_process_internal_name&gt;:&lt;label_name&gt;</code> のように指定します。
<b>performer</b>	アクティビティを実行するユーザー（通知を受け取るユーザー）の名前。この名前は、Oracle Workflow ディレクトリ・サービスのロール名である必要があります。



## Event

### PL/SQL 構文

```
procedure Event
(
  itemtype in varchar2,
  itemkey in varchar2,
  process_name in varchar2 default null,
  event_message in wf_event_t);
```

### 説明

ビジネス・イベント・システムからイベントを受信し、ワークフロー・プロセスに渡します。

指定された項目キーがすでに存在する場合、受信したイベントはその項目に渡されます。項目キーが存在しない場合、指定されたプロセスに組み込まれている「受信」イベント・アクティビティが「開始」アクティビティとしてマークされ、有効であるときは、ワークフロー・エンジンによってそのプロセスを実行する項目が新しく作成されます。

このプロシージャは、イベントを受信するワークフロー・プロセス内で、条件を満たす「受信」イベント・アクティビティを検索します。特定のアクティビティが特定のイベントを受信するには、そのイベント・フィルタに空白またはそのイベントを設定する必要があります。また、そのアクティビティは、以下のステータス要件を満たす必要があります。

- 「開始」アクティビティとしてマークされたアクティビティは、実行されていないイベントだけを受信できます。
- 通常のアクティビティは、アクティビティ・ステータスが「NOTIFIED」である場合にのみイベントを受信できます。「NOTIFIED」は、プロセスがそのアクティビティに遷移し、イベントの受信を待機していることを意味します。

*Event()* は、条件を満たす「受信」イベント・アクティビティごとに、イベント名、イベント・キーおよびイベント・メッセージを、イベント・アクティビティ・ノードに指定された項目タイプ属性に格納します。また、このプロシージャは、イベント・メッセージのパラメータ・リストのパラメータをプロセスの項目タイプ属性として設定します。パラメータに対応する項目タイプ属性が存在しない場合は、新しい項目タイプ属性を作成します。ワークフロー・エンジンは、イベント・アクティビティからスレッドの実行を開始します。

イベントを受信する条件を満たす「受信」イベント・アクティビティが存在しない場合は、プロシージャから例外およびエラー・メッセージが返されます。

**注意：**「受信」イベント・アクティビティが受信したイベントが別のワークフロー・プロセスの「呼出し」イベント・アクティビティによって発生した場合は、そのプロセスの項目タイプと項目キーがイベント・メッセージ内のパラメータ・リストに追加されます。ワークフロー・エンジンは、イベントを受信するプロセスの親として、指定されたプロセスを自動的に設定し、既存の親の設定を上書きします。8-81 ページの「[SetItemParent](#)」を参照してください。

引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	項目タイプの項目を一意に識別するための文字列。項目タイプと項目キーにより、プロセスが識別されます。
<b>process_name</b>	その項目タイプの特定のサブプロセスを選択するためのオプションの引数。プロセス・アクティビティのラベル名を指定します。プロセス・アクティビティのラベル名で特定のサブプロセスを識別できない場合は、ラベル名の前に親プロセスの内部名を追加できます。たとえば、<parent_process_internal_name>:<label_name> のように指定します。この引数が NULL の場合は、項目の最上位レベルのプロセスが開始されます。この引数のデフォルト値は NULL です。
<b>event_message</b>	イベントの詳細を含むイベント・メッセージ。

## HandleError

### PL/SQL 構文

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

### 説明

このプロシージャは通常、**ERROR** プロセスのアクティビティからコールされ、エラーが検出されたプロセス・アクティビティを処理します。

このプロシージャをプロセスの任意のアクティビティでコールして、一部のプロセスをそのアクティビティにロールバックすることもできます。このプロシージャでコールされるアクティビティはどんなステータスでもよく、実行中でなくてもかまいません。また、サブプロセス内のアクティビティでもよく、アクティビティ・ノード・ラベルがプロセス内で一意でなければ、そのアクティビティ・ノード・ラベル名の前に親プロセスの内部名を追加できます。たとえば、`<parent_process_internal_name>:<label_name>` のように指定します。

「再開封時」フラグが「リセット」に設定されていると、このプロシージャは指定されたアクティビティを消去し、**CANCEL** モードで各アクティビティを再実行し、すでに通過した後続のすべてのアクティビティも消去します。8-11 ページの「**ループ**」を参照してください。ステータスが「**ERROR**」になっているアクティビティには、その後に実行済アクティビティがないため、プロシージャはエラーのあるアクティビティのみを消去します。

アクティビティが消去されると、指定されたアクティビティの親プロセスがアクティブでない場合、このプロシージャはそれらの親プロセスすべてのステータスを「有効」にリセットします。

その後、このプロシージャは、**SKIP** または **RETRY** のうち、入力された方のコマンドに基づいて、指定されたアクティビティを処理します。

### 引数（入力）

<b>item_type</b>	有効な項目タイプ。
<b>item_key</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。

<b>activity</b>	エラーが発生したか、または元に戻るアクティビティ・ノード。アクティビティ・ノードのラベル名を指定します。アクティビティ・ノードのラベル名で特定のサブプロセスを識別できない場合は、ラベル名の前に親プロセスの内部名を追加できます。たとえば、 <code>&lt;parent_process_internal_name&gt;:&lt;label_name&gt;</code> のように指定します。
<b>command</b>	<p>プロセス・アクティビティの処理方法を決定する次の 2 つのコマンドのどちらか。</p> <p><b>SKIP:</b> アクティビティを再実行しないけれども、指定された結果でアクティビティが完了したものとしてマークを付け、そのアクティビティからプロセスを続行します。</p> <p><b>RETRY:</b> アクティビティを再実行し、そのアクティビティからプロセスを続行します。</p>
<b>result</b>	SKIP コマンドに必要な結果。

---

**注意：** 項目が作成された後は、その項目の有効日と、項目が進むプロセスのバージョン番号を変更できません。ただし、**HandleError** を使用すると、既存項目のプロセスを手動で変更できることがあります。

プロセスの変更が些細なものであれば、**HandleError** を使用して、項目をエラーになるアクティビティの次のアクティビティに手動で進めるか、プロセス内の別のトランジションにリダイレクトできます。

プロセスの変更が大幅なものであれば、少なくとも次の手順を実行する必要があります。

- **WF\_ENGINE.AbortProcess()** をコールしてプロセスを中止します。
  - **WF\_ENGINE.Items()** をコールして既存の項目を削除します。
  - プロセスを改訂します。
  - **WF\_ENGINE.CreateProcess()** をコールして項目を再作成します。
  - **WF\_ENGINE.HandleError()** をコールして、改訂したプロセスを該当するアクティビティで再開します。
-

## SetItemParent

### PL/SQL 構文

```
procedure SetItemParent
(
  itemtype in varchar2,
  itemkey in varchar2,
  parent_itemtype in varchar2,
  parent_itemkey in varchar2,
  parent_context in varchar2);
```

### 説明

マスター・プロセスとディテール・プロセスの親子関係を定義します。2つのプロセス間の親子関係を定義するには、この API をマスター・プロセスから作成されるディテール・プロセスでコールする必要があります。この API は、**CreateProcess API** をコールした後、ディテール・プロセスで **StartProcess API** をコールするまでにコールしてください。

### 引数（入力）

<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、子プロセスが識別されます。
<b>parent_itemtype</b>	親プロセスの有効な項目タイプ。
<b>parent_itemkey</b>	親項目タイプの項目を一意に識別するために、アプリケーション・オブジェクトの主キーから生成された文字列。親項目タイプおよびキーにより、親プロセスが識別されます。
<b>parent_context</b>	親に関するコンテキスト情報。

## ItemStatus

### PL/SQL 構文

```
procedure ItemStatus
  (itemtype in varchar2,
   itemkey in varchar2,
   status out varchar2,
   result out varchar2);
```

### Java 構文

```
public static WFTwoDataSource itemStatus
  (WFContext wCtx,
   String itemType,
   String itemKey)
```

### 説明

指定した項目インスタンスのルート・プロセスのステータスと結果を戻します。ステータスの有効値は、**ACTIVE**、**COMPLETE**、**ERROR** または **SUSPENDED** のいずれかです。ルート・プロセスが存在しない場合は項目キーが存在しないため、このプロシージャで例外が発生します。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>itemtype</b>	有効な項目タイプ。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、項目インスタンスが識別されます。

### 例

次のコードは、Java プログラムで `itemStatus()` をコールする方法の例です。このコード例は、`WFTest.java` プログラムからの引用です。

```
// get status and result for this item
dataSource = WFEEngineAPI.itemStatus(ctx, itemType, itemKey);
System.out.print("Status and result for " + itemType + "/" + itemKey + " = ");
displayDataSource(ctx, dataSource);
```

## getProcessStatus

### Java 構文

```
public static WFTwoDDataSource getProcessStatus
(WFContext wCtx,
 String itemType,
 String itemKey,
 BigDecimal process)
```

### 説明

指定された項目タイプ・インスタンスに対するプロセス・ステータスを、2次元のデータ・オブジェクトとして戻します。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>itemType</b>	有効な項目タイプ。
<b>itemKey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。
<b>process</b>	項目タイプのプロセス・インスタンス ID。インスタンス ID が不明な場合は、単に負数を指定できます。これにより、このメソッドはルート・プロセスのプロセス・ステータスを戻します。

## Workflow Function API

すべての外部 Java 関数アクティビティの Java プロシージャは、WFFunctionAPI Java クラスという抽象クラスから導出されます。このクラスには、項目タイプおよびアクティビティ属性にアクセスするメソッドの他に、実装される外部 Java 関数アクティビティのメイン・エントリ・ポイント関数を作成する `execute()` メソッドが含まれます。

WFFunctionAPI クラスは、`oracle.apps.fnd.wf` という Java パッケージに格納されています。次の API は、このクラスで利用できる API です。

---

---

**注意：** Java では大文字 / 小文字が区別されます。Java のネーミング規則に従って、すべての Java メソッド名の先頭文字は小文字となります。

---

---

- [8-85 ページ「loadItemAttributes」](#)
- [8-86 ページ「loadActivityAttributes」](#)
- [8-87 ページ「getActivityAttr」](#)
- [8-89 ページ「getItemAttr」](#)
- [8-90 ページ「setItemAttrValue」](#)
- [8-91 ページ「execute」](#)

**関連項目：**

[7-8 ページ「関数アクティビティがコールする Java プロシージャの標準 API」](#)

[4-42 ページ「関数アクティビティ」](#)



## loadItemAttributes

### Java 構文

```
public void loadItemAttributes  
    (WFContext pWCtx) throws SQLException
```

### 説明

外部 Java 関数をコールした項目タイプの項目属性をデータベースから取得します。項目属性はデフォルトではロードされません。項目タイプに多数の項目属性が含まれている場合は、パフォーマンスが低下する可能性があるためです。このメソッドは、関数から項目属性にアクセスする前に項目属性を明示的にロードするときに使用します。

データベース・アクセス・エラーが発生した場合、このメソッドは `SQLException` をスローします。

### 引数（入力）

**pWCtx**

ワークフローのコンテキスト情報。8-6 ページの「[Oracle Workflow のコンテキスト](#)」を参照してください。

## loadActivityAttributes

### Java 構文

```
public void loadActivityAttributes
    (WFContext pWCtx,
     String iType,
     String iKey,
     BigDecimal actid) throws SQLException
```

### 説明

データベースから特定のアクティビティのアクティビティ属性を取得します。このメソッドは、関数アクティビティがインスタンス化されるときおよび `execute()` 関数がコールされる前にデフォルトでコールされます。

データベース・アクセス・エラーが発生した場合、このメソッドは `SQLException` をスローします。

### 引数（入力）

<b>pWCtx</b>	ワークフローのコンテキスト情報。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>iType</b>	有効な項目タイプ。
<b>iKey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。8-22 ページの「 <a href="#">CreateProcess</a> 」を参照してください。
<b>actid</b>	アクティビティ・インスタンス ID。

## getActivityAttr

### Java 構文

```
public WFAttribute getActivityAttr  
    (String aName)
```

### Java 構文

```
public WFAttribute getActivityAttr  
    (WFContext pWCtx,  
     String aName) throws SQLException
```

### 説明

`getActivityAttr()` の実装には、次の 2 種類があります。これらのメソッドは、特定のアクティビティ属性のアクティビティ属性情報を返します。

- アクティビティ属性の名前だけを使用して `getActivityAttr(String aName)` をコールした場合、このメソッドはアクティビティ属性値を返しますが、項目属性への参照は解決しません。アクティビティ属性が項目属性を参照している場合、このメソッドは項目属性の内部名を返します。項目属性の名前を取得すると、その項目属性に基づいて追加処理を実行できます。

たとえば、項目属性に情報を書き込むには、最初に `getActivityAttr(String aName)` を使用して項目属性の名前を取得します。次に、`setItemAttrValue(WFContext pWCtx, WFAttribute pAttr)` を使用して項目属性の値を設定します。この値はアクティビティ属性の値にもなります。8-90 ページの「[setItemAttrValue](#)」を参照してください。

- Workflow のコンテキストとアクティビティ属性の名前を使用して `getActivityAttr(WFContext pWCtx, String aName)` をコールした場合、このメソッドはアクティビティ属性を返します。また、アクティビティ属性が項目属性を参照している場合、このメソッドはその項目属性の値を取得して参照を解決します。実際のアクティビティ属性の値を取得するときに、アクティビティ属性が参照している項目属性を取得する必要がない場合は、`getActivityAttr(WFContext pWCtx, String aName)` を使用できます。このメソッドは、直前にロードされた項目属性内の参照を解決します。項目属性がロードされていない場合は、`loadItemAttributes(WFContext pWCtx)` をコールして項目属性をロードします。8-85 ページの「[loadItemAttributes](#)」を参照してください。

データベース・アクセス・エラーが発生した場合、このメソッドは `SQLException` をスローします。

## 引数（入力）

<b>pWCtx</b>	ワークフローのコンテキスト情報。2 番目のメソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>aName</b>	アクティビティ属性の内部名。

## getItemAttr

### Java 構文

```
public WFAttribute getItemAttr  
    (String aName)
```

### 説明

特定の項目属性の項目属性情報を返します。

### 引数（入力）

<b>aName</b>	項目属性の内部名。
--------------	-----------

## setItemAttrValue

### Java 構文

```
public void setItemAttrValue  
    (WFContext pWCtx,  
     WFAAttribute pAttr)  
    throws NumberFormatException, WFException
```

### 説明

特定の項目属性の値をデータベースに設定します。

項目属性の値を数値または日付タイプの属性に適した形式に変換できない場合、このメソッドは `NumberFormatException` をスローします。文書またはテキスト・タイプの属性の設定中にエラーが発生した場合は、`WFException` をスローします。

### 引数（入力）

<b>pWCtx</b>	ワークフローのコンテキスト情報。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>pAttr</b>	項目属性の属性情報。

## execute

### Java 構文

```
public abstract boolean execute  
    (WFContext pWCtx)
```

### 説明

この抽象メソッドは、拡張クラスによって実装され、実装される外部 Java 関数アクティビティのメイン・エントリ・ポイント関数を作成します。7-3 ページの「[関数アクティビティがコールする PL/SQL プロシージャの標準 API](#)」を参照してください。

### 引数（入力）

pWCtx	ワークフローのコンテキスト情報。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
-------	---

# Workflow Attribute API

WFAttribute Java クラスには、属性の内部名、属性値、属性のデータ型、書式情報、デフォルトの値タイプなど、項目属性またはアクティビティ属性を表す情報が含まれます。属性値はオブジェクト・タイプとして格納されます。このクラスには、属性情報にアクセスするメソッドも含まれます。これらのメソッドは、Java アプリケーションまたは外部 Java 関数アクティビティの Java プロシージャからコールできます。

WFAttribute クラスは、oracle.apps.fnd.wf という Java パッケージに格納されています。次の API は、このクラスで利用できる API です。

**注意：** Java では大文字・小文字が区別されます。すべての Java メソッド名（コンストラクタ・メソッド名を除く）の先頭文字は、Java のネーミング規則に従って小文字で始まります。

- 8-94 ページ [「WFAttribute」](#)
- 8-95 ページ [「value」](#)
- 8-96 ページ [「getName」](#)
- 8-97 ページ [「getValue」](#)
- 8-98 ページ [「getType」](#)
- 8-99 ページ [「getFormat」](#)
- 8-100 ページ [「getValueType」](#)
- 8-101 ページ [「toString」](#)
- 8-102 ページ [「compareTo」](#)

**関連項目：**

[7-8 ページ「関数アクティビティがコールする Java プロシージャの標準 API」](#)

## WFAttribute クラスの定数

WFAttribute クラスには、いくつかの定数が含まれます。次の表は、属性のデータ型を表すときに使用できる定数の一覧です。

表 8-3

定数変数宣言	定数値
public static final String TEXT	"TEXT"
public static final String NUMBER	"NUMBER"



表 8-3 (続き)

定数変数宣言	定数値
<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"

次の表は、属性のデフォルト値のデータ型を表すときに使用できる定数の一覧です。デフォルト値には、定数、または項目タイプ属性への参照（アクティビティ属性の場合）を使用できます。

表 8-4

定数変数宣言	定数値
<code>public static final String CONSTANT</code>	"CONSTANT"
<code>public static final String ITEMATTR</code>	"ITEMATTR"

## WFAttribute

### Java 構文

```
public WFAttribute()
```

### Java 構文

```
public WFAttribute  
    (String pName  
     String pType,  
     Object pValue,  
     String pValueType)
```

### 説明

WFAttribute クラスには、2 つのコンストラクタ・メソッドがあります。1 番目のコンストラクタは、新しい WFAttribute オブジェクトを作成します。2 番目のコンストラクタは、新しい WFAttribute オブジェクトを作成し、特定の属性名、属性タイプ、値、および値タイプにオブジェクトを初期化します。

### 引数（入力）

<b>pName</b>	項目属性またはアクティビティ属性の内部名。2 番目のメソッドの場合にのみ必須です。
<b>pType</b>	属性のデータ型。2 番目のメソッドの場合にのみ必須です。
<b>pValue</b>	属性値。2 番目のメソッドの場合にのみ必須です。
<b>pValueType</b>	属性のデフォルト値のデータ型。デフォルト値には、定数、または項目タイプ属性への参照（アクティビティ属性の場合）を使用できます。2 番目のメソッドの場合にのみ必須です。

## value

### Java 構文

```
public void value  
    (Object pValue)
```

### 説明

WFAttribute オブジェクトの項目属性またはアクティビティ属性の値を設定します。この属性値は、オブジェクト・タイプに明示的に型変換する必要があります。

---

---

**注意：** value() を使用して WFAttribute オブジェクトの属性値を設定しても、データベースの属性値は設定されません。データベースの項目属性の値を設定するには、WFFunctionAPI.setItemAttrValue() を使用します。  
8-90 ページの「[setItemAttrValue](#)」を参照してください。

---

---

### 引数（入力）

pValue	属性値。
--------	------

## getName

### Java 構文

```
public String getName()
```

### 説明

項目属性またはアクティビティ属性の内部名を返します。

## getValue

### Java 構文

```
public Object getValue()
```

### 説明

項目属性またはアクティビティ属性の値をオブジェクト・タイプとして返します。

## getType

### Java 構文

```
public String getType()
```

### 説明

項目属性またはアクティビティ属性のデータ型を返します。4-3 ページの「[属性タイプ](#)」を参照してください。

## getFormat

### Java 構文

```
public String getFormat()
```

### 説明

テキスト・タイプの属性の長さや数値または日付タイプの属性の書式マスクなど、項目属性またはアクティビティ属性の書式文字列を返します。4-8 ページの「[項目タイプまたはアクティビティ属性の定義](#)」を参照してください。

## getValueType

### Java 構文

```
public String getValueType()
```

### 説明

項目属性またはアクティビティ属性のデフォルト値のデータ型を返します。デフォルト値には、定数、または項目タイプ属性への参照（アクティビティ属性の場合）を使用できます。4-8 ページの「[項目タイプまたはアクティビティ属性の定義](#)」を参照してください。



## toString

### Java 構文

```
public String toString()
```

### 説明

項目属性またはアクティビティ属性の内部名と値を、次の書式の文字列として返します。

`<name>=<value>`

このメソッドは、オブジェクト・クラスの `toString()` メソッドを無効にします。

## compareTo

### Java 構文

```
public int compareTo  
    (String pValue) throws Exception
```

### 説明

項目属性またはアクティビティ属性の値を特定の値と比較します。**compareTo()** は、2つの値が等しい場合は 0、指定した値より属性値が小さい場合は -1、指定した値より属性値が大きい場合は 1 を返します。

指定した値を数値または日付タイプの属性に適した書式に変換できない場合、このメソッドは例外をスローします。

### 引数（入力）

<b>pValue</b>	属性値と比較するテスト値。
---------------	---------------



## Workflow CORE API

関数アクティビティによってコールされる PL/SQL プロシージャでは、Oracle Workflow API の一連の CORE API を使用してエラーの発生および検出ができます。

関数アクティビティによってコールされる PL/SQL プロシージャで未処理例外が発生したり、「ERROR:」で始まる結果を戻すと、ワークフロー・エンジンは関数アクティビティのステータスを ERROR に設定し、列 ERROR\_NAME、ERROR\_MESSAGE および ERROR\_STACK を表 WF\_ITEM\_ACTIVITY\_STATUSES に設定してエラーを反映させます。

ERROR\_NAME 列と ERROR\_MESSAGE 列は、WF\_CORE.RAISE() のコールの戻り値に設定されるか、または RAISE() のコールがない場合は SQL エラー名とメッセージに設定されます。列 ERROR\_STACK は、エラー・ソースに関係なく WF\_CORE.CONTEXT() のコールで指定された内容に設定されます。

---

---

**注意：** 列 ERROR\_NAME、ERROR\_MESSAGE および ERROR\_STACK は、事前定義の「システム:エラー」項目タイプの項目タイプ属性としても定義されます。これらの列の情報は、アクティビティに関連付けるエラー・プロセスから参照できます。6-23 ページの「[デフォルト・エラー・プロセス](#)」を参照してください。

---

---

次の API は、ランタイム・フェーズでアプリケーション・プログラムまたはワークフロー関数によってコールされ、エラー処理を実行できます。これらの API は、WF\_CORE という PL/SQL パッケージに格納されています。

- 8-105 ページ [「CLEAR」](#)
- 8-106 ページ [「GET\\_ERROR」](#)
- 8-107 ページ [「TOKEN」](#)
- 8-108 ページ [「RAISE」](#)
- 8-111 ページ [「CONTEXT」](#)
- 8-113 ページ [「TRANSLATE」](#)

### 関連項目：

7-3 ページ [「関数アクティビティがコールする PL/SQL プロシージャの標準 API」](#)

## CLEAR

### 構文

```
procedure CLEAR;
```

### 説明

エラー・バッファを消去します。

#### 関連項目：

8-106 ページ [「GET\\_ERROR」](#)

## GET\_ERROR

### 構文

```
procedure GET_ERROR
    (err_name out varchar2,
     err_message out varchar2
     err_stack out varchar2);
```

### 説明

現行のエラー・メッセージ名およびトークンが置換されたエラー・メッセージを戻します。  
また、エラー・スタックを消去します。現行のエラーがない場合は、NULL を戻します。

### 例 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;
```

#### 関連項目：

8-106 ページ [「CLEAR」](#)

# TOKEN

## 構文

```
procedure TOKEN
(token_name in varchar2,
 token_value in varchar2);
```

## 説明

エラー・トークンを定義して値に置き換えます。TOKEN() と RAISE() のコールは、WF\_RESOURCES 表に保存されている Oracle Workflow の事前定義済のエラーを発生させます。エラー・メッセージには、その発生時に該当する値への置換が必要なトークンが含まれます。これは、PL/SQL の標準的な例外やカスタム定義の例外を発生させる代替方法です。

## 引数（入力）

token_name	トークン名
token_value	トークンを置き換える値

### 関連項目：

8-108 ページ [「RAISE」](#)

8-111 ページ [「CONTEXT」](#)

## RAISE

### 構文

```
procedure RAISE
    (name in varchar2);
```

### 説明

指定されたエラー・メッセージ名に対して正しいエラー番号とトークンが置換されたメッセージを渡し、コール側で例外を発生します。

TOKEN() と RAISE() のコールは、WF\_RESOURCES 表に保存されている Oracle Workflow の事前定義済みのエラーを発生させます。エラー・メッセージには、その発生時に該当する値への置換が必要なトークンが含まれます。これは、PL/SQL の標準的な例外やカスタム定義の例外を発生させる代替方法です。

Oracle Workflow のエラー・メッセージは、最初はメッセージ・ファイル (.msg) 内で定義されています。メッセージ・ファイルは、Oracle Workflow のスタンドアロン版の場合は、Oracle Workflow Server のディレクトリ構造の res/<language> サブディレクトリにあります。Oracle Applications に埋め込まれているバージョンの場合は、サーバー上の \$FND\_TOP の resource/<language> サブディレクトリにあります。Oracle Workflow Server の導入時に、ワークフロー・リソース・ジェネレータ・プログラムにより、指定されたメッセージ・ファイルが取り出され、メッセージが WF\_RESOURCES 表にインポートされます。

### 引数（入力）

name	WF_RESOURCES 表に格納されているエラー・メッセージの内部名。
------	--------------------------------------

#### 関連項目：

8-107 ページ [「TOKEN」](#)

8-111 ページ [「CONTEXT」](#)

### ➤ ワークフロー・リソース・ジェネレータの実行

#### Oracle Workflow のスタンドアロン版の場合：

1. ワークフロー・リソース・ジェネレータ・プログラムは、Oracle ホーム・ディレクトリ構造の bin サブディレクトリにあります。
2. このプログラムをオペレーティング・システム・プロンプトから次のように実行します。



- ソース・ファイル（.msg）からバイナリ・リソース・ファイルを生成するには、次のように入力します。

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

<resourcefile> を生成するリソース・ファイルのフルパス名に置き換え、<source\_file> をソース・ファイルのフルパス名に置き換えてください。オプションの -v フラグを付けると、ソース・ファイルはバイナリ・リソース・ファイルと比較検証されます。

- ソース・ファイル（.msg）からデータベース表 WF\_RESOURCES にシード・データをアップロードするには、次のように入力します。

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

<username/password@database> を、ユーザー名、パスワードおよびデータベースへの Oracle Net 接続文字列または別名に置き換え、<source\_file> をアップロードするソース・ファイルのフルパス名に置き換えます。オプションの -v フラグを付けると、ソース・ファイルはデータベースと比較検証されます。

### Oracle Applications embedded Workflow の場合

1. ワークフロー・リソース・ジェネレータ・プログラムは、コンカレント・プログラムとして登録されています。ワークフロー・リソース・ジェネレータ・コンカレント・プログラムは、「Submit Requests」フォームまたはコマンドラインから実行できます。
2. このコンカレント・プログラムを「Submit Requests」フォームから実行するには、「Submit Requests」フォームにナビゲートします。

---

**注意：** システム管理者は、このプログラムを実行する職責の要求セキュリティ・グループに、このコンカレント・プログラムを追加する必要があります。『Oracle Applications System Administrator's Guide』の「Overview of Concurrent Programs and Requests」を参照してください。

---

3. ワークフロー・リソース・ジェネレータ・コンカレント・プログラムを要求として発行します。『Oracle Applications User's Guide』の「Submitting a Request」を参照してください。
4. 「パラメータ」ウィンドウで次のパラメータの値を入力します。

#### 宛先タイプ

シード・データをソース・ファイル（.msg）からデータベース表 WF\_RESOURCES にアップロードする場合は「データベース」を指定し、ソース・ファイルからリソース・ファイルを生成する場合は「ファイル」を指定します。

**宛先** 「宛先タイプ」に「ファイル」を指定した場合は、生成するリソース・ファイルのフルパスと名前を入力します。「データベース」を指定した場合は、その宛先として現行のデータベース・アカウントが自動的に使用されます。

**ソース** ソース・ファイルのフルパス名を指定します。

5. 「OK」を選択して「パラメータ」ウィンドウを閉じます。
6. この要求の印刷オプションと実行オプションを変更してから、「発行」を選択して要求を発行します。
7. 「Submit Requests」フォームを使用せずに、次のどちらかのコマンドを入力し、ワークフロー・リソース・ジェネレータ・コンカレント・プログラムをコマンドラインから実行することもできます。ソース・ファイルからリソース・ファイルを生成するには、次のように入力します。

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

ソース・ファイルからデータベース表 WF\_RESOURCES にシード・データをアップロードするには、次のように入力します。

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

`apps/pwd` を APPS スキーマのユーザー名とパスワードに置き換え、`res_file` をリソース・ファイルのファイル仕様に置き換え、`source_file` をソース・ファイル（.msg）のファイル仕様に置き換えます。ファイル仕様は次のように指定します。

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

または

```
<native path>
```

## CONTEXT

### 構文

```
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*');
```

### 説明

エラー・スタックにエントリを追加し、エラー・ソースの検索に役立つコンテキスト情報を入力します。このプロシージャは、**TOKEN()** と **RAISE()** のコールで発生した事前定義済のエラーやカスタム定義の例外で使用するか、またはエラー条件が検出されたときに例外なしで使用できます。

### 引数（入力）

<b>pkg_name</b>	プロシージャのパッケージ名
<b>proc_name</b>	プロシージャ名または関数名
<b>arg1</b>	最初の IN 引数
<b>argn</b>	n 番目の IN 引数

### 例 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;
```

**関連項目：**

8-107 ページ [「TOKEN」](#)

8-108 ページ [「RAISE」](#)

## TRANSLATE

### 構文

```
function TRANSLATE  
    (tkn_name IN VARCHAR2)  
    return VARCHAR2;
```

### 説明

WF\_RESOURCES で言語設定に従って定義されている値をトークンに戻して、トークン文字列の値を変換します。

### 引数（入力）

<b>tkn_name</b>	トークン名
-----------------	-------

## Workflow PURGE API

次の API は、ランタイム・フェーズでアプリケーション・プログラムまたはワークフロー関数によってコールされ、不要になったランタイム・データを削除します。これらの API は、WF\_PURGE という PL/SQL パッケージに定義されています。

WF\_PURGE を使用して、完了した項目やプロセスに関する不要なランタイム・データや、使用されなくなった廃止アクティビティのバージョン情報を削除できます。このような廃止データをシステムから定期的に削除することで、パフォーマンスが向上します。

WF\_PURGE パッケージの PL/SQL 変数 `persistence_type` は、`TotalPerm()` を除くすべての WF\_PURGE API がどのように動作するかを定義します。この変数が `TEMP` に設定されている場合は、「一時」（数日間保管）項目タイプに関連付けられているデータのみが削除されます。`persistence_type` はデフォルトで `TEMP` に設定されており、変更できません。維持タイプが「永続」の項目に関するランタイム・データを削除する場合は、`TotalPerm()` を使用する必要があります。4-4 ページの「[維持タイプ](#)」を参照してください。

---

**注意：** 将来の終了日付に WF\_PURGE API を実行することはできません。先の終了日付を指定すると、「一時」の項目タイプの維持期間を超えてしまうことがあります。先の終了日付を指定すると、WF\_PURGE API によりエラー・メッセージが表示されます。

---

最も一般的なプロシージャは、次の 3 つです。

**WF\_PURGE.ITEMS:** 完了した項目、そのプロセスおよびそれらで送信された通知に関連したすべてのランタイム・データを削除します。

**WF\_PURGE.ACTIVITIES:** どの項目でも使用されなくなったアクティビティの廃止バージョンを削除します。

**WF\_PURGE.TOTAL:** 項目データとアクティビティ・データの両方を削除します。

他の補助ルーチンでは、特定の表やクラスのデータのみ削除されるため、全面的な削除が望ましくない場合に使用できます。

PURGE API には、具体的に次のものがあります。

- 8-116 ページ [「Items」](#)
- 8-117 ページ [「Activities」](#)
- 8-118 ページ [「Notifications」](#)
- 8-119 ページ [「Total」](#)
- 8-120 ページ [「TotalPERM」](#)
- 8-121 ページ [「AdHocDirectory」](#)

---

**注意：** Oracle Applications embedded Workflow を使用している場合は、「ワークフローの不要ランタイム・データのパージ」コンカレント・プログラムを使用して、不要な項目タイプのランタイム・ステータス情報を削除することもできます。8-122 ページの「[ワークフローの不要ランタイム・データのパージ](#)」コンカレント・プログラムを参照してください。

---

---

**注意：** また、Oracle9i リリース 2 で利用可能なスタンドアロン版の Oracle Workflow を使用している場合は、Oracle Enterprise Manager から利用可能なスタンドアロン Oracle Workflow Manager コンポーネントを使用して、ワークフローのパージ・データベース・ジョブを管理できます。詳細は、Oracle Workflow Manager のオンライン・ヘルプを参照してください。

---

**関連項目：**

7-3 ページ「[関数アクティビティがコールする PL/SQL プロシージャの標準 API](#)」

C-7 ページ「[パフォーマンス改善のためのパージ](#)」

## Items

### 構文

```
procedure Items
  (itemtype in varchar2 default null,
   itemkey in varchar2 default null,
   enddate in date default sysdate,
   docommit in boolean default TRUE);
```

### 説明

プロセスのステータスおよび通知情報など、指定された項目タイプまたは項目キー（あるいはその両方）、および終了日付に関するすべての項目を削除します。具体的には、表 WF\_NOTIFICATIONS、WF\_ITEM\_ACTIVITY\_STATUSES、WF\_ITEM\_ATTRIBUTE\_VALUES および WF\_ITEMS から削除します。

### 引数（入力）

<b>itemtype</b>	削除する項目タイプ。すべての項目タイプを削除するには、この引数を NULL にします。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。NULL の場合、プロシージャは指定された項目タイプの項目をすべて削除します。
<b>enddate</b>	この日付までのデータを削除するように指定する日付。
<b>docommit</b>	削除中にデータをコミットするかどうかを示す TRUE または FALSE を指定します。ロールバック・セグメント数を減らしてパフォーマンスを改善するために、Items() で削除中にデータをコミットする場合は、TRUE を指定します。自動コミットを実行しない場合は、FALSE を指定します。デフォルト値は TRUE です。



# Activities

## 構文

```
procedure Activities
(
  itemtype in varchar2 default null,
  name in varchar2 default null,
  enddate in date default sysdate);
```

## 説明

表 WF\_ACTIVITY\_ATTR\_VALUES、WF\_ACTIVITY\_TRANSITIONS、WF\_PROCESS\_ACTIVITIES、WF\_ACTIVITY\_ATTRIBUTES\_TL、WF\_ACTIVITY\_ATTRIBUTES、WF\_ACTIVITIES\_TL および WF\_ACTIVITIES から、指定された項目タイプに関連付けられ、指定された終了日以前の END\_DATE を持ち、既存の項目でプロセスまたはアクティビティとして参照されない「適格」アクティビティの旧バージョンを削除します。

**注意：** 廃止項目の参照があるために廃止アクティビティが削除されなくなることを防ぐため、Activities( ) をコールする前に Items( ) をコールする必要があります。

## 引数（入力）

itemtype	削除するアクティビティに関連付けられている項目タイプ。すべての項目タイプのアクティビティを削除するには、この引数を NULL にします。
name	削除するアクティビティの内部名。指定した項目タイプのアクティビティをすべて削除するには、この引数を NULL にします。
enddate	この日付までのデータを削除するように指定する日付。

## Notifications

### 構文

```
procedure Notifications
  (itemtype in varchar2 default null,
   enddate in date default sysdate);
```

### 説明

表 WF\_NOTIFICATION\_ATTRIBUTES および WF\_NOTIFICATIONS から、指定した項目タイプに関連し、指定された終了日以前の END\_DATE を持ち、既存項目で参照されない古い適格通知を削除します。

---

---

**注意：** 廃止項目の参照があるために廃止通知が削除されなくなることを防ぐため、Notifications( ) をコールする前に Items( ) をコールする必要があります。

---

---

### 引数（入力）

itemtype	削除する通知に関連付けられている項目タイプ。すべての項目タイプの通知を削除するには、この引数を NULL にします。
enddate	この日付までのデータを削除するように指定する日付。

## Total

### 構文

```
procedure Total
(
  itemtype in varchar2 default null,
  itemkey in varchar2 default null,
  enddate in date default sysdate,
  docommit in boolean default TRUE);
```

### 説明

不要になった実行時の項目タイプとアクティビティ・データをすべて削除します。具体的には、指定された項目タイプに割り当てられており、指定された終了日付以前の **END\_DATE** が定義されているものを削除します。また、**AdHocDirectory()** がコールされ、使用されなくなったアドホック・ロールおよびユーザーも削除されます。8-121 ページの「**AdHocDirectory**」を参照してください。

**Total()** では、すべてのアクティビティおよびアドホック・ロール情報が削除されるため、パフォーマンス上は **Items()** より高コストです。特定の項目キーを削除する場合は、**Items()** を使用します。**Total()** は、低アクティビティ期間中の日常的な保守の一部として使用します。8-116 ページの「**Items**」を参照してください。

### 引数（入力）

<b>itemtype</b>	削除する不要なランタイム・データに関連した項目タイプ。すべての項目タイプの不要なランタイム・データを削除するには、この引数を NULL にします。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。NULL の場合、プロシージャは指定された項目タイプの項目をすべて削除します。
<b>enddate</b>	この日付までのデータを削除するように指定する日付。
<b>docommit</b>	削除中にデータをコミットするかどうかを示す TRUE または FALSE を指定します。ロールバック・セグメント数を減らしてパフォーマンスを改善するために、 <b>Total()</b> で削除中にデータをコミットする場合は、TRUE を指定します。自動コミットを実行しない場合は、FALSE を指定します。デフォルト値は TRUE です。

## TotalPERM

### 構文

```
procedure TotalPERM
  (itemtype in varchar2 default null,
   itemkey in varchar2 default null,
   enddate in date default sysdate,
   docommit in boolean default TRUE);
```

### 説明

維持タイプが「PERM」（永続）のランタイム・データのうち、不要になったものをすべて削除します。具体的には、指定された項目タイプに割り当てられており、指定された終了日付以前の END\_DATE が定義されているデータを削除します。TotalPERM() は Total() に似ていますが、維持タイプが「PERM」の項目のみを削除します。8-119 ページの「[Total](#)」を参照してください。

### 引数（入力）

<b>itemtype</b>	削除する不要なランタイム・データに関連した項目タイプ。すべての項目タイプの不要なランタイム・データを削除するには、この引数を NULL にします。
<b>itemkey</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。NULL の場合、プロシージャは指定された項目タイプの項目をすべて削除します。
<b>enddate</b>	この日付までのデータを削除するように指定する日付。
<b>docommit</b>	削除中にデータをコミットするかどうかを示す TRUE または FALSE を指定します。ロールバック・セグメント数を減らしてパフォーマンスを改善するために、TotalPERM() で削除中にデータをコミットする場合は、TRUE を指定します。自動コミットを実行しない場合は、FALSE を指定します。デフォルト値は TRUE です。

## AdHocDirectory

### 構文

```
procedure AdHocDirectory  
    (end_date in date default sysdate);
```

### 説明

WF\_LOCAL\_\* 表で、指定した END\_DATE 以前の期日が定義されており、通知で参照されていないユーザーおよびロールをすべて削除します。

### 引数（入力）

<b>end_date</b>	削除の対象となる最終日付。
-----------------	---------------

## 「ワークフローの不要ランタイム・データのパージ」コンカレント・プログラム

Oracle Applications embedded Workflow を使用している場合は、「ワークフローの不要ランタイム・データのパージ」コンカレント・プログラムを発行して、不要な項目タイプのランタイム・ステータス情報を削除できます。このコンカレント・プログラムを発行するには、Oracle Applications の「Submit Requests」フォームを使用します。

---

**注意：** Oracle Applications embedded Workflow を使用し、Oracle Applications Manager を実装している場合は、Oracle Workflow Manager を使用して「ワークフローの不要ランタイム・データのパージ」コンカレント・プログラムを発行および管理することができます。詳細は、Oracle Applications Manager のオンライン・ヘルプを参照してください。

---

### ▶ ワークフローの不要ランタイム・データのパージ

1. Oracle Applications の「Submit Requests」フォームにナビゲートし、「ワークフローの不要ランタイム・データのパージ」コンカレント・プログラムを発行します。システム管理者は、Oracle Applications と Oracle Workflow をインストールして設定するときに、このコンカレント・プログラムを実行する職責の要求セキュリティ・グループに追加する必要があります。このコンカレント・プログラムの実行ファイル名は Oracle Workflow Purge Obsolete Data で、短縮名は FNDWFPR です。『Oracle Applications System Administrator's Guide』の「Overview of Concurrent Programs and Requests」を参照してください。
2. ワークフローの不要ランタイム・データのパージ・コンカレント・プログラムを、要求として発行します。『Oracle Applications User's Guide』の「Submitting a Request」を参照してください。
3. 「パラメータ」ウィンドウで次のパラメータの値を入力します。

#### 項目タイプ

削除する不要なランタイム・データに関連した項目タイプ。すべての項目タイプの不要なランタイム・データを削除するには、この引数を NULL にします。

#### 項目キー

アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。NULL の場合、プログラムは指定された項目タイプの項目をすべて削除します。

#### 持続日数

維持タイプが「TEMP」に設定されている場合に、削除するデータの最短保存日数。デフォルトは 0 です。

#### 維持タイプ

削除の維持タイプとして、「TEMP」（一時）または「PERM」（永続）を指定します。デフォルトは「TEMP」です。

4. 「OK」を選択して「パラメータ」ウィンドウを閉じます。

5. この要求の印刷オプションと実行オプションを変更してから、「発行」を選択して要求を発行します。

## Workflow ディレクトリ・サービス API

次の API は、ランタイム・フェーズでアプリケーション・プログラムまたはワークフロー関数でコールでき、ディレクトリ・サービス内の既存のユーザーとロールに関する情報の取出しや、新規のアドホック・ユーザーとロールの作成および管理を行うことが可能です。これらの API は、WF\_DIRECTORY という PL/SQL パッケージで定義されています。

- 8-126 ページ [「GetRoleUsers」](#)
- 8-127 ページ [「GetUserRoles」](#)
- 8-128 ページ [「GetRoleInfo」](#)
- 8-129 ページ [「GetRoleInfo2」](#)
- 8-130 ページ [「IsPerformer」](#)
- 8-131 ページ [「UserActive」](#)
- 8-132 ページ [「GetUserName」](#)
- 8-133 ページ [「GetRoleName」](#)
- 8-134 ページ [「GetRoleDisplayName」](#)
- 8-135 ページ [「SetAdHocUserStatus」](#)
- 8-136 ページ [「SetAdHocRoleStatus」](#)
- 8-136 ページ [「CreateAdHocUser」](#)
- 8-139 ページ [「CreateAdHocRole」](#)
- 8-141 ページ [「AddUsersToAdHocRole」](#)
- 8-142 ページ [「SetAdHocUserExpiration」](#)
- 8-143 ページ [「SetAdHocRoleExpiration」](#)
- 8-144 ページ [「SetAdHocUserAttr」](#)
- 8-146 ページ [「SetAdHocRoleAttr」](#)
- 8-147 ページ [「RemoveUsersFromAdHocRole」](#)



---

**注意：** OID 統合を実装する場合は、OID 以外のツールを使用してユーザーを管理しないでください。OID との統合後に OID 以外のツールを使用してユーザーを管理すると、ユーザー情報の不一致や予測できない結果が発生する可能性があるため、WF\_LOCAL\_USERS 表にアドホック・ユーザーを作成しないでください。したがって、OID 統合を実装する場合は、WF\_DIRECTORY パッケージの CreateAdHocUser()、SetAdHocUserStatus()、SetAdHocUserExpiration() または SetAdHocUserAttr() API を使用しないでください。

ただし、Workflow ロールは OID では管理されないため、アドホック・ロールは使用できます。

---

**関連項目：**

7-3 ページ「関数アクティビティがコールする PL/SQL プロシージャの標準 API」

## GetRoleUsers

### 構文

```
procedure GetRoleUsers  
    (role in varchar2,  
     users out UserTable);
```

### 説明

指定されたロールに対するユーザーの表を戻します。

### 引数（入力）

<b>role</b>	有効なロール名。
-------------	----------

## GetUserRoles

### 構文

```
procedure GetUserRoles  
  (user in varchar2,  
   roles out RoleTable);
```

### 説明

指定されたユーザーに割り当てられているロールの表を戻します。

### 引数（入力）

<b>user</b>	有効なユーザー名。
-------------	-----------

## GetRoleInfo

### 構文

```
procedure GetRoleInfo
  (Role in varchar2,
   Display_Name out varchar2,
   Email_Address out varchar2,
   Notification_Preference out varchar2, Language out varchar2,
   Territory out varchar2);
```

### 説明

ロールに関する次の情報を戻します。

- 表示名
- 電子メール・アドレス
- 通知環境設定（QUERY、MAILTEXT、MAILHTML、MAILATTH、SUMMARY）
- 言語
- 地域

### 引数（入力）

<b>role</b>	有効なロール名。
-------------	----------

## GetRoleInfo2

### 構文

```
procedure GetRoleInfo2
  (Role in varchar2,
   Role_Info_Tbl out wf_directory.wf_local_roles_tbl_type);
```

### 説明

SQL 表のロールについて、次の情報を戻します。

- 表示名
- 説明
- 通知環境設定 (QUERY、MAILTEXT、MAILHTML、MAILATTH、SUMMARY)
- 言語
- 地域
- 電子メール・アドレス
- ファックス
- ステータス
- 失効日

### 引数 (入力)

**role**                      有効なロール名。

## IsPerformer

### 構文

```
function IsPerformer  
  (user in varchar2,  
   role in varchar2);
```

### 説明

ユーザーがロールの実行者であるかどうかを識別する値 TRUE または FALSE を戻します。

### 引数（入力）

<b>user</b>	有効なユーザー名。
<b>role</b>	有効なロール名。

## UserActive

### 構文

```
function UserActive  
(username in varchar2)  
    return boolean;
```

### 説明

現在、ユーザーのステータスが「ACTIVE」で、ワークフローに参加できるかどうかを判別します。ユーザーのステータスが「ACTIVE」であれば TRUE、それ以外の場合は FALSE を返します。

### 引数（入力）

<b>username</b>	有効なユーザー名。
-----------------	-----------

## GetUserName

### 構文

```
procedure GetUserName  
    (p_orig_system in varchar2,  
     p_orig_system_id in varchar2,  
     p_name out varchar2,  
     p_display_name out varchar2);
```

### 説明

ワークフローの表示名およびユーザー名を、元のユーザーおよびロールのリポジトリのシステム情報で戻します。

引数（入力）

<b>p_orig_system</b>	元のリポジトリ表を識別するコード。
<b>p_orig_system_id</b>	元のリポジトリ表の行の ID。



## GetRoleName

### 構文

```
procedure GetRoleName
    (p_orig_system in varchar2,
     p_orig_system_id in varchar2,
     p_name out varchar2,
     p_display_name out varchar2);
```

### 説明

ワークフローの表示名およびロール名を、元のユーザーおよびロールのリポジトリのシステム情報で戻します。

### 引数（入力）

<b>p_orig_system</b>	元のリポジトリ表を識別するコード。
<b>p_orig_system_id</b>	元のリポジトリ表の行の ID。

## GetRoleDisplayName

### 構文

```
function GetRoleDisplayName
  (p_role_name in varchar2)
  return varchar2;
pragma restrict_references (GetRoleDisplayName, WNDS,
WNPS);
```

### 説明

ワークフローのロールの表示名を、ロールの内部名で戻します。

### 引数（入力）

<b>p_role_name</b>	ロールの内部名。
--------------------	----------

## SetAdHocUserStatus

### 構文

```
procedure SetAdHocUserStatus
  (user_name in varchar2,
   status in varchar2 default 'ACTIVE');
```

### 説明

アドホック・ユーザーのステータスを「ACTIVE」または「INACTIVE」に設定します。

---

---

**注意：** OID 統合を実装する場合は、OID 以外のツールを使用してユーザーを管理しないでください。OID との統合後に OID 以外のツールを使用してユーザーを管理すると、ユーザー情報の不一致や予測できない結果が発生する可能性があるため、SetAdHocUserStatus( ) API を使用して WF\_LOCAL\_USERS 表のユーザー情報を更新しないでください。

---

---

### 引数（入力）

<b>user_name</b>	ユーザーの内部名。
<b>status</b>	ユーザーに設定するステータス。「ACTIVE」または「INACTIVE」。NULL の場合は、「ACTIVE」に設定されます。

## SetAdHocRoleStatus

### 構文

```
procedure SetAdHocRoleStatus
  (role_name in varchar2,
   status in varchar2 default 'ACTIVE');
```

### 説明

アドホック・ロールのステータスを「ACTIVE」または「INACTIVE」に設定します。

### 引数（入力）

<b>role_name</b>	ロールの内部名。
<b>status</b>	ロールに設定するステータス。「ACTIVE」または「INACTIVE」。 NULL の場合は、「ACTIVE」に設定されます。

## CreateAdHocUser

### 構文

```
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);
```

### 説明

WF\_LOCAL\_USERS 表に値を作成し、実行時にユーザーを作成します。これは、アドホック・ユーザーと呼ばれます。

---

**注意：** OID 統合を実装する場合は、OID 以外のツールを使用してユーザーを管理しないでください。OID との統合後に OID 以外のツールを使用してユーザーを管理すると、ユーザー情報の不一致や予測できない結果が発生する可能性があるため、**CreateAdHocUser()** API を使用して WF\_LOCAL\_USERS 表に新しいユーザーを作成しないでください。

---

## 引数（入力）

<b>name</b>	ユーザーの内部名。この名前は、30 文字以内で、すべて大文字で入力する必要があります。このプロシージャは、入力された名前が WF_USERS に存在するかどうかをチェックし、存在する場合はエラーを返します。内部名を指定しない場合は、接頭辞「~WF_ADHOC-」の後に連番を付加した内部名が生成されます。
<b>display_name</b>	ユーザーの表示名。このプロシージャは、入力された表示名が WF_USERS に存在するかどうかをチェックし、存在する場合はエラーを返します。表示名を指定しない場合は、接頭辞「~WF_ADHOC-」の後に連番を付加した表示名が生成されます。
<b>language</b>	データベースの NLS_LANGUAGE 初期化パラメータの値。ユーザーの通知セッションに対して、デフォルトの言語依存動作を指定します。NULL の場合は、現行のセッションの言語設定が使用されます。
<b>territory</b>	データベースの NLS_TERRITORY 初期化パラメータの値。ユーザーの通知セッションで使用される地域依存の日付 / 数値書式のデフォルト値を指定します。NULL の場合は、現行のセッションの地域設定が使用されます。
<b>description</b>	ユーザーの説明（オプション）
<b>notification_preference</b>	ユーザーが通知を受信する方法を指定します。有効値は MAILTEXT、MAILHTML、MAILATTH、QUERY または SUMMARY です。NULL の場合は、MAILHTML に設定されます。
<b>email_address</b>	ユーザーの電子メール・アドレス（オプション）。
<b>fax</b>	ユーザーの FAX 番号（オプション）。
<b>status</b>	ワークフロー・プロセスに参加するユーザーの使用可能ステータス。有効値は ACTIVE、EXTLEAVE、INACTIVE および TMPLEAVE です。NULL の場合は、ACTIVE に設定されます。
<b>expiration_date</b>	ディレクトリ・サービスにおいて、ユーザーが有効でなくなる日付。NULL の場合は、デフォルトの失効日として sysdate が設定されます。

**関連項目：**

2-20 ページ [「手順 WF-3 Oracle Workflow のディレクトリ・サービスの設定」](#)

## CreateAdHocRole

### 構文

```
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);
```

### 説明

WF\_LOCAL\_ROLES 表に値を作成し、実行時にロールを作成します。これは、アドホック・ロールと呼ばれます。

### 引数（入力）

<b>role_name</b>	ロールの内部名。この名前は、30 文字以内で、すべて大文字で入力する必要があります。このプロシージャは、指定された名前が WF_ROLES に存在するかどうかをチェックし、存在する場合はエラーを返します。内部名を指定しない場合は、接頭辞「~WF_ADHOC-」の後に連番を付加した内部名が生成されます。
<b>role_display_name</b>	ロールの表示名。このプロシージャは、指定された表示名が WF_ROLES に存在するかどうかをチェックし、存在する場合はエラーを返します。表示名を指定しない場合は、接頭辞「~WF_ADHOC-」の後に連番を付加した表示名が生成されます。
<b>language</b>	データベースの NLS_LANGUAGE 初期化パラメータの値。ユーザーの通知セッションに対して、デフォルトの言語依存動作を指定します。NULL の場合は、現行のセッションの言語設定が使用されます。
<b>territory</b>	データベースの NLS_TERRITORY 初期化パラメータの値。ユーザーの通知セッションで使用される地域依存の日付 / 数値書式のデフォルト値を指定します。NULL の場合は、現行のセッションの地域設定が使用されます。

<b>role_description</b>	ロールの説明（オプション）。
<b>notification_preference</b>	ロールが通知を受信する方法を指定します。有効値は MAILTEXT、MAILHTML、MAILATTH、QUERY または SUMMARY です。NULL の場合は、MAILHTML に設定されます。
<b>role_users</b>	ロールに定義されているユーザー名を表します。複数のユーザー名を列記する場合は、カンマまたはスペースで区切ります。
<b>email_address</b>	各自の電子メール・システムで定義されているロールの電子メール・アドレス、またはメール配信リスト（オプション）。
<b>fax</b>	ロールの FAX 番号（オプション）。
<b>status</b>	ワークフロー・プロセスに参加するためのロールの使用可能ステータス。有効値は ACTIVE、EXTLEAVE、INACTIVE および TMPLEAVE です。NULL の場合は、ACTIVE に設定されます。
<b>expiration_date</b>	ディレクトリ・サービスにおいて、ロールが有効でなくなる日付。NULL の場合は、デフォルトの失効日として sysdate が設定されます。

**関連項目：**

2-20 ページ [「手順 WF-3 Oracle Workflow のディレクトリ・サービスの設定」](#)



## AddUsersToAdHocRole

### 構文

```
procedure AddUsersToAdHocRole
  (role_name in varchar2,
   role_users in varchar2);
```

### 説明

既存のアドホック・ロールにユーザーを追加します。

### 引数（入力）

<b>role_name</b>	アドホック・ロールの内部名。
<b>role_users</b>	複数のユーザーをスペースまたはカンマで区切って指定します。アドホック・ユーザーまたはアプリケーションに定義したユーザーを定義できます。Oracle Workflow ディレクトリ・サービスに定義されていなければなりません。

## SetAdHocUserExpiration

### 構文

```
procedure SetAdHocUserExpiration
  (user_name in varchar2,
   expiration_date in date default sysdate);
```

### 説明

アドホック・ユーザーの失効日を更新します。

---

---

**注意：** OID 統合を実装する場合は、OID 以外のツールを使用してユーザーを管理しないでください。OID との統合後に OID 以外のツールを使用してユーザーを管理すると、ユーザー情報の不一致や予測できない結果が発生する可能性があるため、SetAdHocUserExpiration( ) API を使用して WF\_LOCAL\_USERS 表のユーザー情報を更新しないでください。

---

---

### 引数（入力）

user_name	アドホック・ユーザーの内部名。
expiration_date	新しい失効日。NULL の場合は、デフォルトの失効日として sysdate が設定されます。

## SetAdHocRoleExpiration

### 構文

```
procedure SetAdHocRoleExpiration
  (role_name in varchar2,
   expiration_date in date default sysdate);
```

### 説明

アドホック・ロールの失効日を更新します。

### 引数（入力）

<b>role_name</b>	アドホック・ロールの内部名。
<b>expiration_date</b>	新しい失効日。NULL の場合は、デフォルトの失効日として sysdate が設定されます。

## SetAdHocUserAttr

### 構文

```
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);
```

### 説明

アドホック・ユーザーの属性を更新します。

**注意：** OID 統合を実装する場合は、OID 以外のツールを使用してユーザーを管理しないでください。OID との統合後に OID 以外のツールを使用してユーザーを管理すると、ユーザー情報の不一致や予測できない結果が発生する可能性があるため、SetAdHocUserAttr() API を使用して WF\_LOCAL\_USERS 表のユーザー情報を更新しないでください。

### 引数（入力）

user_name	更新するアドホック・ユーザーの内部名。
display_name	アドホック・ユーザーの新しい表示名。NULL の場合、表示名は更新されません。
notification_preference	新しい通知環境を表します。有効値は MAILTEXT、MAILHTML、MAILATTH、QUERY または SUMMARY です。NULL の場合、通知環境は更新されません。
language	アドホック・ユーザーに対する、データベースの NLS_LANGUAGE 初期化パラメータの新しい値。NULL の場合、言語設定は更新されません。
territory	アドホック・ユーザーに対する、データベースの NLS_TERRITORY 初期化パラメータの新しい値。NULL の場合、地域設定は更新されません。
email_address	アドホック・ユーザーの新しい電子メール・アドレス。NULL の場合、電子メール・アドレスは更新されません。

**fax**

アドホック・ユーザーの新しい Fax 番号。NULL の場合、Fax 番号は更新されません。

## SetAdHocRoleAttr

### 構文

```
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);
```

### 説明

アドホック・ロールの属性を更新します。

### 引数（入力）

<b>role_name</b>	更新するアドホック・ロールの内部名。
<b>display_name</b>	アドホック・ロールの新しい表示名。NULL の場合、表示名は更新されません。
<b>notification_preference</b>	新しい通知環境を表します。有効値は MAILTEXT、MAILHTML、QUERY または SUMMARY です。NULL の場合、通知環境は更新されません。
<b>language</b>	アドホック・ロールに対する、データベースの NLS_LANGUAGE 初期化パラメータの新しい値。NULL の場合、言語設定は更新されません。
<b>territory</b>	アドホック・ロールに対する、データベースの NLS_TERRITORY 初期化パラメータの新しい値。NULL の場合、地域設定は更新されません。
<b>email_address</b>	アドホック・ロールの新しい電子メール・アドレス。NULL の場合、電子メール・アドレスは更新されません。
<b>fax</b>	アドホック・ロールの新しい Fax 番号。NULL の場合、Fax 番号は更新されません。

## RemoveUsersFromAdHocRole

### 構文

```
procedure RemoveUsersFromAdHocRole
  (role_name in varchar2,
   role_users in varchar2 default null);
```

### 説明

既存のアドホック・ロールからユーザーを削除します。

### 引数（入力）

<b>role_name</b>	アドホック・ロールの内部名。
<b>role_users</b>	アドホック・ロールから削除するユーザーのリスト。複数のユーザーを指定するときは、カンマまたはスペースで区切ります。NULL の場合は、ロールからすべてのユーザーが削除されます。

## Workflow LDAP API

次の API をコールして、Workflow ディレクトリ・サービスのローカル・ユーザー情報を Oracle Internet Directory (OID) などの LDAP ディレクトリのユーザーに同期させます。これらの API は、WF\_LDAP という PL/SQL パッケージで定義されています。

- 8-149 ページ [「Synch\\_changes」](#)
- 8-150 ページ [「Synch\\_all」](#)
- 8-151 ページ [「Schedule\\_changes」](#)

### 関連項目：

2-29 ページ [「手順 WF-4 Workflow ディレクトリ・サービスと Oracle Internet Directory の同期」](#)



## Synch\_changes

### 構文

```
function synch_changes  
    return boolean;
```

### 説明

LDAP の変更ログ・レコードを問い合せて、前回の同期後に LDAP ディレクトリのユーザーに変更があるかどうかを確認します。作成、変更および削除などの変更がある場合、Synch\_changes() はユーザー属性情報を属性キャッシュに格納し、oracle.apps.wf.public.user.change イベントを呼び出して関係するユーザーに警告を送信します。この関数は、「グローバル・ワークフロー・プリファレンス」で指定されている LDAP ディレクトリに接続します。変更されたユーザーごとに 1 つのイベントが呼び出されます。

関数が正常に完了した場合は TRUE、例外が発生した場合は FALSE を戻します。

#### 関連項目：

2-29 ページ [「手順 WF-4 Workflow ディレクトリ・サービスと Oracle Internet Directory の同期」](#)

2-13 ページ [「グローバル・ワークフロー・プリファレンスの設定」](#)

14-18 ページ [「User Entry Has Changed イベント」](#)

## Synch\_all

### 構文

```
function synch_all  
    return boolean;
```

### 説明

LDAP ディレクトリからすべてのユーザーを取得し、ユーザー属性情報を属性キャッシュに格納し、`oracle.apps.wf.public.user.change` イベントを呼び出して関係するユーザーに警告を送信します。この関数は、「グローバル・ワークフロー・プリファレンス」で指定されている LDAP ディレクトリに接続します。ユーザーごとに 1 つのイベントが呼び出されます。

`Synch_all()` は、LDAP ディレクトリに格納されているすべてのユーザーの情報を取得します。このため、設定時またはリカバリやクリーン・アップが必要なときに、一度だけ使用してください。`Synch_all()` を実行した後で、`Synch_changes()` または `Schedule_changes()` を使用すると、変更されたユーザー情報のみを取得できます。

関数が正常に完了した場合は `TRUE`、例外が発生した場合は `FALSE` を返します。

OID 統合を実装する場合は、Workflow ディレクトリ・サービスと Oracle Internet Directory の同期を開始するために、`Synch_all()` を実行します。

#### 関連項目：

[2-29 ページ「手順 WF-4 Workflow ディレクトリ・サービスと Oracle Internet Directory の同期」](#)

[2-13 ページ「グローバル・ワークフロー・プリファレンスの設定」](#)

[14-18 ページ「User Entry Has Changed イベント」](#)

[8-149 ページ「Synch\\_changes」](#)

[8-151 ページ「Schedule\\_changes」](#)

## Schedule\_changes

### 構文

```
procedure schedule_changes
(l_day in pls_integer default 0,
 l_hour in pls_integer default 0,
 l_minute in pls_integer default 10);
```

### 説明

LDAP ディレクトリのユーザーの変更をチェックし、変更に関係するユーザーに警告を送信するには、指定した間隔で `Synch_changes()` API を実行します。デフォルトの間隔は 10 秒です。`Schedule_changes()` は、DBMS\_JOB ユーティリティを使用してデータベース・ジョブを送信し、`Synch_changes()` を実行します。

OID 統合を実装する場合は、`Schedule_changes()` を実行して、Workflow ディレクトリ・サービスと Oracle Internet Directory の同期を管理します。

### 引数（入力）

<b>l_day</b>	Synch_changes() API の実行間隔の日数。デフォルト値はゼロです。
<b>l_hour</b>	Synch_changes() API の実行間隔の時間数。デフォルト値はゼロです。
<b>l_minute</b>	Synch_changes() API の実行間隔の分数。デフォルト値は 10 です。

#### 関連項目：

8-149 ページ [「Synch\\_changes」](#)

2-29 ページ [「手順 WF-4 Workflow ディレクトリ・サービスと Oracle Internet Directory の同期」](#)

## Workflow Preference API

次の API をコールし、ユーザー設定項目の情報を取得します。この API は、WF\_PREF という PL/SQL パッケージに定義されています。

### get\_pref

#### 構文

```
function get_pref  
  (p_user_name in varchar2,  
   p_preference_name in varchar2)  
  return varchar2;
```

#### 説明

指定したユーザーの特定の設定項目の値を取得します。

#### 引数（入力）

<b>p_user_name</b>	ユーザーの内部名。グローバルな設定項目値を取得するには、ユーザーに WF_DEFAULT と指定します。
<b>p_preference_name</b>	値を取得するユーザー設定項目の名前。次の設定項目名を取得できます。  LANGUAGE  TERRITORY  MAILTYPE  DMHOME  DATEFORMAT

## Workflow Monitor API

次の API をコールして、アクセス・キーを取得するか、ワークフロー・モニターの各種ページにアクセスする完全な URL を生成できます。ワークフロー・モニター API はすべて、WF\_MONITOR という PL/SQL パッケージに定義されています。

- 8-154 ページ [「GetAccessKey」](#)
- 8-155 ページ [「GetDiagramURL」](#)
- 8-157 ページ [「GetEnvelopeURL」](#)
- 8-159 ページ [「GetAdvancedEnvelopeURL」](#)

---

**注意：** Oracle Workflow の旧バージョンの GetURL API は、GetEnvelopeURL API と GetDiagramURL API に変更になりました。旧 GetURL API の機能は、新規の GetDiagramURL API と直接関連していません。Oracle Workflow の現行バージョンでも GetURL API は認識されますが、将来的にも、該当する場合はこの 2 つの新規 API のみを使用するようにしてください。

---

## GetAccessKey

### 構文

```
function GetAccessKey  
  (x_item_type varchar2,  
   x_item_key varchar2,  
   x_admin_mode varchar2)  
  return varchar2;
```

### 説明

ワークフロー・モニターへのアクセスを制御するアクセス・キー・パスワードを取得します。各プロセス・インスタンスには、ワークフロー・モニターを **ADMIN** モードまたは **USER** モードで実行できるように、別個のアクセス・キーがあります。

### 引数（入力）

<b>x_item_type</b>	有効な項目タイプ。
<b>x_item_key</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、レポートするプロセスが識別されます。
<b>x_admin_mode</b>	値 YES または NO。YES は、関数に対して、モニターを <b>ADMIN</b> モードで実行するアクセス・キー・パスワードを取り出すように指示します。NO の場合は、 <b>USER</b> モードでモニターを実行するアクセス・キー・パスワードが取り出されます。

## GetDiagramURL

### 構文

```
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;
```

### 説明

アプリケーションからコールし、添付されたアクセス・キー・パスワードを使用してワークフロー・モニターへのアクセスを可能にする URL を戻すことができます。この URL では、ADMIN または USER モードで動作しているワークフロー・モニターに、ワークフロー・プロセスの特定インスタンスのダイアグラムが表示されます。

WF\_MONITOR.GetDiagramURL() 関数からは、次のような URL が戻されます。

```
<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> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。2-12 ページの「[手順 WF-2 グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

wf\_monitor.html は、プロセス・インスタンスのワークフロー・モニター・ダイアグラムを生成する PL/SQL パッケージのプロシージャ名を表します。

wf\_monitor.html プロシージャには、4 つの引数が必要です。<item\_type> と <item\_key> は、特定のプロセス・インスタンスを識別する項目タイプと項目キーの内部名を表します。<YES or NO> が YES であれば、モニターは ADMIN モードで実行され、NO であれば USER モードで実行されます。<access\_key> は、モニターが ADMIN モードと USER モードのうちどちらで実行されているかを判別するアクセス・キー・パスワードです。

## 引数（入力）

<b>x_agent</b>	Web サーバー上の Oracle Workflow または Oracle Self-Service Web Applications に定義されている Web エージェントのベース文字列。Web エージェントのベース文字列は、WF_RESOURCES 表に保存する必要があります。次のように指定します。 <code>http://&lt;server.com:portID&gt;/&lt;PLSQL_agent_path&gt;</code>  この関数をコールする場合、アプリケーションで最初に WF_CORE.TRANSLATE() をコールして、WF_RESOURCES 表の WF_WEB_AGENT トークンから Web エージェントの文字列を取り出す必要があります。2-12 ページの「 <a href="#">手順 WF-2 グローバル・ワークフロー・プリファレンスの設定</a> 」を参照してください。
<b>x_item_type</b>	有効な項目タイプ。
<b>x_item_key</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、レポートするプロセスが識別されます。
<b>x_admin_mode</b>	値 YES または NO。YES は、関数に対して、モニターを ADMIN モードで実行するアクセス・キー・パスワードを取り出すように指示します。NO の場合は、USER モードでモニターを実行するアクセス・キー・パスワードが取り出されます。

## 例

次の例は、GetDiagramUrl をコールする方法を示しています。この例では、項目タイプ WFDEMO と項目キー 10022 で識別されるプロセス・インスタンスについて、ワークフロー・モニター・ダイアグラムを USER モードで表示する URL が戻されます。

```
URL := WF_MONITOR.GetDiagramURL  
      (WF_CORE.Translate('WF_WEB_AGENT'),  
       'WFDEMO',  
       '10022',  
       'NO');
```

### 関連項目：

8-113 ページ「[TRANSLATE](#)」



## GetEnvelopeURL

### 構文

```
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;
```

### 説明

アプリケーションからコールし、添付されたアクセス・キー・パスワードを使用してワークフロー・モニターの通知リストへのアクセスを可能にする URL を戻すことができます。この URL では、ワークフロー・プロセスの特定インスタンスの通知リストがワークフロー・モニターに表示されます。

WF\_MONITOR.GetEnvelopeURL() 関数からは、次のような URL が戻されます。

```
<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> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。2-12 ページの「[手順 WF-2 グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

wf\_monitor.envelope は、プロセス・インスタンスのワークフロー・モニター通知リストを生成する PL/SQL パッケージのプロシージャ名を表します。

### 引数（入力）

<b>x_agent</b>	<p>Web サーバー上の Oracle Workflow または Oracle Self-Service Web Applications に定義されている Web エージェントのベース文字列。Web エージェントのベース文字列は、WF_RESOURCES 表に保存する必要があります。次のように指定します。</p> <pre>http://&lt;server.com:portID&gt;/&lt;PLSQL_agent_path&gt;</pre> <p>この関数をコールする場合、アプリケーションで最初に WF_CORE.TRANSLATE() をコールして、WF_RESOURCES 表の WF_WEB_AGENT トークンから Web エージェントの文字列を取り出す必要があります。2-12 ページの「<a href="#">手順 WF-2 グローバル・ワークフロー・プリファレンスの設定</a>」を参照してください。</p>
<b>x_item_type</b>	有効な項目タイプ。

<b>x_item_key</b>	アプリケーション・オブジェクトの主キーから生成される文字列。 この文字列により、項目タイプの項目が一意に識別されます。項目 タイプと項目キーにより、レポートするプロセスが識別されます。
<b>x_admin_mode</b>	値 YES または NO。YES は、関数に対して、モニターを ADMIN モードで実行するアクセス・キー・パスワードを取り出すように指 示します。NO の場合は、USER モードでモニターを実行するアク セス・キー・パスワードが取り出されます。

**関連項目：**

8-113 ページ [「TRANSLATE」](#)

## GetAdvancedEnvelopeURL

### 構文

```
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;
```

### 説明

アプリケーションからコールし、添付されたアクセス・キー・パスワードを使用してワークフロー・モニターのアクティビティ・リストを表示する URL を戻すことができます。この URL では、ワークフロー・プロセスの特定インスタンスのアクティビティ・リストがワークフロー・モニターに表示されます。アクティビティ・リストでは、プロセス・インスタンスのアクティビティ・リストを表示するときに、強力なフィルタ・オプションを使用できます。

x\_options 引数が NULL の場合、WF\_MONITOR.GetAdvancedEnvelopeURL() 関数からは、次のような URL が戻されます。

### 例

```
<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> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。2-12 ページの「[手順 WF-2 グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

wf\_monitor.envelope は、プロセス・インスタンスのワークフロー・モニター通知リストを生成する PL/SQL パッケージのプロシージャ名を表します。

## 引数（入力）

<b>x_agent</b>	Web サーバー上の Oracle Workflow または Oracle Self-Service Web Applications に定義されている Web エージェントのベース文字列。Web エージェントのベース文字列は、WF_RESOURCES 表に保存する必要があります。次のように指定します。  <code>http://&lt;server.com:portID&gt;/&lt;PLSQL_agent_path&gt;</code>  この関数をコールする場合、アプリケーションで最初に WF_CORE.TRANSLATE() をコールして、WF_RESOURCES 表の WF_WEB_AGENT トークンから Web エージェントの文字列を取り出す必要があります。2-12 ページの「 <a href="#">手順 WF-2 グローバル・ワークフロー・プリファレンスの設定</a> 」を参照してください。
<b>x_item_type</b>	有効な項目タイプ。
<b>x_item_key</b>	アプリケーション・オブジェクトの主キーから生成される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、レポートするプロセスが識別されます。
<b>x_admin_mode</b>	値 YES または NO。YES は、関数に対して、モニターを ADMIN モードで実行するアクセス・キー・パスワードを取り出すように指示します。NO の場合は、USER モードでモニターを実行するアクセス・キー・パスワードが取り出されます。
<b>x_options</b>	アクティビティ・リストの URL を返すときに、すべてのフィルタ・オプションを選択する場合は、「すべて」を指定します。この引数が NULL の場合は、アクティビティ・リストの URL を返すときに、すべてのフィルタ・オプションを外します。この引数を使用して、任意のフィルタ・オプションを追加できます。デフォルトは NULL です。

### 関連項目：

8-113 ページ「[TRANSLATE](#)」

# Oracle Workflow のビュー

ワークフロー・データへのアクセスには、次のパブリック・ビューを使用できます。Oracle Applications embedded Workflow を使用している場合、これらのビューは APPS アカウントにインストールされています。Oracle Workflow のスタンドアロン版を使用している場合、これらのビューは Oracle Workflow サーバーと同じアカウントにインストールされています。

- 8-161 ページ [「WF\\_ITEM\\_ACTIVITY\\_STATUSES\\_V」](#)
- 8-163 ページ [「WF\\_NOTIFICATION\\_ATTR\\_RESP\\_V」](#)
- 8-164 ページ [「WF\\_RUNNABLE\\_PROCESSES\\_V」](#)
- 8-165 ページ [「WF\\_ITEMS\\_V」](#)

**注意：** これらのデータベース・ビューはパブリックなので、カスタム・データ要件を満たすために利用できます。ただし、これらのビューに対する一部の権限は PUBLIC には付与されていません。

## WF\_ITEM\_ACTIVITY\_STATUSES\_V

このビューには、ワークフロー・プロセスに関して正規化されていない情報と、そのアクティビティのステータスが含まれています。このビューを使用して、特定の項目やプロセスのステータスに関するユーザー定義の問合せとレポートを作成します。ビューの列の説明は、次のとおりです。

ビュー名	NULL?	型
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

このビューには、通知グループの「応答」メッセージ属性に関する情報が含まれています。ユーザー定義の投票アクティビティを作成するには、このビューを使用して、通知グループのユーザーからの応答を集計する関数を作成します。4-58 ページの「投票アクティビティ」を参照してください。

ビューの列の説明は、次のとおりです。

ビュー名	NULL?	型
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

このビューには、ACTIVITIES 表内の実行可能な全ワークフロー・プロセスのリストが含まれています。

ビューの列の説明は、次のとおりです。

ビュー名	NULL?	型
-----	-----	-----
ITEM_TYPE	NOT NULL	VARCHAR2 (8)
PROCESS_NAME	NOT NULL	VARCHAR2 (30)
DISPLAY_NAME	NOT NULL	VARCHAR2 (80)



## WF\_ITEMS\_V

このビューは、WF\_ITEMS 表の選択のみが可能なバージョンです。

ビューの列の説明は、次のとおりです。

ビュー名	NULL?	型
-----	-----	----
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 API

Oracle Workflow のキュー API は、ランタイム・フェーズでアプリケーション・プログラムまたはワークフロー関数によってコールされ、ワークフローの **Advanced Queuing** 処理を行います。Oracle Workflow では、送信用と受信用のキューが作成されます。キューにあるデータ・パッケージは、イベントまたはメッセージと呼ばれます。

---

**注意：** この場合のイベントは、ビジネス・イベント・システムに関連付けられているビジネス・イベントとは異なります。また、この場合のメッセージは、通知アクティビティに関連付けられているメッセージとは異なります。

---

イベントは送信キューに入れられ、エージェントによって取り込まれたり、処理されます。エージェントは、データベース外にあるアプリケーションの場合もあります。同様に、エージェントがメッセージを受信キューに入れ、ワークフロー・エンジンがこれを取り込んだり、処理する場合もあります。送信用と受信用のキューによって、ワークフロー・プロセスへの外部アクティビティの統合が容易になります。

---

**注意：** バックグラウンド・エンジンは、これとは異なる遅延キューを使用します。

---

Oracle Workflow のキュー API はすべて、WF\_QUEUE という PL/SQL パッケージに定義されています。これらのキュー API はアカウントに依存するため、同一の Oracle Workflow アカウントから実行する必要があります。

---

**注意：** これらの API を使用するには、Oracle Advanced Queuing の概念と用語をよく理解している必要があります。Oracle Advanced Queuing の詳細は、『Oracle9i アプリケーション開発者ガイド - アドバンスト・キューイング』を参照してください。

---

---

**注意：** 今後のリリースでは、このワークフローの Advanced Queuing 処理はビジネス・イベント・システム内に実装され、専用のキュー・ハンドラを使用してデキュー / エンキュー操作が行われる予定です。

---

### キュー API

- 8-170 ページ [「EnqueueInbound」](#)
- 8-172 ページ [「DequeueOutbound」](#)

- 8-175 ページ [「DequeueEventDetail」](#)
- 8-177 ページ [「PurgeEvent」](#)
- 8-178 ページ [「PurgeItemType」](#)
- 8-179 ページ [「ProcessInboundQueue」](#)
- 8-180 ページ [「GetMessageHandle」](#)
- 8-181 ページ [「DequeueException」](#)
- 8-182 ページ [「DeferredQueue」](#)
- 8-183 ページ [「InboundQueue」](#)
- 8-184 ページ [「OutboundQueue」](#)

**受信キュー用開発者 API**

次の API は、開発者が、WF\_QUEUE.EnqueueInbound() を使用せずに、内部スタックにメッセージを作成して、受信キューに書込みを行う場合に使用します。内部スタックは単なる格納領域で、スタックに作成した各メッセージは、最終的には受信キューに書き込む必要があります。

---

---

**注意：** パフォーマンスの向上のためには、スタックが大きくなりすぎないように、定期的に受信キューへの書込みを行ってください。

---

---

- 8-185 ページ [「ClearMsgStack」](#)
- 8-186 ページ [「CreateMsg」](#)
- 8-187 ページ [「WriteMsg」](#)
- 8-188 ページ [「SetMsgAttr」](#)
- 8-189 ページ [「SetMsgResult」](#)

**ペイロード構造**

Oracle Workflow Queue はすべて、system.wf\_payload\_t データ型を使用して、特定のメッセージに対するペイロードを定義します。ペイロードには、イベントに関する必須情報がすべて含まれています。次の表に、system.wf\_payload\_t の属性を示します。

**表 8-5**

属性名	データ型	説明
ITEMTYPE	VARCHAR2(8)	イベントの項目タイプ。

表 8-5 (続き)

属性名	データ型	説明
ITEMKEY	VARCHAR2(240)	イベントの項目キー。
ACTID	NUMBER	関数アクティビティのインスタンス ID。
FUNCTION_NAME	VARCHAR2(200)	実行する関数の名前。
PARAM_LIST	VARCHAR2(4000)	「値名 = 値」のペアのリスト。受信の場合、このペアは、項目属性および項目属性値として渡されます。送信では、このペアは、関数のすべての属性と属性値（アクティビティ属性）として渡されます。
RESULT	VARCHAR2(30)	オプションのアクティビティの完了結果。有効な値は、関数アクティビティの「結果タイプ」、またはエンジンの標準の結果の 1 つによって決まります。

**関連項目：**

7-3 ページ「[関数アクティビティがコールする PL/SQL プロシージャの標準 API](#)」

アドバンスト・キューイング：『Oracle9i アプリケーション開発者ガイド - アドバンスト・キューイング』

## EnqueueInbound

### 構文

```
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);
```

### 説明

送信イベントの結果を受信キューに入れます。送信イベントは、エージェントによって取り込まれる送信キューのメッセージで定義されています。

Oracle Workflow では、受信キューを処理するときに、外部関数アクティビティが指定の結果とともに完了としてマークされます。ただし、結果値は正常終了の場合にのみ有効です。`error_stack` パラメータに外部プログラム・エラーを指定すると、Oracle Workflow では結果値が上書きされ、外部関数アクティビティが **ERROR** ステータスで完了としてマークされます。また、対応するエラー・プロセスが項目タイプで定義されていれば、そのエラー・プロセスが開始されます。

### 引数（入力）

<b>itemtype</b>	イベントの項目タイプ。
<b>itemkey</b>	イベントの項目キー。項目キーは、アプリケーション・オブジェクトの主キーから生成される文字列です。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセス・インスタンスが識別されます。
<b>actid</b>	このイベントが関連付けられている関数アクティビティのインスタンス ID。
<b>result</b>	オブションのアクティビティの完了結果。有効な値は、関数アクティビティの「結果タイプ」によって決まります。
<b>attrlist</b>	項目属性および項目属性値として戻す「値名 = 値」のペアの長いリスト。各ペアは、「ATTR1=A^ATTR2=B^ATTR3=C」のように、カレット文字 (^) で区切られています。指定した値名が項目属性として存在しない場合、Oracle Workflow によって <code>varchar2</code> 型の項目属性が作成されます。

<b>correlation</b>	キューに入れられるメッセージのオプションの相関識別子を指定します。 <b>Oracle Advanced Queuing</b> では、特定の相関値に基づいてメッセージのキューを検索できます。 <b>NULL</b> の場合、ワークフロー・エンジンでは <b>Workflow</b> スキーマ名と項目タイプに基づいて相関識別子が作成されます。
<b>error_stack</b>	<b>Oracle Workflow</b> の内部エラー・スタックに置かれるオプションの外部プログラム・エラーを指定します。最大 200 文字以内でテキスト値を指定できます。

## DequeueOutbound

### 構文

```
procedure DequeueOutbound
  (dequeueumode 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);
```

### 説明

エージェントが取り込めるように送信キューからメッセージをデキューします。

---

---

**注意：** このプロシージャをループ内でコールする場合は、戻されるメッセージ・ハンドルを NULL に設定しないと、プロシージャで同じメッセージが再度デキューされます。これは望ましくない動作であり、無限ループの原因となることがあります。

---

---

### 引数（入力）

#### dequeueumode

番号 1、2 および 3 にそれぞれ対応する DBMS\_AQ.BROWSE、DBMS\_AQ.LOCKED または DBMS\_AQ.REMOVE という値で、デキューのロック動作を表します。DBMS\_AQ.BROWSE モードでは、メッセージのロックを取得せずにキューからメッセージを読み込みます。DBMS\_AQ.LOCKED モードでは、メッセージを読み込み、トランザクションが完了するまでメッセージの書き込みロックを取得します。DBMS\_AQ.REMOVE モードでは、メッセージを読み込んで削除します。

#### navigation

番号 1、2 にそれぞれ対応する DBMS\_AQ.FIRST\_MESSAGE または DBMS\_AQ.NEXT\_MESSAGE を指定し、取り出されるメッセージの場所を示します。DBMS\_AQ.FIRST\_MESSAGE という値では、取出し可能で関連基準と合致している最初のメッセージを取り出します。基本的には、キューの始まりが最初のメッセージとなります。DBMS\_AQ.NEXT\_MESSAGE という値では、取出し可能で関連基準と合致しており、キューを通して読み込める次のメッセージを取り出します。デフォルトは 1 です。



<b>correlation</b>	デキューされるメッセージのオプションの関連識別子を指定します。 Oracle Advanced Queuing では、特定の関連値に基づいてメッセージのキューを検索できます。「%」などの LIKE 比較演算子を使用して、識別子の文字列を指定できます。NULL の場合、ワークフロー・エンジンでは Workflow スキーマ名と項目タイプに基づいて関連識別子が作成されます。
<b>itemtype</b>	イベントの項目タイプ。
<b>message_handle</b>	デキューされる特定のイベントに対する、オプションのメッセージ・ハンドル ID を指定します。メッセージのハンドル ID を指定すると関連識別子は無視されます。

---

---

**注意：** キューから読み込むものがなくなると、タイムアウト出力は TRUE という値を戻します。

---

---

## 例

次の例は、送信キューをループして出力を表示するコードを示しています。

```
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

### 構文

```
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);
```

### 説明

指定のメッセージに対するすべてのイベントの詳細を、送信キューからデキューします。この API は DequeueOutbound と同様ですが、ペイロード・タイプを参照しません。そのかわりにペイロードの一部である itemkey、actid、function\_name および param\_list を出力します。

---

---

**注意：** このプロシージャをループ内でコールする場合は、戻されるメッセージ・ハンドルを NULL に設定しないと、プロシージャで同じメッセージが再度デキューされます。これは望ましくない動作であり、無限ループの原因となることがあります。

---

---

### 引数（入力）

**dequeueemode**

番号 1、2 および 3 にそれぞれ対応する DBMS\_AQ.BROWSE、DBMS\_AQ.LOCKED または DBMS\_AQ.REMOVE という値で、デキューのロック動作を表します。DBMS\_AQ.BROWSE モードでは、メッセージのロックを取得せずにキューからメッセージを読み込みます。DBMS\_AQ.LOCKED モードでは、メッセージを読み込み、トランザクションが完了するまでメッセージの書き込みロックを取得します。DBMS\_AQ.REMOVE モードでは、メッセージを読み込んでから更新か削除を行います。

<b>navigation</b>	番号 1、2 にそれぞれ対応する DBMS_AQ.FIRSTMESSAGE または DBMS_AQ.NEXTMESSAGE を指定し、取り出されるメッセージの場所を示します。DBMS_AQ.FIRSTMESSAGE という値では、取出し可能で関連基準と合致している最初のメッセージを取り出します。キューの始まりの位置もリセットします。DBMS_AQ.NEXTMESSAGE という値では、取出し可能で関連基準に合致している次のメッセージを取り出します。デフォルトは 1 です。
<b>correlation</b>	デキューされるメッセージのオプションの関連識別子を指定します。Oracle Advanced Queuing では、特定の関連値に基づいてメッセージのキューを検索できます。「%」などの LIKE 比較演算子を使用して、識別子の文字列を指定できます。NULL の場合、ワークフロー・エンジンでは Workflow スキーマ名と項目タイプに基づいて関連識別子が作成されます。
<b>acctname</b>	Oracle Workflow データベースのアカウント名。アカウント名が NULL の場合、デフォルトは USER という疑似列になります。
<b>itemtype</b>	関連を指定していない場合、デキューするメッセージに対するオプションの項目タイプを指定します。
<b>message_handle</b>	デキューされる特定のイベントに対する、オプションのメッセージ・ハンドル ID を指定します。メッセージのハンドル ID を指定すると関連識別子は無視されます。

---

---

**注意：** キューから読み込むものがなくなると、タイムアウト出力は TRUE という値を戻します。

---

---

## PurgeEvent

### 構文

```
procedure PurgeEvent  
    (queueName in varchar2,  
     message_handle in raw);
```

### 説明

特定のキューから、あるイベントを後の処理をせずに削除します。

### 引数（入力）

<b>queueName</b>	イベントが削除されるキューの名前。
<b>message_handle</b>	削除する特定のイベントに対するメッセージのハンドル ID。

## PurgeItemType

### 構文

```
procedure PurgeItemType  
  (queueName in varchar2,  
   itemType in varchar2 default null,  
   correlation in varchar2 default null);
```

### 説明

特定のキューから、指定した項目タイプに属するすべてのイベントを、後の処理をせずに削除します。

### 引数（入力）

<b>queueName</b>	イベントが削除されるキューの名前。
<b>itemType</b>	削除するイベントのオプションの項目タイプ。
<b>correlation</b>	削除されるメッセージのオプションの相関識別子を指定します。 <b>Oracle Advanced Queuing</b> では、特定の相関値に基づいてメッセージのキューを検索できます。「%」などの <b>LIKE</b> 比較演算子を使用して、識別子の文字列を指定できます。 <b>NULL</b> の場合、ワークフロー・エンジンでは <b>Workflow</b> スキーマ名と項目タイプに基づいて相関識別子が作成されます。

## ProcessInboundQueue

### 構文

```
procedure ProcessInboundQueue
  (itemtype in varchar2 default null,
   correlation in varchar2 default null);
```

### 説明

受信キューからすべてのメッセージを読み込み、各メッセージを完了イベントとして記録します。完了イベントの結果と、完了イベントの結果として更新される項目属性のリストは、受信キューの各メッセージによって指定されます。8-170 ページの「[EnqueueInbound](#)」を参照してください。

### 引数（入力）

<b>itemtype</b>	処理するイベントのオプションの項目タイプ。
<b>correlation</b>	特定の相関があるメッセージのみを処理する場合は、相関識別子を入力します。相関が NULL の場合、ワークフロー・エンジンでは <b>Workflow</b> スキーマ名と項目タイプに基づいて相関識別子が作成されます。

## GetMessageHandle

### 構文

```
function GetMessageHandle
(queueName in varchar2,
 itemType in varchar2,
 itemkey in varchar2,
 actid in number,
 correlation in varchar2 default null)
return raw;
```

### 説明

特定のメッセージのメッセージ・ハンドル ID を戻します。

### 引数（入力）

<b>queueName</b>	メッセージのハンドルが取り出されるキューの名前。
<b>itemtype</b>	メッセージの項目タイプ。
<b>itemkey</b>	メッセージの項目キー。項目キーは、アプリケーション・オブジェクトの主キーから生成される文字列です。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセス・インスタンスが識別されます。
<b>actid</b>	このメッセージが関連付けられている関数アクティビティのインスタンス ID。
<b>correlation</b>	メッセージのオプションの相関識別子を指定します。相関が NULL の場合、ワークフロー・エンジンでは <b>Workflow</b> スキーマ名と項目タイプに基づいて相関識別子が作成されます。



## DequeueException

### 構文

```
procedure DequeueException  
    (queueName in varchar2);
```

### 説明

例外キューからすべてのメッセージをデキューし、これらのメッセージとエラー・メッセージ「メッセージが期限切れです」をビジネス・イベント・システムの標準の WF\_ERROR キューに格納します。それらのメッセージが WF\_ERROR からデキューされると、「デフォルト・イベント・エラー・プロセス」を起動する事前定義済のサブスクリプションがトリガーされます。

### 引数（入力）

**queueName**                      デキューに使用できる例外キューの名前。

#### 関連項目：

6-23 ページ [「デフォルト・エラー・プロセス」](#)

## DeferredQueue

### 構文

```
function DeferredQueue return varchar2;
```

### 説明

遅延処理のためにバックグラウンド・エンジンで使用されている、キューとスキーマの名前を戻します。

## InboundQueue

### 構文

```
function InboundQueue return varchar2;
```

### 説明

受信キューとスキーマの名前を戻します。受信キューは、ワークフロー・エンジンに取り込まれるメッセージを含みます。

## OutboundQueue

### 構文

```
function OutboundQueue return varchar2;
```

### 説明

送信キューとスキーマの名前を戻します。送信キューは、外部エージェントに取り込まれるメッセージを含みます。

## ClearMsgStack

### 構文

```
procedure ClearMsgStack;
```

### 説明

内部スタックを消去します。「8-168 ページの「[受信キュー用開発者 API](#)」を参照してください。

## CreateMsg

### 構文

```
procedure CreateMsg  
  (itemtype in varchar2,  
   itemkey in varchar2,  
   actid in number);
```

### 説明

存在していない新規メッセージを内部スタックに作成します。8-168 ページの「[受信キュー用開発者 API](#)」を参照してください。

### 引数（入力）

<b>itemtype</b>	メッセージの項目タイプ。
<b>itemkey</b>	メッセージの項目キー。項目キーは、アプリケーション・オブジェクトの主キーから生成される文字列です。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセス・インスタンスが識別されます。
<b>actid</b>	このメッセージが関連付けられている関数アクティビティのインスタンス ID。

## WriteMsg

### 構文

```
procedure WriteMsg  
  (itemtype in varchar2,  
   itemkey in varchar2,  
   actid in number);
```

### 説明

内部スタックから受信キューにメッセージを書き込みます。8-168 ページの「[受信キュー用開発者 API](#)」を参照してください。

### 引数（入力）

<b>itemtype</b>	メッセージの項目タイプ。
<b>itemkey</b>	メッセージの項目キー。項目キーは、アプリケーション・オブジェクトの主キーから生成される文字列です。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。
<b>actid</b>	このメッセージが関連付けられている関数アクティビティのインスタンス ID。

# SetMsgAttr

## 構文

```
procedure SetMsgAttr
  (itemtype in varchar2,
   itemkey in varchar2,
   actid in number,
   attrName in varchar2,
   attrValue in varchar2);
```

## 説明

内部スタックのメッセージに項目属性を追加します。8-168 ページの「[受信キュー用開発者 API](#)」を参照してください。

## 引数（入力）

<b>itemtype</b>	メッセージの項目タイプ。
<b>itemkey</b>	メッセージの項目キー。項目キーは、アプリケーション・オブジェクトの主キーから生成される文字列です。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセス・インスタンスが識別されます。
<b>actid</b>	このメッセージが関連付けられている関数アクティビティのインスタンス ID。
<b>attrName</b>	メッセージに追加する項目属性の内部名。
<b>attrValue</b>	追加する項目属性の値。



## SetMsgResult

### 構文

```
procedure SetMsgResult
  (itemtype in varchar2,
   itemkey in varchar2,
   actid in number,
   result in varchar2);
```

### 説明

内部スタックに書き込まれたメッセージに結果を設定します。8-168 ページの「[受信キュー用開発者 API](#)」を参照してください。

### 引数（入力）

<b>itemtype</b>	メッセージの項目タイプ。
<b>itemkey</b>	メッセージの項目キー。項目キーは、アプリケーション・オブジェクトの主キーから生成される文字列です。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセス・インスタンスが識別されます。
<b>actid</b>	このメッセージが関連付けられている関数アクティビティのインスタンス ID。
<b>result</b>	メッセージの完了結果。有効な値は、アクティビティの「結果タイプ」によって決まります。



## 文書管理 API

---

**注意：** 文書管理機能は今後使用する目的で確保されています。Oracle Workflow 文書管理 API に関する説明は、参考のために記載しています。

---

次の文書管理 API は、URL を戻すためにユーザー・インタフェース (UI) のエージェントにコールされるか、または、サポートされている文書管理システムへの統合アクセスを可能にする JavaScript 関数によってコールされます。サポートされているすべての文書管理 (DM) システムは、文書にアクセスする URL インタフェースを内包します。

文書管理 API を使用すると、同一ネットワーク内にある他のベンダーの DM システム内で複数のインスタンスを通して文書にアクセスできるのみでなく、同一の DM システム内でも複数のインスタンスを通して文書にアクセスできます。

文書管理 API は、FND\_DOCUMENT\_MANAGEMENT と呼ばれる PL/SQL パッケージで定義されています。

- 8-192 ページ [「get\\_launch\\_document\\_url」](#)
- 8-193 ページ [「get\\_launch\\_attach\\_url」](#)
- 8-194 ページ [「get\\_open\\_dm\\_display\\_window」](#)
- 8-195 ページ [「get\\_open\\_dm\\_attach\\_window」](#)
- 8-196 ページ [「set\\_document\\_id\\_html」](#)

### 関連項目：

7-3 ページ [「関数アクティビティがコールする PL/SQL プロシージャの標準 API」](#)

## get\_launch\_document\_url

### 構文

```
procedure get_launch_document_url
  (username in varchar2,
   document_identifier in varchar2,
   display_icon in Boolean,
   launch_document_url out varchar2);
```

### 説明

新規のブラウザ・ウィンドウを起動するアンカー URL を戻し、そのウィンドウには指定された文書を示す DM 統合画面が表示されます。画面は上下 2 つのフレームに分かれています。上のフレームにはカスタマイズ可能な会社のロゴと、Oracle Workflow 統合文書管理機能のツールバーが表示されます。下のフレームには指定した文書が表示されます。

### 引数（入力）

<b>username</b>	文書管理システムにアクセスしているユーザーのユーザー名。
<b>document_identifier</b>	表示する文書の文書 ID。文書 ID は、文書タイプの項目属性の値として格納されています。GetItemAttrDocument API を使用して文書 ID を取り出すことができます。8-61 ページの「 <a href="#">GetItemAttrDocument</a> 」および 8-54 ページの「 <a href="#">SetItemAttrDocument</a> 」を参照してください。
<b>display_icon</b>	TRUE または FALSE。TRUE の場合は、URL とともに、ペーパー・クリップ・アイコンと変換されたプロンプト名が返されます。FALSE の場合は、URL のみが返されます。この引数を利用すれば、このプロシージャをフォームまたは HTML ベースの UI エージェントからコールすることができます。

## get\_launch\_attach\_url

### 構文

```
procedure get_launch_attach_url  
  (username in varchar2,  
   callback_function in varchar2,  
   display_icon in Boolean,  
   launch_attach_url out varchar2);
```

### 説明

新規のブラウザ・ウィンドウを起動するアンカー URL を返し、そのウィンドウには文書の添付に使用できる DM 統合画面が表示されます。画面は上下 2 つのフレームに分かれています。上のフレームにはカスタマイズ可能な会社のロゴと、Oracle Workflow 統合文書管理機能のツールバーが表示されます。下のフレームにはデフォルトの文書管理システムの検索画面が表示されます。

### 引数（入力）

<b>username</b>	文書管理システムにアクセスしているユーザーのユーザー名。
<b>callback_function</b>	ユーザーが添付する文書を選択した後にコールする URL。このコールバック関数は、set_document_id_html という API から戻された callback_url 構文です。
<b>display_icon</b>	TRUE または FALSE。TRUE の場合は、URL とともに、ペーパー・クリップ・アイコンと変換されたプロンプト名が返されます。FALSE の場合は、URL のみが返されます。この引数を利用すれば、このプロシージャをフォームまたは HTML ベースの UI エージェントからコールすることができます。

## get\_open\_dm\_display\_window

### 構文

```
procedure get_open_dm_display_window
```

### 説明

現行の UI から添付文書を表示する JavaScript 関数を戻します。JavaScript 関数は、ユーザーが添付文書上で実行できるすべての文書管理機能によって使用されます。各 DM 機能は現在の DM 統合画面に名前も表示するため、ドキュメントの Transport Window は現行のウィンドウに JavaScript 関数をコールバックできます。

## get\_open\_dm\_attach\_window

### 構文

```
procedure get_open_dm_attach_window
```

### 説明

ユーザーが現行の UI で文書を添付するときに文書転送ウィンドウを開く、JavaScript 関数を戻します。JavaScript 関数は、ユーザーが文書を添付するために実行できるすべての文書管理機能によって使用されます。各 DM 機能は現在の DM 統合画面に名前も表示するため、ドキュメントの Transport Window は現行のウィンドウに JavaScript 関数をコールバックできます。

## set\_document\_id\_html

### 構文

```
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);
```

### 説明

ユーザーが DM システムから文書を選択するときに実行される、コールバック URL を戻します。このプロシージャを使用して、文書管理の検索機能から選択した文書を、HTML ページの指定した宛先フィールドに設定します。宛先フィールドは、ユーザーが文書を添付するために DM 統合画面を起動するフィールドです。戻されたコールバック URL を引数として get\_launch\_attach\_url API に渡します。

### 引数（入力）

<b>frame_name</b>	現行の UI で対話する HTML フレームの名前。
<b>form_name</b>	現行の UI で対話する HTML フォームの名前。
<b>document_id_field_name</b>	<p>結果的な文書 ID を書き込む、現行の UI の HTML フィールドの名前。結果的な文書 ID は、ユーザーが文書管理の検索機能から選択した文書によって決まります。文書 ID は、次に示す値が連結されたものです。</p> <p>DM:&lt;node_id&gt;:&lt;document_id&gt;:&lt;version&gt;</p> <p>&lt;nodeid&gt; は、「文書管理ノード」 Web 画面で定義されている、文書管理システム・ノードに割り当てられたノード ID です。</p> <p>&lt;documentid&gt; は文書の文書 ID で、文書が保存されている文書管理システムによって割り当てられています。</p> <p>&lt;version&gt; は文書のバージョンです。バージョンを指定しなければ、最新バージョンとみなされます。</p>
<b>document_name_field_name</b>	<p>結果的な文書名を書き込む、現行の UI の HTML フィールドの名前。</p>



## Oracle Workflow 通知システムの概要

Oracle Workflow では、通知を送信してユーザーと通信します。通知に含まれるメッセージによって、ユーザーになんらかのアクションを要求したり、情報を提供できます。ユーザーは、通知アクティビティと、通知アクティビティで送信する通知メッセージをワークフロー・ビルダーで定義します。メッセージにオプション属性を指定すると、追加リソースの指定や応答の要求が可能です。

ユーザーは、HTML ブラウザで「通知」Web 画面を使用して、通知をオンラインで問合せできます。また、電子メール・アプリケーションで通知を受け取ることもできます。電子メールでの通知には、オプションの添付ファイルとして HTML のコンテンツやその他の文書を挿入できます。通知システムは、メッセージを配信し、受信応答を処理します。

## 通知モデル

ワークフロー・プロセス内の通知アクティビティには、設計時のメッセージとメッセージ属性のリストが使用されます。また、項目タイプ属性という実行時に指定される多数の値があり、この値からメッセージ属性の値が取り出されます。

ワークフロー・エンジンでは、ワークフロー・プロセスを移動して、各アクティビティを順番に評価します。通知アクティビティに到達すると、エンジンは通知システムの `Send()` API または `SendGroup()` API をコールして通知を送信します。

### 通知メッセージの送信

通知アクティビティに到達すると、ワークフロー・エンジンは `Send()` API または `SendGroup()` API をコールします。これらの API によって、次のことが行われます。

- 通知アクティビティの実行者ロールの有効性をチェックします。
- 実行者ロールの通知環境設定を識別します。
- メッセージのメッセージ属性を参照します。
  - メッセージ属性がソース「SEND」の場合、`Send()` API または `SendGroup()` API によって、メッセージ属性が参照する項目タイプ属性から値が取り出されます。プロシージャが項目タイプ属性を検出できない場合、使用可能なメッセージ属性のデフォルト値があれば、それを使用します。メッセージの件名や本文にソース「SEND」のメッセージ属性が含まれている場合があります。これは、通知の作成時に、`Send()` API または `SendGroup()` API トークンによって、各属性の現在値に置き換えられます。
  - メッセージにソース RESPOND のメッセージ属性が含まれている場合、`Send()` API または `SendGroup()` API によって、デフォルト値が割り当てられているかどうかチェックされます。その後、プロシージャによって、この RESPOND 属性を使用して通知のデフォルト応答セクションが作成されます。
- Workflow 通知表に関連情報を入れ、通知内容を作成します。

- 通知アクティビティのステータスを、応答が必要な場合は「NOTIFIED」に、応答が不要な場合は「COMPLETE」に更新します。

---

**注意：** 通知アクティビティで、情報を伝えることのみ（FYI）を目的として実行者にメッセージを送信する（応答メッセージ属性がメッセージに関連付けられていない）場合は、通知システムがメッセージを配信すると、通知アクティビティには即時に完了マークが設定されます。

---

---

**注意：** 投票アクティビティの場合、ステータスは「NOTIFIED」ではなく「WAITING」に更新されます。8-200 ページの「[投票アクティビティでの特別処理](#)」を参照してください。

---

通知の実行者ロールが、MAILTEXT、MAILHTML、MAILATTN または SUMMARY の通知環境設定を持つ場合、通知表では、対応する値のフラグが通知に立てられています。通知メーラーが通知表でこのフラグを検索して、電子メール版の通知を作成し、実行者に送信します。2-45 ページの「[手順 WF-9 通知メーラーの導入](#)」を参照してください。

ユーザーが、「通知」Web 画面から自分の通知を参照する場合、通知環境設定に関係なく、単にこれらのインタフェースから Workflow 通知表に問い合わせます。

通知受信者は、通知に対して次の 4 つの処理のうち 1 つを行えます。

- 通知に応答するか、応答が不要であれば通知を完了します。8-198 ページの「[通知応答の処理](#)」を参照してください。
- 他のロールに通知を転送します。8-199 ページの「[通知の転送](#)」を参照してください。
- 他のロールに通知の所有権を譲渡します。8-199 ページの「[通知の譲渡](#)」を参照してください。
- 通知を無視し、通知がタイムアウトになります。8-200 ページの「[タイムアウト通知の処理](#)」を参照してください。

## 通知応答の処理

受信者の応答後、「通知」Web 画面または通知メーラーは、応答値を通知応答属性に割り当てて、Respond() 通知 API をコールします。Respond() API は、最初に通知コールバック関数をコールし、通知アクティビティの通知後関数（存在する場合）を RESPOND モードで実行します。通知後関数は応答を解釈し、密結合の応答後処理を行います。通知後関数に例外が発生すると、応答は異常終了します。8-14 ページの「[通知後関数](#)」を参照してください。

例外が発生しなければ、Respond() は通知をクローズとしてマークし、通知コールバック関数を SET モードで再度コールして、対応する項目属性を RESPOND 通知属性値で更新します。通知メッセージが、メッセージのプロパティ画面の「結果」タブで指定されている応答を要求した場合、その応答値も通知アクティビティの結果として設定されます。

最後に、Respond() は WF\_ENGINE.CompleteActivity() をコールし、通知アクティビティが終了したため、該当する次のアクティビティへ移るようエンジンに通知します。

## 通知の転送

受信者が他のロールに通知を転送すると、「通知」 Web 画面は、通知システムの Forward() API をコールします。

---

---

**注意：** 通知システムでは、電子メールを介して転送された通知は追跡できません。最終的な応答者の電子メール・アドレスと応答に含まれる「応答」メッセージ属性値のみを記録します。

---

---

Forward() API はロールを検証した後、通知コールバック関数をコールし、通知アクティビティの通知後関数（存在する場合）を FORWARD モードで実行します。たとえば、通知後関数は通知の転送先のロールに通知の参照や応答を行うための権限があるかどうかを検証します。権限がない場合、通知後関数はエラーを戻し、転送操作は行われません。8-14 ページの「[通知後関数](#)」を参照してください。

その後、Forward() は追加されたコメントとともに通知を新規のロールに転送します。

---

---

**注意：** Forward() は、通知の所有者や元の宛先は更新しません。

---

---

## 通知の譲渡

受信者が他のロールに通知の所有権を譲渡する場合、「通知」 Web 画面は、通知システムの Transfer() API をコールします。

---

---

**注意：** 電子メール・アプリケーションから通知を参照する受信者は、通知を譲渡できません。通知を譲渡するには、「通知」 Web 画面を使用する必要があります。

---

---

Transfer() API はロールを検証した後、通知コールバック関数をコールし、通知アクティビティの通知後関数（存在する場合）を TRANSFER モードで実行します。たとえば、通知後関数は通知の譲渡先のロールに適切な権限があるかどうかを検証します。権限がない場合、通知後関数はエラーを戻し、譲渡操作は行われません。8-14 ページの「[通知後関数](#)」を参照してください。

その後、Transfer() は通知の所有権を新しいロールに割り当て、追加されたコメントとともに渡します。譲渡も通知のコメントに記録されることに注意してください。

## タイムアウト通知の処理

タイムアウト通知またはサブプロセス・アクティビティは、最初はバックグラウンド・エンジンによって検出されます。バックグラウンド・エンジンは、指定のタイムアウト値を持つアクティビティを定期的にチェックし、タイムアウトになったアクティビティを処理するように設定されています。アクティビティがタイムアウト値を持ち、現在の日時がタイムアウト値を超えている場合、バックグラウンド・エンジンはアクティビティのステータスを **TIMEOUT** とマークし、ワークフロー・エンジンをコールします。次に、ワークフロー・エンジンは、タイムアウト・トランジションが指すアクティビティを実行して処理を再開します。

## 投票アクティビティでの特別処理

投票アクティビティは、次のように定義された通知アクティビティです。

- ロールを拡張し、その結果、通知メッセージの個々のコピーが実行者ロールの各メンバーに送られます。
- 指定された結果付きのメッセージを持ち、受信者にリストから値を選択して返答するように要求します。
- **RUN** モードのロジックを含む、関連付けられた通知後関数を持ち、実行者メンバーから投げられた応答を処理して、ワークフロー・エンジンが通知アクティビティの結果と解釈するような、1 つの応答を作成します。4-58 ページの「[投票アクティビティ](#)」を参照してください。

通知システムによって投票アクティビティに関する通知が送られると、投票アクティビティのステータスは「**NOTIFIED**」とマークされます。応答をいくつか受け取ったが、投票基準を満たすほどでない場合、投票アクティビティのステータスは「**WAITING**」に更新されます。

通知メッセージを受け取った各ロール・メンバーは、通知を参照する 2 つの通知インタフェースのいずれかを使用して、通知に応答したり、通知を転送できます。また、「通知」Web 画面を使用すると、通知を譲渡できます。

通知ユーザー・インタフェースでは、実行者の行う処理に応じて **Respond()**、**Forward()** または **Transfer()** API がコールされます。続いて、各 API が、それぞれ、**RESPOND**、**FORWARD** または **TRANSFER** モードで通知後関数を実行する通知コールバック関数をコールします。通知システムは、**FORWARD** または **TRANSFER** モードで通知後関数の実行を終了すると、それぞれ転送または譲渡操作を行います。

通知システムが **RESPOND** モードで通知後関数の実行を終了すると、ワークフロー・エンジンは **RUN** モードで通知後関数を再実行します。すべての応答を受け取ると、投票集計ロジックを実行する関数を **RUN** モードでコールします。

また、投票アクティビティがループの途中で再実行するようにリセットされた場合や、タイムアウトになった場合、ワークフロー・エンジンは、それぞれ **CANCEL** モードまたは **TIMEOUT** モードで通知後関数を実行します。投票アクティビティの通知後関数の

TIMEOUT モードに対するロジックでは、タイムアウトまでに受け取った投票を集計する方法を識別します。

## 通知 API

次の API は、通知エージェントでコールされ、通知アクティビティの通知を管理できます。これらの API は、WF\_NOTIFICATION という PL/SQL パッケージに格納されています。

これらの通知 API の多くには、対応する Java メソッドが定義されています。これらの Java メソッドは、任意の Java プログラムからコールして、Oracle Workflow に取り込むことができます。次のリストは、通知 API が PL/SQL 関数またはプロシージャとして使用可能なのか、Java メソッドとして使用可能なのか、あるいはその両方なのかを表しています。8-6 ページの「[Oracle Workflow Java インタフェース](#)」を参照してください。

---

---

**注意：** Java では大文字 / 小文字が区別されます。Java のネーミング規則に従って、すべての Java メソッド名の先頭文字は小文字となります。

---

---

- 8-204 ページ [「Send」](#) : PL/SQL および Java
- 8-208 ページ [「SendGroup」](#) : PL/SQL
- 8-210 ページ [「Forward」](#) : PL/SQL および Java
- 8-212 ページ [「Transfer」](#) : PL/SQL および Java
- 8-214 ページ [「Cancel」](#) : PL/SQL および Java
- 8-215 ページ [「CancelGroup」](#) : PL/SQL
- 8-216 ページ [「Respond」](#) : PL/SQL および Java
- 8-218 ページ [「Responder」](#) : PL/SQL および Java
- 8-219 ページ [「VoteCount」](#) : PL/SQL および Java
- 8-220 ページ [「OpenNotificationsExist」](#) : PL/SQL および Java
- 8-221 ページ [「Close」](#) : PL/SQL および Java
- 8-222 ページ [「AddAttr」](#) : PL/SQL および Java
- 8-223 ページ [「SetAttribute」](#) : PL/SQL および Java
- 8-225 ページ [「GetAttrInfo」](#) : PL/SQL および Java
- 8-226 ページ [「GetInfo」](#) : PL/SQL および Java
- 8-227 ページ [「GetText」](#) : PL/SQL
- 8-228 ページ [「GetShortText」](#) : PL/SQL
- 8-229 ページ [「GetAttribute」](#) : PL/SQL および Java
- 8-231 ページ [「GetAttrDoc」](#) : PL/SQL および Java
- 8-232 ページ [「GetSubject」](#) : PL/SQL および Java

- 8-233 ページ [「GetBody」](#) : PL/SQL および Java
- 8-234 ページ [「GetShortBody」](#) : PL/SQL
- 8-235 ページ [「TestContext」](#) : PL/SQL
- 8-236 ページ [「AccessCheck」](#) : PL/SQL および Java
- 8-237 ページ [「WorkCount」](#) : PL/SQL および Java
- 8-238 ページ [「getNotifications」](#) -- Java
- 8-239 ページ [「getNotificationAttributes」](#) : Java
- 8-240 ページ [「WriteToClob」](#) : PL/SQL

## Send

### PL/SQL 構文

```
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 構文

```
public static BigDecimal send
(WFContext wCtx,
 String role,
 String messageType,
 String messageName,
 String dueDate,
 String callback,
 String context,
 String sendComment,
 BigDecimal priority)
```

### 説明

この関数は指定されたメッセージをロールに送信し、成功すると通知 ID を戻します。今後、この通知の参照には、この通知 ID を使用する必要があります。

メッセージにメッセージ属性がある場合、プロシージャはメッセージ属性表から属性の値を探すか、オプションのコールバック・インタフェース関数を使用して項目タイプ属性表から値を取得します。コールバック関数は、通知に対して応答があった場合にも使用できます。

---

---

**注意：** Oracle Workflow 通知システムと、電子メール・ベースまたは Web ベースの通知クライアントを使用している場合、Send プロシージャは、暗黙的に WF\_ENGINE.CB コールバック関数をコールします。ワークフロー・エンジンをコールしない独自のカスタム通知システムを使用している場合は、標準書式に従ってカスタム・コールバック関数を定義し、コールバックの引数として名前を指定する必要があります。8-205 ページの「[カスタム・コールバック関数](#)」を参照してください。

---

---



## 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>role</b>	通知アクティビティの実行者として割り当てられたロール名。
<b>msg_type または messageType</b>	メッセージに関連付けられている項目タイプ。
<b>msg_name または messageName</b>	メッセージの内部名。
<b>due_date または dueDate</b>	応答期日。このオプションの期日は、受信者に知らせるためのものであり、処理には影響しません。
<b>callback</b>	SEND および RESPOND のソース・メッセージ属性の通信に使用されるコールバック関数の名前。
<b>context</b>	コールバック関数に渡されるコンテキスト情報。
<b>send_comment または sendComment</b>	メッセージに表示するコメント。
<b>priority</b>	メッセージの優先度。#PRIORITY 通知アクティビティ属性から取り出されます。#PRIORITY が存在しない場合や、値が NULL の場合、ワークフロー・エンジンはメッセージのデフォルト優先度を使用します。

## カスタム・コールバック関数

デフォルト・コールバック関数は、WF\_NOTIFICATION API の処理によって、様々な時点でコールされます。独自のカスタム・コールバック関数を作成できますが、次の仕様に従ってください。

```
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);
```

## 引数（入力）

<b>command</b>	要求に応じて、GET、SET、COMPLETE、ERROR、TESTCTX、FORWARD、TRANSFER または RESPOND を指定します。属性値を取得するには GET を、属性値を設定するには SET を、応答が完了したことを示すには COMPLETE を、関連する通知アクティビティのステータスを ERROR にするには ERROR を、項目タイプのセクタ / コールバック関数をコールして現行のコンテキストをテストするには TESTCTX を、通知後関数を FORWARD モードで実行するには FORWARD を、通知後関数を TRANSFER モードで実行するには TRANSFER を、通知後関数を RESPOND モードで実行するには RESPOND を使用します。
<b>context</b>	SEND( ) または SendGroup( ) に渡されるコンテキスト。書式は <itemtype>:<itemkey>:<activityid> です。
<b>attr_name</b>	コマンドが GET または SET の場合、設定または取得する属性名。
<b>attr_type</b>	コマンドが SET または GET の場合、属性タイプ。
<b>text_value</b>	コマンドが SET の場合、テキスト属性値。コマンドが GET の場合、戻されるテキスト属性値。
<b>number_value</b>	コマンドが SET の場合、数値属性値。コマンドが GET の場合、戻される数値属性値。
<b>date_value</b>	コマンドが SET の場合、日付属性値。コマンドが GET の場合、戻される日付属性値。

**注意：** 引数 text\_value、number\_value および date\_value は相互に排他的です。つまり、attr\_type 引数の値に応じて、これらの引数のうち 1 つのみを使用します。

通知が送られると、システムは（属性値を取得するために）各 SEND 属性に指定されたコールバック関数をコールします。

### 例 1

SEND 属性ごとに、次のようにコールしてください。

```
your_callback('GET', context, 'BUGNO', 'NUMBER', textval, numval, dateval)
```

## 例 2

ユーザーが通知に応答すると、RESPOND 属性ごとに 1 回ずつコールバックが再コールされます。

```
your_callback('SET', context, 'STATUS', 'TEXT', 'COMPLETE', numval, dateval);
```

## 例 3

最後に、通知システムは COMPLETE コマンドをコールし、応答が終了したことを示します。

```
your_callback('COMPLETE', context, attrname, attrtype, textval, numval, dateval);
```

## SendGroup

### PL/SQL 構文

```
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;
```

### 説明

この関数は、特定のロールに割り当てられている全ユーザーに通知を個別に送信し、成功すると通知グループ ID がコールされた回数を返します。通知グループ ID では、ユーザーのグループと各グループが受け取った通知が識別されます。

メッセージにメッセージ属性がある場合、プロシージャはメッセージ属性表から属性の値を探すか、オプションのコールバック・インタフェース関数を使用して項目タイプ属性表から値を取得します。コールバック関数は、通知に対して応答があった場合にも使用できます。

---

---

**注意：** Oracle Workflow 通知システムと、電子メール・ベースまたは Web ベースの通知クライアントを使用している場合、Send プロシージャは、暗黙的に WF\_ENGINE.CB コールバック関数をコールします。独自のカスタム通知システムを使用している場合は、標準書式に従って独自のコールバック関数を定義し、コールバックの引数として名前を指定する必要があります。8-205 ページの「[カスタム・コールバック関数](#)」を参照してください。

---

---

通常、この関数がコールされるのは、通知アクティビティのプロパティ画面で「ロールの拡張」がオンになっている場合のみです。「ロールの拡張」がオンになっていない場合は、かわりに Send() 関数がコールされます。4-58 ページの「[投票アクティビティ](#)」を参照してください。

## 引数（入力）

<b>role</b>	通知アクティビティの実行者として割り当てられたロール名。
<b>msg_type</b>	メッセージに関連付けられている項目タイプ。
<b>msg_name</b>	メッセージの内部名。
<b>due_date</b>	応答期日。このオプションの期日は、受信者に知らせるためのものであり、処理には影響しません。
<b>callback</b>	SEND のソース・メッセージ属性の通信に使用されるコールバック関数の名前。
<b>context</b>	コールバック関数に渡されるコンテキスト情報。
<b>send_comment</b>	メッセージに表示するコメント。
<b>priority</b>	メッセージの優先度。 <b>#PRIORITY</b> 通知アクティビティ属性から取り出されます。 <b>#PRIORITY</b> が存在しない場合や、値が <b>NULL</b> の場合、ワークフロー・エンジンはメッセージのデフォルト優先度を使用します。

## Forward

### PL/SQL 構文

```
procedure FORWARD
(nid in number,
 new_role in varchar2,
 forward_comment in varchar2 default null);
```

### Java 構文

```
public static boolean forward
(WFContext wCtx,
 BigDecimal nid,
 String newRole
 String comment)
```

### 説明

このプロシージャは、作業を行う新しいロールに通知を委任します。ただし、通知アクティビティの所有権は、元の受信ロールに残ったままになります。また、**Send** または **SendGroup** 関数で指定したコールバック関数を、**FORWARD** モードで暗黙的にコールします。転送が行われた理由を説明するコメントを付けることもできます。既存の通知属性（期日を含む）はリフレッシュされず、変更もされません。通知システムの委任機能はこのプロシージャをコールします。通知を転送するときに、通知の **USER\_COMMENT** フィールドに転送が記録されることに注意してください。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>nid</b>	通知 ID。
<b>new_role または newRole</b>	メモが再割当てされる個人のロール名。
<b>forward_comment または comment</b>	転送に関するオプションのコメント。

### 例

次のコードは、Java プログラムで **forward()** をコールする方法の例です。このコード例は、**WFTest.java** プログラムからの引用です。

```
// 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 構文

```
procedure TRANSFER
(nid in number,
 new_role in varchar2,
 forward_comment in varchar2 default null);
```

### Java 構文

```
public static boolean transfer
(WFContext wCtx,
 BigDecimal nid,
 String newRole
 String comment)
```

### 説明

このプロシージャは、通知を新しいロールに転送し、通知の所有権を新しいロールに譲渡します。また、**Send** または **SendGroup** 関数で指定したコールバック関数を、**TRANSFER** モードで暗黙的にコールします。転送が行われた理由を説明するコメントを付けることもできます。通知システムの譲渡機能はこのプロシージャをコールします。通知を譲渡するときに、通知の **USER\_COMMENT** フィールドに譲渡が記録されることに注意してください。

---

---

**注意：** 既存の通知属性（期日を含む）はリフレッシュされません。また、通知の所有者を識別する **ORIGINAL\_RECIPIENT** 以外も変更されません。

---

---

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>nid</b>	通知 ID。
<b>new_role または newRole</b>	メモが転送される個人のロール名。
<b>forward_comment または comment</b>	通知に追加するオプションのコメント。



## 例

次のコードは、Java プログラムで `transfer()` をコールする方法の例です。このコード例は、`WFTest.java` プログラムからの引用です。

```
// 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 構文

```
procedure CANCEL
(nid in number,
 cancel_comment in varchar2 default null);
```

### Java 構文

```
public static boolean cancel
(WFContext wCtx,
 BigDecimal nid,
 String comment)
```

### 説明

このプロシージャは、通知を取り消すために送信者や管理者がコールできます。コールされると、通知ステータスは「CANCELED」に変更されますが、削除操作が行われるまで WF\_NOTIFICATIONS 表から行は削除されません。

電子メールで通知が配信され、応答が必要な場合は、その通知が有効でなくなったことを警告するために、元の受信者に「キャンセル」の電子メールが送信されます。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>nid</b>	通知 ID。
<b>cancel_comment または comment</b>	取消しに関するオプションのコメント。

## CancelGroup

### PL/SQL 構文

```
procedure CancelGroup  
(gid in number,  
 cancel_comment in varchar2 default null);
```

### 説明

このプロシージャは、通知グループ内の全ユーザーに送信された特定の通知の個別コピーを取り消すために、送信者や管理者がコールできます。通知は、通知グループ ID (gid) で識別されます。コールされると、通知ステータスは「CANCELED」に変更されますが、削除操作が行われるまで WF\_NOTIFICATIONS 表から行は削除されません。

電子メールで通知が配信され、応答が必要な場合は、その通知が有効でなくなったことを警告するために、元の受信者に「キャンセル」の電子メールが送信されます。

通常、この関数がコールされるのは、通知アクティビティのプロパティ画面で「ロールの拡張」がオンになっている場合のみです。「ロールの拡張」がオンになっていない場合は、かわりに Cancel() 関数がコールされます。4-58 ページの「[投票アクティビティ](#)」を参照してください。

### 引数（入力）

<b>gid</b>	通知グループ ID。
<b>cancel_comment</b>	取消しに関するオプションのコメント。

# Respond

## PL/SQL 構文

```
procedure RESPOND
(nid in number,
 respond_comment in varchar2 default null,
 responder in varchar2 default null);
```

## Java 構文

```
public static boolean respond
(WFContext wCtx,
 BigDecimal nid,
 String comment,
 String responder)
```

## 説明

このプロシージャは、実行者が通知への応答を終了すると、通知エージェント（「通知」Web 画面または電子メール・エージェント）によってコールされます。プロシージャは通知に「CLOSED」とマークし、コールバック関数（存在する場合）を介して、データベースに RESPOND 属性を戻します。

このプロシージャには、実際に通知に応答した個人の名前を使用できます。これは、通知がマルチ・ユーザーのロールに割り当てられている場合に特に便利です。情報は、WF\_NOTIFICATIONS 表の RESPONDER 列に保存されます。この列に格納される値は、ユーザーが通知に応答する方法によって決まります。次の表に、格納される値を応答メカニズムごとに示します。

表 8-6

応答メカニズム	格納される値
Web	Web へのログイン・ユーザー名
電子メール	応答メールに表示される電子メール・ユーザー名

## 引数（入力）

- wCtx

ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「[Oracle Workflow のコンテキスト](#)」を参照してください。
- nid

通知 ID。
- comment

応答に関するオプションのコメント。

**responder**

通知に応答したユーザー。

## Responder

### PL/SQL 構文

```
function RESPONDER  
(nid in number)  
returns varchar2;
```

### Java 構文

```
public static String responder  
(WFContext wCtx,  
    BigDecimal nid)
```

### 説明

この関数は、完了した通知の応答者を戻します。

Web 通知インタフェースを使用して通知を完了した場合、戻り値はビュー WF\_ROLES で定義されている有効なロールとなります。電子メール・インタフェースを使用して通知を完了した場合、戻り値は電子メール・アドレスです。8-216 ページの「[Respond](#)」を参照してください。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>nid</b>	通知 ID。

## VoteCount

### PL/SQL 構文

```
procedure VoteCount
(gid in number,
 ResultCode in varchar2,
 ResultCount out number,
 PercentOfTotalPop out number,
 PercentOfVotes out number);
```

### Java 構文

```
public static WFTwoDDataSource voteCount
(WFContext wCtx,
 BigDecimal gid,
 String resultCode)
```

### 説明

指定した結果コードに対する応答数をカウントします。

このプロシージャを使用するのは、ユーザー定義の投票アクティビティを記述する場合のみです。4-58 ページの「[投票アクティビティ](#)」を参照してください。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>gid</b>	通知グループ ID。
<b>ResultCode</b>	集計される結果コード。

## OpenNotificationsExist

### PL/SQL 構文

```
function OpenNotificationsExist  
  (gid in number)  
  return boolean;
```

### Java 構文

```
public static boolean openNotificationsExist  
  (WFContext wCtx,  
   BigDecimal gid)
```

### 説明

この関数は、指定した通知グループ ID に関連する通知がオープンされている場合は TRUE を返し、そうでない場合は FALSE を返します。

このプロシージャを使用するのは、ユーザー定義の投票アクティビティを記述する場合のみです。4-58 ページの「[投票アクティビティ](#)」を参照してください。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>gid</b>	通知グループ ID。



## Close

### PL/SQL 構文

```
procedure Close
(nid in number,
 responder in varchar2 default null);
```

### Java 構文

```
public static boolean close
(WFContext wCtx,
 BigDecimal nid,
 String responder)
```

### 説明

このプロシージャは通知を完了します。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>nid</b>	通知 ID。
<b>responder</b>	通知に応答したユーザーまたはロール。

## AddAttr

### PL/SQL 構文

```
procedure AddAttr
  (nid in number,
   aname in varchar2);
```

### Java 構文

```
public static boolean addAttr
  (WFContext wCtx,
   BigDecimal nid,
   String aName)
```

### 説明

新規のランタイム通知属性を追加します。**Oracle Workflow** では完全に検証されないため、検証を行い、属性の使用に関して一貫性を保つ必要があります。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>nid</b>	通知 ID。
<b>aname</b>	属性名。
<b>avalue</b>	属性値。

### 例

次のコードは、Java プログラムで `addAttr()` をコールする方法の例です。このコード例は、`WFTest.java` プログラムからの引用です。

```
if (WFNotificationAPI.addAttr(ctx, myNid, myAttr) == false)
{
  System.out.println("Add attribute " + myAttr + " failed.");
}
```

## SetAttribute

### PL/SQL 構文

```
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 構文

```
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)
```

### 説明

通知属性の値を設定するために、送信時と応答時に使用されます。通知エージェント（送信者）は、SEND 属性の値を設定できます。実行者（応答者）は、RESPOND 属性の値を設定できます。

## 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>nid</b>	通知 ID。
<b>aname</b>	属性名。
<b>avalue</b>	属性値。

## 例

次のコードは、Java プログラムで `setAttribute` メソッドをコールする方法の例です。このコード例は、`WFTTest.java` プログラムからの引用です。

```
if (WFNotificationAPI.setAttrDate(ctx, myNid, myAttr, value)
    == false)
{
    System.out.println("set attribute " + myAttr + " to " +
        value + " failed.");
}
```

## GetAttrInfo

### PL/SQL 構文

```
procedure GetAttrInfo
(nid in number,
 aname in varchar2,
 atype out varchar2,
 subtype out varchar2,
 format out varchar2);
```

### Java 構文

```
public static WFTwoDataSource getAttrInfo
(WFContext wCtx,
 BigDecimal nid,
 String aName)
```

### 説明

タイプ、サブタイプおよび書式など、通知属性に関して指定されている情報があれば、それに戻します。属性のソースを示すサブタイプは、常に SEND または RESPOND です。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>nid</b>	通知 ID。
<b>aname</b>	属性名。

### 例

次のコードは、Java プログラムで `getAttrInfo()` をコールする方法の例です。このコード例は、`WFTTest.java` プログラムからの引用です。

```
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 構文

```
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 構文

```
public static WFTwoDDataSource getInfo
(WFContext wCtx,
 BigDecimal nid)
```

### 説明

指定した通知について、通知の送信先のロール、メッセージの項目タイプ、メッセージの名前、通知の優先度、期日およびステータスを戻します。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>nid</b>	通知 ID。

### 例

次のコードは、Java プログラムで `getInfo()` をコールする方法の例です。このコード例は、`WFTest.java` プログラムからの引用です。

```
// Notification Info
System.out.println("Notification Info for nid " + myNid);
dataSource = WFNotificationAPI.getInfo(ctx, myNid);
displayDataSource(ctx, dataSource);
```

## GetText

### PL/SQL 構文

```
function GetText
  (some_text in varchar2,
   nid in number,
   disptype in varchar2 default '')
  return varchar2;
```

### 説明

特定の通知からのトークン値を使用して、任意のテキスト文字列のトークンを置き換えます。この関数は、最大 32KB の文字を戻します。ビュー定義や Oracle Forms Developer フォームでは、この関数は使用できません。ビューとフォームには、値を 1950 文字に切り捨てる `GetShortText()` を使用してください。

エラーが検出されると、この関数は置換されなかった `some_text` を戻し、例外は発生しません。

### 引数（入力）

<b>some_text</b>	置き換えられるテキスト。
<b>nid</b>	トークン値に使用する通知の通知 ID。
<b>disptype</b>	テキストでトークンを置換するメッセージ本文の表示タイプ。有効な表示タイプは次のとおりです。 <ul style="list-style-type: none"><li>■ <code>wf_notification.doc_text</code>。text/plain を戻します。</li><li>■ <code>wf_notification.doc_html</code>。text/html を戻します。</li><li>■ <code>wf_notification.doc_attach</code>。NULL を戻します。</li></ul> デフォルトは NULL です。

## GetShortText

### PL/SQL 構文

```
function GetShortText
  (some_text in varchar2,
   nid in number)
  return varchar2;
```

### 説明

特定の通知からのトークン値を使用して、任意のテキスト文字列のトークンを置き換えます。この関数は、最大 1950 文字を戻します。この関数は、フィールド・サイズが 1950 文字に限られているビュー定義や **Oracle Forms Developer** フォームでの使用に適しています。最大で 32KB の文字を取り出す必要がある場合は、**GetText()** を使用してください。

エラーが検出されると、この関数は置換されなかった `some_text` を戻し、例外は発生しません。

### 引数（入力）

<b>some_text</b>	置き換えられるテキスト。
<b>nid</b>	トークン値に使用する通知の通知 ID。



## GetAttribute

### PL/SQL 構文

```
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 構文

```
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)
```

### 説明

指定されたメッセージ属性の値を戻します。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
-------------	---

**nid**                                      通知 ID。

**aname**                                    メッセージ属性の名前。

## 例

次のコードは、Java プログラムで `getAttribute` メソッドをコールする方法の例です。このコード例は、`WFTTest.java` プログラムからの引用です。

```
// 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 構文

```
function GetAttrDoc
  (nid in number,
   aname in varchar2,
   disptype in varchar2)
  return varchar2;
```

### Java 構文

```
public static String getAttrDoc
  (WFContext wCtx,
   BigDecimal nid,
   String aName,
   String dispType)
```

### 説明

文書タイプ属性の表示値を戻します。参照先の文書は、要求に応じてプレーン・テキストまたは HTML 形式で表示されます。

実際の属性値、つまり実際の文書ではなく文書キー文字列を取り出す場合は、`GetAttrText()` を使用してください。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>nid</b>	通知 ID。
<b>aname</b>	メッセージ属性の名前。
<b>disptype</b>	戻される文書の表示タイプ。有効な表示タイプは次のとおりです。 <ul style="list-style-type: none"><li>■ <code>wf_notification.doc_text</code>。text/plain を戻します。</li><li>■ <code>wf_notification.doc_html</code>。text/html を戻します。</li><li>■ <code>wf_notification.doc_attach</code>。NULL を戻します。</li></ul>

## GetSubject

### PL/SQL 構文

```
function GetSubject  
(nid in number)  
return varchar2
```

### Java 構文

```
public static String getSubject  
(WFContext wCtx,  
BigDecimal nid)
```

### 説明

通知メッセージの件名の行を戻します。件名のメッセージ属性はすべて、対応するメッセージ属性の値でトークンが置換されます。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>nid</b>	通知 ID。

## GetBody

### PL/SQL 構文

```
function GetBody
(nid in number,
 disptype in varchar2 default '')
return varchar2;
```

### Java 構文

```
public static String getBody
(WFContext wCtx,
 BigDecimal nid,
 String dispType)
```

### 説明

指定されたメッセージ本文のタイプに応じて、通知の HTML またはプレーン・テキストのメッセージ本文を返します。本文のメッセージ属性はすべて、対応する通知属性の値でトークンが置換されます。この関数は、最大 32KB の文字を返します。ビュー定義や Oracle Applications フォームでは、この関数は使用できません。ビューとフォームには、値を 1950 文字に切り捨てる `GetShortBody()` を使用してください。

戻されるプレーン・テキストのメッセージ本文はフォーマットされないため注意してください。出力デバイスにあわせてワードラップする必要があります。本文テキストには、タブ（インデントを表す）および改行（段落の終了を表す）が含まれます。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>nid</b>	通知 ID。
<b>disptype</b>	フェッチする文書の表示タイプ。有効な表示タイプは次のとおりです。 <ul style="list-style-type: none"><li>■ wf_notification.doc_text。text/plain を返します。</li><li>■ wf_notification.doc_html。text/html を返します。</li><li>■ wf_notification.doc_attach。NULL を返します。</li></ul> デフォルトは NULL です。

## GetShortBody

### PL/SQL 構文

```
function GetShortBody  
(nid in number)  
return varchar2;
```

### 説明

通知のメッセージ本文を返します。本文のメッセージ属性はすべて、対応する通知属性の値でトークンが置換されます。この関数は、最大 1950 文字を返します。この関数は、フィールド・サイズが 1950 文字に限られているビュー定義や Oracle Forms Developer フォームでの使用に適しています。最大で 32KB の文字を取り出す必要がある場合は、GetBody() を使用してください。

戻されるプレーン・テキストのメッセージ本文はフォーマットされないため注意してください。出力デバイスにあわせてワードラップする必要があります。本文テキストには、タブ（インデントを表す）および改行（段落の終了を表す）が含まれます。

エラーが検出されると、この関数は置換されなかった本文、または他のすべてが失敗した場合は NULL を返し、例外は発生しません。

---

---

**注意：** この関数は、メッセージをフォームやビューでのみ表示することを意図しています。

---

---

### 引数（入力）

nid	通知 ID。
-----	--------

## TestContext

### PL/SQL 構文

```
function TestContext  
(nid in number)  
return boolean;
```

### 説明

項目タイプのセクタ / コールバック関数をコールして、現行のコンテキストが正しいかどうかをテストします。コンテキスト・チェックで問題がない場合、またはセクタ / コールバック関数が実装されない場合、この関数は TRUE を返します。コンテキスト・チェックで問題があった場合は FALSE を返します。

### 引数（入力）

<b>nid</b>	通知 ID。
------------	--------

## AccessCheck

### PL/SQL 構文

```
function AccessCheck  
(access_str in varchar2)  
return varchar2;
```

### Java 構文

```
public static String accessCheck  
(WFContext wCtx,  
String accessString)
```

### 説明

通知のアクセス文字列が有効で、通知がオープンされている場合はユーザー名を戻し、それ以外の場合は NULL を戻します。アクセス文字列は、通知メーカーによって自動的に作成され、電子メール通知のテキスト版と HTML 版の両方の信頼性を検証するために使用されます。

### 引数（入力）

**wCtx**

ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「[Oracle Workflow のコンテキスト](#)」を参照してください。

**access\_str または accessString**

nid/nkey 形式のアクセス文字列。nid は通知 ID で、nkey は通知キーです。



## WorkCount

### PL/SQL 構文

```
function WorkCount  
(username in varchar2)  
return number;
```

### Java 構文

```
public static BigDecimal workCount  
(WFContext wCtx,  
String userName)
```

### 説明

ロールに割り当てられているオープン通知の数を返します。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>username</b>	ロールの内部名。

## getNotifications

### Java 構文

```
public static WFTwoDDDataSource getNotifications  
    (WFContext wCtx,  
     String itemType,  
     String itemKey)
```

### 説明

指定された項目タイプおよび項目キーの通知のリストを戻します。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>itemType</b>	項目タイプの内部名。
<b>itemKey</b>	アプリケーション・オブジェクトの主キーから導出される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセス・インスタンスが識別されます。

## getNotificationAttributes

### Java 構文

```
public static WFTwoDDataSource getNotificationAttributes  
    (WFContext wCtx,  
     BigDecimal nid)
```

### 説明

指定された通知 ID の通知属性と対応する値のリストを返します。

### 引数（入力）

<b>wCtx</b>	ワークフローのコンテキスト情報。Java メソッドの場合にのみ必須です。8-6 ページの「 <a href="#">Oracle Workflow のコンテキスト</a> 」を参照してください。
<b>nid</b>	通知 ID。

### 例

次のコードは、Java プログラムで `getNotificationAttributes()` をコールする方法の例です。このコード例は、`WFTest.java` プログラムからの引用です。

```
// List available Notification Attributes  
System.out.println("List of Attributes for id " + myNid  
    + ");  
dataSource =  
    WFNotificationAPI.getNotificationAttributes(ctx, myNid);  
displayDataSource(ctx, dataSource);
```

## WriteToClob

### PL/SQL 構文

```
procedure WriteToClob  
(clob_loc in out clob,  
 activity in varchar2);
```

### 説明

キャラクタ・ラージ・オブジェクト（CLOB）の最後に文字列を追加します。このプロシージャを使用すると、通知に追加する PL/SQL CLOB 文書属性用の CLOB を簡単に作成できます。

### 引数（入力）

<b>clob_loc</b>	文字列の追加先の CLOB。
<b>msg_string</b>	文字データの文字列。

#### 関連項目：

4-13 ページ [「文書属性の定義」](#)

7-17 ページ [「PL/SQL CLOB 文書」](#)

## Oracle Workflow ビジネス・イベント・システムの概要

Oracle Workflow ビジネス・イベント・システムは、Oracle Advanced Queuing のインフラストラクチャを活用しながら、システム間でビジネス・イベントを交換します。重要なビジネス・イベントがシステム上のインターネットまたはイントラネットのアプリケーションで発生すると、そのイベントに対して実行される処理が指定されているイベント・サブスクリプションがトリガーされます。

サブスクリプションには、次のタイプの処理を指定できます。

- イベント情報に対するカスタム・コードの実行
- ワークフロー・プロセスへのイベント情報の送信
- ローカル・システムまたは外部システム上でエージェントをコールした名前付き通信ポイントに対して、イベント情報を送信する

ビジネス・イベント・システムから伝達されるイベント情報は、イベント・メッセージと呼ばれます。イベント・メッセージは、イベントを識別するヘッダー・プロパティと、イベントの内容を説明するイベント・データから構成されます。

イベント、システム、エージェントおよびサブスクリプションは、イベント・マネージャに定義します。また、**Workflow Builder** にイベント・アクティビティを定義すれば、ビジネス・イベントをワークフロー・プロセスに組み込むことができます。

### 関連項目：

[13-2 ページ「ビジネス・イベントの管理」](#)

[4-42 ページ「イベント・アクティビティ」](#)

## ビジネス・イベント・システムのデータ型

Oracle Workflow では、いくつかの抽象データ型（ADT）を使用して、ビジネス・イベント・システムのデータの構造および動作をモデル化しています。次のデータ型があります。

- エージェント構造 : WF\_AGENT\_T
- パラメータ構造 : WF\_PARAMETER\_T
- パラメータ・リスト構造 : WF\_PARAMETER\_LIST\_T
- イベント・メッセージ構造 : WF\_EVENT\_T

ビジネス・イベント・システムのデータ型は、`wftypes.sql` というスクリプトによって作成されます。このスクリプトは、Oracle Workflow のスタンドアロン版の場合は Oracle Workflow の `sql` サブディレクトリに、Oracle Applications embedded Workflow の場合は `$FND_TOP` の `sql` サブディレクトリに格納されています。

### 関連項目：

『Oracle9i データベース概要』の「ユーザー定義データ型」

## エージェント構造

Oracle Workflow では、オブジェクト・タイプ WF\_AGENT\_T を使用して、エージェントに関する情報を、イベント・メッセージから参照できる形式で格納します。次の表に、WF\_AGENT\_T データ型の属性を示します。

表 8-7

属性名	データ型	説明
NAME	VARCHAR2(30)	エージェントの名前。
SYSTEM	VARCHAR2(30)	エージェントが配置されているシステム。

オブジェクト・タイプ WF\_AGENT\_T には、次のメソッドも組み込まれています。これらのメソッドを使用して、属性の値を取得および設定できます。

- getName
- getSystem
- setName
- setSystem

### getName

#### PL/SQL 構文

```
MEMBER FUNCTION getName
    return varchar2
```

#### 説明

WF\_AGENT\_T オブジェクトの NAME 属性の値を返します。

### getSystem

#### PL/SQL 構文

```
MEMBER FUNCTION getSystem
    return varchar2
```

#### 説明

WF\_AGENT\_T オブジェクトの SYSTEM 属性の値を返します。

## setName

### PL/SQL 構文

```
MEMBER PROCEDURE setName  
    (pName in varchar2)
```

### 説明

WF\_AGENT\_T オブジェクトの NAME 属性の値を設定します。

### 引数（入力）

**pName**                      NAME 属性の値。

## setSystem

### PL/SQL 構文

```
MEMBER PROCEDURE setSystem  
    (pSystem in varchar2)
```

### 説明

WF\_AGENT\_T オブジェクトの SYSTEM 属性の値を設定します。

### 引数（入力）

**pSystem**                      SYSTEM 属性の値。

#### 関連項目：

13-24 ページ [「エージェント」](#)



## パラメータ構造

Oracle Workflow では、オブジェクト・タイプ WF\_PARAMETER\_T を使用して、パラメータの名前と値のペアを、イベント・メッセージのパラメータ・リストに追加できる形式で格納します。WF\_PARAMETER\_T を使用すると、カスタム値を WF\_EVENT\_T イベント・メッセージ・オブジェクトに追加できます。次の表に、WF\_PARAMETER\_T データ型の属性を示します。

表 8-8

属性名	データ型	説明
NAME	VARCHAR2(30)	パラメータ名。
VALUE	VARCHAR2(2000)	パラメータ値。

オブジェクト・タイプ WF\_PARAMETER\_T には、次のメソッドも組み込まれています。これらのメソッドを使用して、属性の値を取得および設定できます。

- getName
- getValue
- setName
- setValue

### getName

#### PL/SQL 構文

```
MEMBER FUNCTION getName
    return varchar2
```

#### 説明

WF\_PARAMETER\_T オブジェクトの NAME 属性の値を返します。

### getValue

#### PL/SQL 構文

```
MEMBER FUNCTION getValue
    return varchar2
```

## 説明

WF\_PARAMETER\_T オブジェクトの VALUE 属性の値を返します。

## setName

### PL/SQL 構文

```
MEMBER PROCEDURE setName  
    (pName in varchar2)
```

## 説明

WF\_PARAMETER\_T オブジェクトの NAME 属性の値を設定します。

## 引数（入力）

**pName**                      NAME 属性の値。

## setValue

### PL/SQL 構文

```
MEMBER PROCEDURE setValue  
    (pValue in varchar2)
```

## 説明

WF\_PARAMETER\_T オブジェクトの VALUE 属性の値を設定します。

## 引数（入力）

**pValue**                      VALUE 属性の値。

## パラメータ・リスト構造

Oracle Workflow では、名前付き可変配列 (varray) `WF_PARAMETER_LIST_T` を使用して、パラメータのリストを、イベント・メッセージに組み込める形式で格納します。`WF_PARAMETER_LIST_T` を使用すると、カスタム値を `WF_EVENT_T` イベント・メッセージ・オブジェクトに追加できます。`WF_PARAMETER_LIST_T` データ型には、パラメータの名前と値のペアを最大 100 個指定できます。このデータ型の概要です。

### **WF\_PARAMETER\_LIST\_T**

- 最大サイズ : 100
- 要素のデータ型 : `WF_PARAMETER_T`

イベント・メッセージ構造

Oracle Workflow では、オブジェクト・タイプ WF\_EVENT\_T を使用してイベント・メッセージを格納します。このデータ型には、イベント・メッセージのすべてのヘッダー・プロパティと、イベント・データのペイロードで構成されます。これらの要素は、シリアル化されており、システムの外部への転送に適しています。

WF\_EVENT\_T には、ビジネス・イベント・システムとワークフロー・エンジンがビジネス・イベントを表現するときに使用する、イベント・メッセージ構造を定義します。Oracle Workflow 内部では、ビジネス・イベント・システムとワークフロー・エンジンはこの形式でイベントを伝達します。

ビジネス・イベント・システムの標準キュー（WF\_IN、WF\_OUT、WF\_DEFERRED および WF\_ERROR）では、ペイロード・タイプとして WF\_EVENT\_T が使用されます。別のペイロード・タイプのキュー（システムに定義済の既存のキューなど）を使用する場合は、キュー・ハンドラを作成して、Workflow 標準の WF\_EVENT\_T 構造とカスタム・ペイロード・タイプとを変換する必要があります。2-91 ページの「[キューの設定](#)」および 7-21 ページの「[キュー・ハンドラの標準 API](#)」を参照してください。

次の表に、WF\_EVENT\_T データ型の属性を示します。

表 8-9

属性名	データ型	説明
PRIORITY	NUMBER	メッセージ受信者がメッセージをデキューするときの優先度。数値が小さいほど、優先度は高くなります。たとえば、1 は高い優先度、50 は通常の優先度、99 は低い優先度を表します。
SEND_DATE	DATE	DATE メッセージをデキューできる日時。この送信日には、システム日付（すぐにデキューされる）または未来日付（後でデキューされる）を設定できます。  イベントが呼び出されたときに送信日が未来日付に設定されている場合、イベント・メッセージは WF_DEFERRED キューに格納され、指定された日付までサブスクリプション処理は開始されません。イベントがエージェントに送信されたときに送信日が未来日付に設定されている場合、イベント・メッセージはそのエージェントのキューに伝播されますが、コンシューマは指定された日付までイベント・メッセージをデキューできません。
RECEIVE_DATE	DATE	エージェント・リスナーによってメッセージがデキューされる日時。

表 8-9 (続き)

属性名	データ型	説明
CORRELATION_ID	VARCHAR2(240)	このメッセージを他のメッセージと関連付けるための相関 ID。この属性は初期値は空白で、関数によって設定されます。相関 ID に値が設定されている場合、イベントがワークフロー・プロセスに送信されると、その値は項目キーとして使用されます。
PARAMETER_LIST	WF_PARAMETER_LIST_T	追加パラメータの名前と値のペアのリスト。
EVENT_NAME	VARCHAR2(240)	イベントの内部名。
EVENT_KEY	VARCHAR2(240)	イベントのインスタンスを一意に識別するための文字列。
EVENT_DATA	CLOB	イベントの内容の詳細情報。イベント・データは、XML 文書として作成できます。
FROM_AGENT	WF_AGENT_T	イベントの送信元のエージェント。イベントがローカルで発生した場合、この属性の初期値は NULL です。
TO_AGENT	WF_AGENT_T	イベントの送信先のエージェント (メッセージ受信者)。
ERROR_SUBSCRIPTION	RAW(16)	このイベントの処理時にエラーが発生した場合、この属性にはエラー発生時に実行されていたサブスクリプションが設定されます。
ERROR_MESSAGE	VARCHAR2(4000)	このイベントの処理時にエラーが発生した場合、イベント・マネージャによって生成されるエラー・メッセージ。
ERROR_STACK	VARCHAR2(4000)	このイベントの処理時にエラーが発生した場合、イベント・マネージャによって生成される引数のエラー・スタック。エラー・スタックには、エラーの原因を特定するときに役立つコンテキスト情報が格納されます。

オブジェクト・タイプ WF\_EVENT\_T には、次のメソッドも組み込まれています。これらのメソッドを使用して、属性の値を取得および設定できます。

- 8-251 ページ [「Initialize」](#)
- 8-251 ページ [「getPriority」](#)
- 8-252 ページ [「getSendDate」](#)
- 8-252 ページ [「getReceiveDate」](#)
- 8-252 ページ [「getCorrelationID」](#)

- 8-252 ページ [「getParameterList」](#)
- 8-253 ページ [「getEventName」](#)
- 8-253 ページ [「getEventKey」](#)
- 8-253 ページ [「getEventData」](#)
- 8-253 ページ [「getFromAgent」](#)
- 8-254 ページ [「getToAgent」](#)
- 8-254 ページ [「getErrorSubscription」](#)
- 8-254 ページ [「getErrorMessage」](#)
- 8-254 ページ [「getErrorStack」](#)
- 8-255 ページ [「setPriority」](#)
- 8-255 ページ [「setSendDate」](#)
- 8-255 ページ [「setReceiveDate」](#)
- 8-256 ページ [「setCorrelationID」](#)
- 8-256 ページ [「setParameterList」](#)
- 8-257 ページ [「setEventName」](#)
- 8-257 ページ [「setEventKey」](#)
- 8-257 ページ [「setEventData」](#)
- 8-258 ページ [「setFromAgent」](#)
- 8-258 ページ [「setToAgent」](#)
- 8-258 ページ [「setErrorSubscription」](#)
- 8-259 ページ [「setErrorMessage」](#)
- 8-259 ページ [「setErrorStack」](#)
- 8-260 ページ [「Content」](#)
- 8-260 ページ [「Address」](#)
- 8-261 ページ [「AddParameterToList」](#)
- 8-261 ページ [「GetValueForParameter」](#)

---

---

**注意：** EVENT\_NAME、EVENT\_KEY および EVENT\_DATA 属性の値は、setEventName、setEventKey および setEventData の各メソッドを使用して個別に設定する以外に、Content メソッドを使用してイベント・コンテンツ属性をまとめて設定することもできます。8-260 ページの「[Content](#)」を参照してください。

同様に、FROM\_AGENT、TO\_AGENT、PRIORITY および SEND\_DATE 属性の値も、setFromAgent、setToAgent、setPriority、setSendDate の各メソッドを使用して個別に設定したり、Address メソッドを使用してアドレス属性をまとめて設定することもできます。8-260 ページの「[Address](#)」を参照してください。

---

---

## Initialize

### PL/SQL 構文

```
STATIC PROCEDURE initialize  
    (new_wf_event_t in out wf_event_t)
```

### 説明

新しい WF\_EVENT\_T オブジェクトを初期化するために、PRIORITY 属性を 0 に設定し、Empty\_CLOB() 関数を使用して EVENT\_DATA 属性を EMPTY に初期設定し、他のすべての属性を NULL に設定します。

---

---

**注意：** 新しい WF\_EVENT\_T オブジェクトを操作するには、まず Initialize メソッドをコールする必要があります。

---

---

### 引数（入力）

**new\_wf\_event\_t**                      初期化する WF\_EVENT\_T オブジェクト。

## getPriority

### PL/SQL 構文

```
MEMBER FUNCTION getPriority  
    return number
```

### 説明

WF\_EVENT\_T オブジェクトの PRIORITY 属性の値を返します。

## getSendDate

### PL/SQL 構文

```
MEMBER FUNCTION getSendDate  
    return date
```

### 説明

WF\_EVENT\_T オブジェクトの SEND\_DATE 属性の値を返します。

## getReceiveDate

### PL/SQL 構文

```
MEMBER FUNCTION getReceiveDate  
    return date
```

### 説明

WF\_EVENT\_T オブジェクトの RECEIVE\_DATE 属性の値を返します。

## getCorrelationID

### PL/SQL 構文

```
MEMBER FUNCTION getCorrelationID  
    return varchar2
```

### 説明

WF\_EVENT\_T オブジェクトの CORRELATION\_ID 属性の値を返します。

## getParameterList

### PL/SQL 構文

```
MEMBER FUNCTION getParameterList  
    return wf_parameter_list_t
```



**説明**

WF\_EVENT\_T オブジェクトの PARAMETER\_LIST 属性の値を返します。

**getEventName****PL/SQL 構文**

```
MEMBER FUNCTION getEventName  
    return varchar2
```

**説明**

WF\_EVENT\_T オブジェクトの EVENT\_NAME 属性の値を返します。

**getEventKey****PL/SQL 構文**

```
MEMBER FUNCTION getEventKey  
    return varchar2
```

**説明**

WF\_EVENT\_T オブジェクトの EVENT\_KEY 属性の値を返します。

**getEventData****PL/SQL 構文**

```
MEMBER FUNCTION getEventData  
    return clob
```

**説明**

WF\_EVENT\_T オブジェクトの EVENT\_DATA 属性の値を返します。

**getFromAgent****PL/SQL 構文**

```
MEMBER FUNCTION getFromAgent  
    return wf_agent_t
```

### 説明

WF\_EVENT\_T オブジェクトの FROM\_AGENT 属性の値を返します。

## getToAgent

### PL/SQL 構文

```
MEMBER FUNCTION getToAgent  
    return wf_agent_t
```

### 説明

WF\_EVENT\_T オブジェクトの TO\_AGENT 属性の値を返します。

## getErrorSubscription

### PL/SQL 構文

```
MEMBER FUNCTION getErrorSubscription  
    return raw
```

### 説明

WF\_EVENT\_T オブジェクトの ERROR\_SUBSCRIPTION 属性の値を返します。

## getErrorMessage

### PL/SQL 構文

```
MEMBER FUNCTION getErrorMessage  
    return varchar2
```

### 説明

WF\_EVENT\_T オブジェクトの ERROR\_MESSAGE 属性の値を返します。

## getErrorStack

### PL/SQL 構文

```
MEMBER FUNCTION getErrorStack  
    return varchar2
```

## 説明

WF\_EVENT\_T オブジェクトの ERROR\_STACK 属性の値を返します。

## setPriority

### PL/SQL 構文

```
MEMBER PROCEDURE setPriority  
    (pPriority in number)
```

## 説明

WF\_EVENT\_T オブジェクトの PRIORITY 属性の値を設定します。

## 引数（入力）

pPriority                      PRIORITY 属性の値。

## setSendDate

### PL/SQL 構文

```
MEMBER PROCEDURE setSendDate  
    (pSendDate in date default sysdate)
```

## 説明

WF\_EVENT\_T オブジェクトの SEND\_DATE 属性の値を設定します。

## 引数（入力）

pSendDate                    SEND\_DATE 属性の値。

## setReceiveDate

### PL/SQL 構文

```
MEMBER PROCEDURE setReceiveDate  
    (pReceiveDate in date default sysdate)
```

## 説明

WF\_EVENT\_T オブジェクトの RECEIVE\_DATE 属性の値を設定します。

## 引数（入力）

**pReceiveDate**                      RECEIVE\_DATE 属性の値

## setCorrelationID

### PL/SQL 構文

```
MEMBER PROCEDURE setCorrelationID  
    (pCorrelationID in varchar2)
```

## 説明

WF\_EVENT\_T オブジェクトの CORRELATION\_ID 属性の値を設定します。

## 引数（入力）

**pCorrelationID**                      CORRELATION\_ID 属性の値。

## setParameterList

### PL/SQL 構文

```
MEMBER PROCEDURE setParameterList  
    (pParameterList in wf_parameter_list_t)
```

## 説明

WF\_EVENT\_T オブジェクトの PARAMETER\_LIST 属性の値を設定します。

## 引数（入力）

**pParameterList**                      PARAMETER\_List 属性の値。

## setEventName

### PL/SQL 構文

```
MEMBER PROCEDURE setEventName  
    (pEventName in varchar2)
```

### 説明

WF\_EVENT\_T オブジェクトの EVENT\_NAME 属性の値を設定します。

### 引数（入力）

**pEventName**                      EVENT\_NAME 属性の値。

## setEventKey

### PL/SQL 構文

```
MEMBER PROCEDURE setEventKey  
    (pEventKey in varchar2)
```

### 説明

WF\_EVENT\_T オブジェクトの EVENT\_KEY 属性の値を設定します。

### 引数（入力）

**pEventKey**                      EVENT\_KEY 属性の値。

## setEventData

### PL/SQL 構文

```
MEMBER PROCEDURE setEventData  
    (pEventData in clob)
```

### 説明

WF\_EVENT\_T オブジェクトの EVENT\_DATA 属性の値を設定します。

## 引数（入力）

**pEventData**                      EVENT\_DATA 属性の値。

## setFromAgent

### PL/SQL 構文

```
MEMBER PROCEDURE setFromAgent  
    (pFromAgent in wf_agent_t)
```

### 説明

WF\_EVENT\_T オブジェクトの FROM\_AGENT 属性の値を設定します。

## 引数（入力）

**pFromAgent**                      FROM\_AGENT 属性の値。

## setToAgent

### PL/SQL 構文

```
MEMBER PROCEDURE setToAgent  
    (pToAgent in wf_agent_t)
```

### 説明

WF\_EVENT\_T オブジェクトの TO\_AGENT 属性の値を設定します。

## 引数（入力）

**pToAgent**                      TO\_AGENT 属性の値。

## setErrorSubscription

### PL/SQL 構文

```
MEMBER PROCEDURE setErrorSubscription  
    (pErrorSubscription in raw)
```

## 説明

WF\_EVENT\_T オブジェクトの ERROR\_SUBSCRIPTION 属性の値を設定します。

## 引数（入力）

pErrorSubscription      ERROR\_SUBSCRIPTION 属性の値。

# setErrorMessage

## PL/SQL 構文

```
MEMBER PROCEDURE setErrorMessage  
    (pErrorMessage in varchar2)
```

## 説明

WF\_EVENT\_T オブジェクトの ERROR\_MESSAGE 属性の値を設定します。

## 引数（入力）

pErrorMessage              ERROR\_MESSAGE 属性の値。

# setErrorStack

## PL/SQL 構文

```
MEMBER PROCEDURE setErrorStack  
    (pErrorStack in varchar2)
```

## 説明

WF\_EVENT\_T オブジェクトの ERROR\_STACK 属性の値を設定します。

## 引数（入力）

pErrorStack                ERROR\_STACK 属性の値。

## Content

### PL/SQL 構文

```
MEMBER PROCEDURE Content
  (pName in varchar2,
   pKey in varchar2,
   pData in clob)
```

### 説明

EVENT\_NAME、EVENT\_KEY および EVENT\_DATA など、WF\_EVENT\_T オブジェクトのすべてのイベント・コンテンツ属性の値を設定します。

### 引数（入力）

<b>pName</b>	EVENT_NAME 属性の値。
<b>pKey</b>	EVENT_KEY 属性の値。
<b>pData</b>	EVENT_DATA 属性の値。

## Address

### PL/SQL 構文

```
MEMBER PROCEDURE Address
  (pOutAgent in wf_agent_t,
   pToAgent in wf_agent_t,
   pPriority in number,
   pSendDate in date)
```

### 説明

FROM\_AGENT、TO\_AGENT、PRIORITY および SEND\_DATE など、WF\_EVENT\_T オブジェクトのすべてのアドレス属性の値を設定します。

### 引数（入力）

<b>pOutAgent</b>	FROM_AGENT 属性の値。
<b>pToAgent</b>	TO_AGENT 属性の値。
<b>pPriority</b>	PRIORITY 属性の値。



**pSendDate**                      SEND\_DATE 属性の値。

## AddParameterToList

### PL/SQL 構文

```
MEMBER PROCEDURE AddParameterToList
    (pName in varchar2,
     pValue in varchar2)
```

### 説明

WF\_EVENT\_T オブジェクトの PARAMETER\_LIST 属性に格納されているリストに、新しいパラメータの名前と値のペアを追加します。指定された名前のパラメータがパラメータ・リストにすでに存在する場合は、そのパラメータの前の値が上書きされます。

### 引数（入力）

**pName**                      パラメータ名。  
**pValue**                    パラメータ値。

## GetValueForParameter

### PL/SQL 構文

```
MEMBER FUNCTION GetValueForParameter
    (pName in varchar2) return varchar2
```

### 説明

WF\_EVENT\_T オブジェクトの PARAMETER\_LIST 属性に格納されているリストから、指定されたパラメータの値を返します。このメソッドは、パラメータ・リストの終わりから先頭方向に検索します。指定された名前のパラメータがパラメータ・リストにない場合は、GetValueForParameter メソッドによって NULL が返されます。

### 引数（入力）

**pName**                      パラメータ名。

## 抽象データ型の使用例

次の例は、SQL スクリプトで抽象データ型メソッドを使用する方法をいくつか示しています。

- Initialize メソッドを使用して、新しいイベント・メッセージ構造を初期化する。

---

**注意：** 新しい WF\_EVENT\_T オブジェクトを操作するには、まず Initialize メソッドをコールする必要があります。

---

- CLOB ロケータを初期化する。
- テキスト変数を CLOB 変数に書き込む。
- Content メソッドを使用して、イベント・メッセージ構造のコンテンツ属性を設定する。
- Address メソッドを使用して、イベント・メッセージ構造のアドレス属性を設定する。

次のコード例は、スクリプト wfeventnq.sql の一部で、オーバーライド・エージェントを使用してイベント・メッセージをキューに格納します。16-11 ページの「wfeventnq.sql」を参照してください。

```
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;
```

WF\_EVENT\_T および OMBAQ\_TEXT\_MSG 間のマッピング

Oracle8iを使用している場合は、Oracle Message Broker（OMB）を実装してシステム間でイベント・メッセージを伝播できます。OMB キューにメッセージを格納するときは、OMBAQ\_TEXT\_MSG と呼ばれる Java Message Service の抽象データ型で定義した構造でなければなりません。

Oracle Workflow には、WF\_EVENT\_OMB\_QH と呼ばれるキュー・ハンドラが用意されています。このキュー・ハンドラを使用すれば、Workflow 標準の WF\_EVENT\_T メッセージ構造と OMBAQ\_TEXT\_MSG 構造とを変換できます。2-94 ページの「[手順 WF-15 WF\\_EVENT\\_OMB\\_QH キュー・ハンドラの設定](#)」および 13-24 ページの「[エージェント](#)」を参照してください。

OMBAQ\_TEXT\_MSG データ型は、TEXT\_LOB および HEADER 属性で構成されます。TEXT\_LOB 属性には、CLOB 形式でメッセージのペイロードが格納されます。HEADER 属性のデータ型は、OMBAQ\_HEADER と呼ばれる ADT です。

OMBAQ\_HEADER には、PROPERTIES と呼ばれる属性が格納されます。データ型は ADT で、OMBAQ\_PROPERTIES と呼ばれる名前付き可変配列です。OMBAQ\_PROPERTIES の最大サイズは 1000 です。その要素のデータ型も ADT で、OMBAQ\_PROPERTY と呼ばれます。

次の一覧に、WF\_EVENT\_T メッセージ構造の属性と、OMBAQ\_TEXT\_MSG 構造内の属性の対応表です。

表 8-10

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]
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 = FROMAGENTSYSYSTEM]

表 8-10 (続き)

WF_EVENT_T	OMBAQ_TEXT_MSG
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 = TOAGENTSYSYSTEM]
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>]

**注意：** イベントのパラメータ・リスト内のパラメータには、イベント・プロパティの予約語を除く、任意の名前を使用できます。予約語は、次のとおりです。

PRIORITY、SENDDATE、RECEIVEDATE、CORRELATIONID、  
EVENTNAME、EVENTKEY、FROMAGENTNAME、  
FROMAGENTSYSYSTEM、TOAGENTNAME、TOAGENTSYSYSTEM、  
ERRORSUBSCRIPTION、ERRORMESSAGE1、ERRORMESSAGE2、  
ERRORSTACK1、ERRORSTACK2

**注意：** Oracle Message Broker および OMBAQ\_TEXT\_MSG データ型は、Oracle9i では使用されていません。Oracle9i では、Oracle Message Broker のかわりに、Oracle Advanced Queuing のメッセージ・ゲートウェイおよびインターネット・アクセス機能を使用して、イベント・メッセージを伝播できます。

**関連項目：**

『Oracle Message Broker 管理者ガイド』の「Oracle AQ Driver ADTs」

## イベントの API

イベントの API は、アプリケーション・プログラムまたはワークフロー・プロセスから実行時にコールされ、ビジネス・イベント・システムとの通信およびイベントの管理を行います。これらの API は、WF\_EVENT と呼ばれる PL/SQL パッケージに定義されています。

- 8-268 ページ [「Raise」](#)
- 8-272 ページ [「Send」](#)
- 8-274 ページ [「NewAgent」](#)
- 8-275 ページ [「Test」](#)
- 8-276 ページ [「Enqueue」](#)
- 8-277 ページ [「Listen」](#)

---

**注意：** Oracle Applications embedded Workflow を使用している場合は、「ワークフロー・エージェント・リスナー」コンカレント・プログラムを使用して、受信イベント・メッセージをリスニングできます。8-279 ページの「[「ワークフロー・エージェント・リスナー」コンカレント・プログラム](#)」を参照してください。

---

- 8-280 ページ [「SetErrorInfo」](#)
- 8-281 ページ [「SetDispatchMode」](#)
- 8-282 ページ [「AddParameterToList」](#)
- 8-283 ページ [「AddParameterToListPos」](#)
- 8-284 ページ [「GetValueForParameter」](#)
- 8-285 ページ [「GetValueForParameterPos」](#)

## Raise

### PL/SQL 構文

```
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);
```

### 説明

イベント・マネージャにローカル・イベントを呼び出します。**Raise()** は、このイベント・インスタンスに対して **WF\_EVENT\_T** 構造を作成し、指定されたイベント名、イベント・キー、イベント・データ、パラメータ・リストおよび送信日をその構造に設定します。

イベント・データは、**Raise()** API をコールしたときに、イベント・マネージャに渡されます。つまり、イベント・マネージャがイベント・データを取得するタイミングは、サブスクリプションがイベント・データを必要とするかどうかをチェックし、そのイベントに対してジェネレート関数をコールしたときです。イベント・データがアプリケーションに渡されない場合のパフォーマンスを向上させるには、実行時に常にアプリケーションからイベント・データを生成するのではなく、イベント・データを必要とするサブスクリプションが存在するときにだけ、イベント・マネージャからジェネレート関数を実行してイベント・データを生成します。13-4 ページの「[イベント](#)」および 7-19 ページの「[イベント・データ・ジェネレート関数の標準 API](#)」を参照してください。

送信日には、イベントのサブスクリプション処理が可能になる日時を設定することもできます。送信日が **NULL** の場合、送信日は **Raise()** によって現在のシステム日付に設定されます。送信日をシステム日付より後の日付に設定すると、イベントを遅延することができます。この場合、イベント・メッセージはイベント・マネージャによって標準の **WF\_DEFERRED** キューに格納され、送信日まで **WAIT** 状態になります。送信日になると、イベント・メッセージはデキューできるようになり、エージェント・リスナーが **WF\_DEFERRED** キュー上で実行されたときにデキューされます。

---

**注意：** イベントが呼び出されたときにイベントが遅延されていた場合、**WF\_DEFERRED** キューからデキューされても、イベントの元の「ローカル」ソース・タイプが保持されます。

---

イベントが呼び出されたときにイベントが遅延されていない場合、または遅延されていたイベントが **WF\_DEFERRED** キューからデキューされた場合は、イベント・マネージャによってイベントのサブスクリプション処理が開始されます。イベント・マネージャは、「ローカル」ソース・タイプを持つイベントに対して、ローカル・システム単位に有効なサブスクリプションを検索および実行します。また、「ローカル」ソース・タイプを持つ **Any** イベントに対して、ローカル・システム単位に有効なサブスクリプションを検索および実行します。



発生したイベントに有効なサブスクリプション（Any イベントのサブスクリプション以外）が存在しない場合、Oracle Workflow は、「ローカル」ソース・タイプを持つ Unexpected イベントに対して、ローカル・システム単位に有効なサブスクリプションを実行します。

---

**注意：** イベントが定義されていない場合、イベント・マネージャではエラーは発生しません。

---

イベント・マネージャはサブスクリプションを実行する前に、各サブスクリプションにイベント・データが必要かどうかをチェックします。必要なイベント・データが渡されていない場合、イベント・マネージャはそのイベントに対してジェネレート関数をコールし、イベント・データを生成します。イベント・データを必要とするイベントにジェネレート関数が定義されていない場合、イベント名およびイベント・キーを使用してデフォルトのイベント・データが作成されます。

---

**注意：** Raise() を処理しているときに例外が発生した場合、その例外はトラップされず、Raise() プロシージャをコールしたコードに公開されます。この場合、サブスクリプションとそのルール関数によって、妥当性が検証されます。妥当性検証ロジックを直接コーディングした場合と、同じ結果を得ることができます。

---

## 引数（入力）

<b>p_event_name</b>	イベントの内部名。
<b>p_event_key</b>	プログラムまたはアプリケーション内でイベントが発生したときに、生成される文字列。このイベント・キーにより、イベントの特定のインスタンスが一意に識別されます。
<b>p_event_data</b>	イベントの内容を説明する一連の情報（オプション）。イベント・マネージャはサブスクリプションを実行する前に、各サブスクリプションにイベント・データが必要かどうかをチェックします。必要なイベント・データが渡されていない場合、イベント・マネージャはそのイベントに対してジェネレート関数をコールし、イベント・データを生成します。13-4 ページの「 <a href="#">イベント</a> 」および 7-19 ページの「 <a href="#">イベント・データ・ジェネレート関数の標準 API</a> 」を参照してください。
<b>p_parameters</b>	追加パラメータの名前と値の組合せのリスト（オプション）。
<b>p_send_date</b>	イベントのサブスクリプション処理が可能になる日付（オプション）。

## 例

```
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;
```

```
/*
** 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;
```

**関連項目：**

14-13 ページ [「Any イベント」](#)

14-15 ページ [「Unexpected イベント」](#)

## Send

### PL/SQL 構文

```
procedure Send  
    (p_event in out wf_event_t);
```

### 説明

特定のエージェントから別のエージェントにイベント・メッセージを送信します。イベント・メッセージに送信元エージェントと宛先エージェントの両方が指定されている場合、メッセージは送信元エージェントの送信キューに格納されます。次に、AQ 伝播やエージェントのプロトコルに実装されている伝播によって、宛先エージェントに非同期に送信されます。

イベント・メッセージに宛先エージェントは指定されているが、送信元エージェントが指定されていない場合、そのメッセージは宛先エージェントのキュー・タイプと一致するデフォルトの送信エージェントから送信されます。

イベント・メッセージに送信元エージェントは指定されているが、宛先エージェントが指定されていない場合、そのイベント・メッセージは、受信者を指定しないで送信元エージェントのキューに格納されます。

- 送信元エージェントが複数コンシューマ・キューを使用するときに、サブスクライバ・リストが定義されている場合は、宛先エージェントを省略できます（標準の Workflow キュー・ハンドラは複数コンシューマ・キューでのみ機能します）。この場合、キューのサブスクライバ・リストによって、メッセージをデキューできるコンシューマが決定されます。キューに対してサブスクライバ・リストが定義されていない場合、イベント・メッセージはエラー処理のために WF\_ERROR キューに格納されます。

---

**注意：** Oracle Advanced Queuing が使用する複数コンシューマ・キューのサブスクライバ・リストは、Oracle Workflow ビジネス・イベント・システムのイベント・サブスクリプションとは異なります。詳細は、『Oracle9i アプリケーション開発者ガイド-アドバンスト・キューイング』の「サブスクリプション・リストおよび受信者リスト」を参照してください。

---

- 送信元エージェントが単一コンシューマ・キューを使用するときに、カスタム・キュー・ハンドラが定義されている場合は、宛先エージェントを省略できます。単一コンシューマ・キューの場合、コンシューマを指定する必要はありません。

イベント・メッセージ内の送信日には、コンシューマがメッセージをデキューできる日時を指定します。送信日が空白の場合は、Send() プロシージャによって現在のシステム日付にリセットされ、伝播されたメッセージはすぐにデキューされます。送信日に未来日付が設定されている場合、その日付に対応する遅延時間がメッセージに設定され、遅延時間が経過した

ときにメッセージがデキューされます。詳細は、『Oracle9i アプリケーション開発者ガイド - アドバンスト・キューイング』の「時間指定: 遅延」を参照してください。

---

**注意：** 送信日を使用して宛先エージェントでメッセージがデキューされる日時を指定する場合は、**Send()** がコールされる前にサブスクリプション処理中に送信日を設定する必要があります。

---

**Send()** は、送信された最後のイベント・メッセージ（このプロシージャによって設定されたプロパティを含む）を返します。

## 引数（入力）

**p\_event**                      イベント・メッセージ。

## NewAgent

### PL/SQL 構文

```
function NewAgent  
    (p_agent_guid in raw) return wf_agent_t;
```

### 説明

指定されたエージェントに対して WF\_AGENT\_T 構造を作成し、エージェントのシステムおよび名前をその構造に設定します。8-243 ページの「[エージェント構造](#)」を参照してください。

### 引数（入力）

<b>p_agent_guid</b>	エージェントのグローバル一意識別子。
---------------------	--------------------

## Test

### PL/SQL 構文

```
function Test
  (p_event_name in varchar2) return varchar2;
```

### 説明

指定されたイベントが有効であるかどうかをテストします。さらに、そのイベントを参照する、またはそのイベントが含まれる有効なイベント・グループを参照するローカル・システム単位に、有効なサブスクリプションがあるかどうかをテストします。**Test()** は、次の結果コードを使用して、これらのサブスクリプションの中で最も高いデータ要件を返します。

- **NONE:** イベントを参照する有効なローカル・サブスクリプションがないか、イベントが存在しません。
- **KEY:** イベントを参照する有効なローカル・サブスクリプションが 1 つ以上存在します。ただし、イベント・キーだけを必要とします。
- **MESSAGE:** イベントを参照する有効なローカル・サブスクリプションが 1 つ以上存在します。すべてのイベント・データを必要とします。

### 引数（入力）

**p\_event\_name**                      イベントの内部名。

## Enqueue

### PL/SQL 構文

```
procedure Enqueue
  (p_event in wf_event_t,
   p_out_agent_override in wf_agent_t default null);
```

### 説明

送信エージェントと関連付けられたキューにイベント・メッセージを格納します。オーバーライド・エージェントを指定して、そこにイベント・メッセージをエンキューすることもできます。指定しない場合、イベント・メッセージはメッセージ内に指定されている送信元エージェントにエンキューされます。メッセージ受信者は、イベント・メッセージに指定されている宛先エージェントに設定されます。**Enqueue()** は、送信エージェントのキュー・ハンドラを使用して、メッセージをキューに格納します。

### 引数（入力）

<b>p_event</b>	イベント・メッセージ。
<b>p_out_agent_override</b>	送信エージェント。ここに関連付けられているキューに対して、イベント・メッセージをエンキューします。



## Listen

### PL/SQL 構文

```
procedure Listen  
  (p_agent_name in varchar2);
```

### 説明

受信イベント・メッセージのエージェントを監視し、エージェントのキュー・ハンドラを使用してメッセージをデキューします。

標準の WF\_EVENT\_QH キュー・ハンドラによって、イベント・メッセージがイベント・メッセージの RECEIVE\_DATE 属性にデキューされる日時が設定されます。カスタム・キュー・ハンドラが Dequeue API に組み込まれている場合は、この機能を使用して RECEIVE\_DATE 値を設定することもできます。

イベントがデキューされると、イベント・マネージャは、「外部」ソース・タイプを持つそのイベントの有効なサブスクリプションを、ローカル・システム単位に検索および実行します。また、「外部」ソース・タイプを持つ Any イベントの有効なサブスクリプションを、ローカル・システム単位に検索および実行します。発生したイベントに有効なサブスクリプション (Any イベントのサブスクリプション以外) が存在しない場合、Oracle Workflow は、「外部」ソース・タイプを持つ Unexpected イベントの有効なサブスクリプションを、ローカル・システム単位に実行します。

Listen プロシージャは、エージェントのキューに入っているイベント・メッセージがすべてデキューされると終了します。

---

---

**注意：**「セットアップのチェック」Web 画面を使用して、すべての有効な受信キューおよびエラー・キューに対して Listen プロシージャをスケジューリングすることができます。13-61 ページの「[ローカル・インバウンド・エージェントのリスナーのスケジューリング](#)」を参照してください。

---

---

---

---

**注意：** また、Oracle9i リリース 2 で利用可能なスタンドアロン版の Oracle Workflow を使用している場合は、Oracle Enterprise Manager から利用可能なスタンドアロン版の Oracle Workflow Manager コンポーネントを使用して、ワークフロー・エージェント・リスナーのデータベース・ジョブを管理できます。詳細は、Oracle Workflow Manager のオンライン・ヘルプを参照してください。

---

---

## 引数（入力）

**p\_agent\_name**                      受信エージェントの名前。

### 関連項目：

8-279 ページ [「ワークフロー・エージェント・リスナー」コンカレント・プログラム](#)」

14-13 ページ [「Any イベント」](#)

14-15 ページ [「Unexpected イベント」](#)

16-6 ページ [「wfagtlst.sql」](#)

7-21 ページ [「キュー・ハンドラの標準 API」](#)

## 「ワークフロー・エージェント・リスナー」コンカレント・プログラム

Oracle Applications embedded Workflow を使用している場合は、Listen プロシージャをコンカレント・プログラムとして発行して、受信イベント・メッセージをリスニングできます。ワークフロー・エージェント・リスナーを発行するには、Oracle Applications の「Submit Requests」フォームを使用します。

---

**注意：** Oracle Applications embedded Workflow を使用し、Oracle Applications Manager を実装している場合は、Oracle Workflow Manager を使用して「ワークフロー・エージェント・リスナー」コンカレント・プログラムを発行および管理することができます。詳細は、Oracle Applications Manager のオンライン・ヘルプを参照してください。

---

### ► 受信イベント・メッセージのリスニング

1. Oracle Applications の「Submit Requests」フォームにナビゲートし、「ワークフロー・エージェント・リスナー」コンカレント・プログラムを発行します。Oracle Applications と Oracle Workflow をインストールして設定するときに、システム管理者はこのコンカレント・プログラムを実行する職責の要求セキュリティ・グループに追加する必要があります。このコンカレント・プログラムの実行ファイル名は「Workflow Agent Listener」で、短縮名は「FNDWFLST」です。『Oracle Applications System Administrator's Guide』の「Overview of Concurrent Programs and Requests」を参照してください。
2. 「ワークフロー・エージェント・リスナー」コンカレント・プログラムを要求として発行します。『Oracle Applications User's Guide』の「Submitting a Request」を参照してください。
3. 「パラメータ」ウィンドウに、受信イベント・メッセージを監視するエージェントの名前を入力します。
4. 「OK」を選択して「パラメータ」ウィンドウを閉じます。
5. 印刷オプションと実行オプションを変更してこの要求のスケジュールを定義したら、「発行」を選択して要求を発行します。

#### 関連項目：

8-277 ページ [「Listen」](#)

## SetErrorInfo

### PL/SQL 構文

```
procedure SetErrorInfo
  (p_event in out wf_event_t,
   p_type in varchar2);
```

### 説明

エラー・スタックからエラー情報を取り出し、イベント・メッセージに設定します。エラー・メッセージとエラー・スタックは、イベント・メッセージの対応する属性に設定されます。エラー名とエラー・タイプは、イベント・メッセージの `PARAMETER_LIST` 属性に追加されます。

### 引数（入力）

<b>p_event</b>	イベント・メッセージ。
<b>p_type</b>	エラー・タイプ。「ERROR」または「WARNING」のどちらかを指定します。

## SetDispatchMode

### PL/SQL 構文

```
procedure SetDispatchMode  
    (p_mode in varchar2);
```

### 説明

イベント・マネージャのディスパッチ・モードを遅延または同期サブスクリプション処理に設定します。Raise() をコールする直前に ASYNC モードで SetDispatchMode() をコールすると、呼び出すイベントのサブスクリプション処理がすべて永久に遅延されます。この場合、イベントはイベント・マネージャによって WF\_DEFERRED キューに格納されてから、そのイベントに対するサブスクリプションが実行されます。エージェント・リスナーが実行され、イベントが WF\_DEFERRED キューからデキューされるまで、サブスクリプションは実行されません。

SetDispatchMode() を SYNC モードでコールすると、ディスパッチ・モードの設定を通常の同期サブスクリプション処理に戻すことができます。このモードでは、サブスクリプションがすぐに実行されるか遅延されるかは、各サブスクリプションのフェーズ番号によって決まります。

---

**注意：** サブスクリプション処理を遅延する方法はできるだけ使用しないでください。この方法は、アプリケーション内に遅延をコード化する必要があるため、特別な状況でのみ使用してください。アプリケーションの柔軟性を維持しながらサブスクリプション処理を変更するには、サブスクリプションのフェーズ番号を使用して一部またはすべてのサブスクリプションを遅延します。

---

### 引数（入力）

<b>p_mode</b>	ディスパッチ・モード。遅延（非同期）サブスクリプション処理の場合は「ASYNC」、同期サブスクリプション処理の場合は「SYNC」を入力します。
---------------	---

#### 関連項目：

13-43 ページ [「遅延サブスクリプション処理」](#)

8-268 ページ [「Raise」](#)

## AddParameterToList

### PL/SQL 構文

```
procedure AddParameterToList  
  (p_name in varchar2,  
   p_value in varchar2,  
   p_parameterlist in out wf_parameter_list_t);
```

### 説明

特定のパラメータの名前と値の組合せを、特定のパラメータ・リスト (VARRAY) の最後に追加します。VARRAY が NULL の場合、AddParameterToList() は新しいパラメータを使用して VARRAY を初期化します。

### 引数 (入力)

<b>p_name</b>	パラメータ名。
<b>p_value</b>	パラメータ値。
<b>p_parameterlist</b>	パラメータ・リスト。

## AddParameterToListPos

### PL/SQL 構文

```
procedure AddParameterToListPos
  (p_name in varchar2,
   p_value in varchar2,
   p_position out integer,
   p_parameterlist in out wf_parameter_list_t);
```

### 説明

特定のパラメータの名前と値の組合せを、特定のパラメータ・リスト (VARRAY) の最後に追加します。VARRAY が NULL の場合、AddParameterToListPos() は新しいパラメータを使用して VARRAY を初期化します。また、このプロシージャは、VARRAY 内でパラメータが格納されている位置の索引を戻します。

### 引数 (入力)

<b>p_name</b>	パラメータ名。
<b>p_value</b>	パラメータ値。
<b>p_parameterlist</b>	パラメータ・リスト。

## GetValueForParameter

### PL/SQL 構文

```
function GetValueForParameter  
  (p_name in varchar2,  
   p_parameterlist in wf_parameter_list_t)  
  return varchar2;
```

### 説明

特定のパラメータ・リスト (VARRAY) から特定のパラメータの値を取得します。  
GetValueForParameter() は、パラメータ・リストの終わりから先頭方向に検索します。

### 引数 (入力)

<b>p_name</b>	パラメータ名。
<b>p_parameterlist</b>	パラメータ・リスト。



## GetValueForParameterPos

### PL/SQL 構文

```
function GetValueForParameterPos  
  (p_position in integer,  
   p_parameterlist in wf_parameter_list_t)  
  return varchar2;
```

### 説明

指定したパラメータ・リスト (VARRAY) の特定の位置に格納されているパラメータの値を取得します。

### 引数 (入力)

<b>p_position</b>	パラメータ・リスト内のパラメータの位置を表す索引。
<b>p_parameterlist</b>	パラメータ・リスト。

## イベント・サブスクリプションのルール関数の API

イベント・サブスクリプションのルール関数の API には、イベント・サブスクリプションに割り当てることができる標準のルール関数がいくつか用意されています。ルール関数には、サブスクリプションのトリガー・イベントが発生したときに、Oracle Workflow によって実行される処理が指定されています。

Oracle Workflow には、基本的なサブスクリプション処理を実行する標準の `Default_Rule` 関数が用意されています。この関数は、サブスクリプションに対してルール関数が指定されていない場合に、デフォルトで実行されます。デフォルトのルール関数では、次の処理が行われます。

- ワークフロー・プロセスへのイベント・メッセージの送信（サブスクリプション定義で指定されている場合）
- エージェントへのイベント・メッセージの送信（サブスクリプション定義で指定されている場合）

Oracle Workflow には、標準のルール関数が他にもいくつか用意されています。アプリケーションをテストおよびデバッグするときには、`Log`、`Error`、`Warning` および `Success` 関数を使用できます。`Workflow_Protocol` 関数は、エージェントに送信されるイベント・メッセージをワークフロー・プロセスに渡します。`Error_Rule` 関数は、`Default_Rule` 関数と同じ処理を行いますが、例外を呼び出します。`Workflow_Protocol` および `Error_Rule` 関数は、事前定義済の Oracle Workflow イベント・サブスクリプションで使用されます。

`SetParametersIntoParameterList` 関数は、サブスクリプション・パラメータをイベント・メッセージのパラメータ・リストに追加します。

これらのルール関数 API は、WF\_RULE と呼ばれる PL/SQL パッケージに定義されています。

- [8-287 ページ「Default\\_Rule\(\)」](#)
- [8-289 ページ「Log」](#)
- [8-290 ページ「Error」](#)
- [8-291 ページ「Warning」](#)
- [8-292 ページ「Success」](#)
- [8-293 ページ「Workflow\\_Protocol」](#)
- [8-294 ページ「Error\\_Rule」](#)
- [8-295 ページ「SetParametersIntoParameterList」](#)

### 関連項目：

[13-37 ページ「イベント・サブスクリプション」](#)

[8-286 ページ「イベント・サブスクリプションのルール関数の API」](#)

## Default\_Rule()

### PL/SQL 構文

```
function Default_Rule
  (p_subscription_guid in raw,
   p_event in out wf_event_t) return varchar2;
```

### 説明

イベント・サブスクリプションにルール関数が指定されていないときに、デフォルトのサブスクリプション処理を実行します。次の処理が、デフォルトで実行されます。

- ワークフロー・プロセスへのイベント・メッセージの送信（サブスクリプション定義で指定されている場合）
- エージェントへのイベント・メッセージの送信（サブスクリプション定義で指定されている場合）

これらの操作のどちらかで例外が発生した場合、Default\_Rule() は、その例外をトラップし、エラー情報をイベント・メッセージに格納して、ステータス・コード ERROR を返します。例外が発生しなかった場合は、ステータス・コード SUCCESS を返します。

---

**注意：** イベント・メッセージが「デフォルト・イベント・エラー」ワークフロー・プロセスに送信される場合、Default\_Rule() は、プロセスの項目キーとして関連 ID を新しく生成し、項目キーが一意になるようにします。

---

イベント・メッセージに対してカスタム・ルール関数を実行してから、イベント・メッセージを送信する場合は、カスタム・ルール関数を使用するサブスクリプションを下位のフェーズ番号で定義し、イベントを送信するデフォルト・ルール関数を使用するサブスクリプションを上位のフェーズ番号で定義します。

たとえば、次の手順で行います。

1. カスタム・ルール関数とフェーズ番号 10 を使用して、対象となるイベントへのサブスクリプションを定義します。
2. ルール関数 WF\_EVENT.Default\_Rule とフェーズ番号 20 を使用して、そのイベントへのサブスクリプションをもう 1 つ定義し、イベントの送信先となるワークフローまたはエージェントを指定します。
3. イベントを呼び出して、それらのサブスクリプションをトリガーします。まず、フェーズ番号が下位のサブスクリプションが実行され、イベント・メッセージに対してカスタム・ルール関数が実行されます。イベントが 2 番目のサブスクリプションに渡されると、変更されたイベント・メッセージが指定のワークフローまたはエージェントに送信されます。

Default\_Rule() をコールして、カスタム・ルール関数内にデフォルトの送信処理を追加することもできます。サブスクリプションに対して Default\_Rule() 以外のルール関数を入力した場合は、サブスクリプションに指定されたワークフローおよびエージェントに対して、イベント・メッセージは自動的に送信されません。このサブスクリプションからメッセージを送信する場合は、送信処理をカスタム・ルール関数に明示的に組み込む必要があります。Default\_Rule() をコールして組み込むこともできます。7-24 ページの「[イベント・サブスクリプションのルール関数の標準 API](#)」を参照してください。

---

---

**注意：** 再利用できない複雑で特別なルール関数を作成するのではなく、再利用できる単純なルール関数を作成して複数のサブスクリプションをイベントに定義することをお勧めします。

---

---

引数（入力）

- |                     |                       |
|---------------------|-----------------------|
| p_subscription_guid | サブスクリプションのグローバル一意識別子。 |
| p_event             | イベント・メッセージ。           |

## Log

### PL/SQL 構文

```
function Log
  (p_subscription_guid in raw,
   p_event in out wf_event_t) return varchar2;
```

### 説明

DBMS\_OUTPUT.put\_line を使用して、指定されたイベント・メッセージの内容をログに記録し、ステータス・コード SUCCESS を返します。この関数を使用して、イベント・メッセージの内容を SQL\*Plus セッションに出力し、テストおよびデバッグに使用できます。

たとえば、イベント・メッセージを変更するカスタム・ルール関数をテストする場合は、Log() を使用してカスタム・ルール関数の実行前と実行後にイベント・メッセージを表示できます。対象となるイベントに対して、次の 3 つのサブスクリプションを定義します。

- フェーズ番号 10 とルール関数 WF\_RULE.Log を使用するサブスクリプションを定義します。
- フェーズ番号 20 とカスタム・ルール関数を使用するサブスクリプションを定義します。
- フェーズ番号 30 とルール関数 WF\_RULE.Log を使用するサブスクリプションを定義します。

次に、SQL\*Plus に接続します。次のコマンドを実行します。

```
set serveroutput on size 100000
```

次に、WF\_EVENT.Raise を使用してイベントを呼び出します。イベント・マネージャがフェーズ番号の順にイベントへのサブスクリプションを実行し、カスタム・ルール関数の実行前と実行後にイベント・メッセージの内容が表示されます。

---

**注意：** Oracle Workflow の本番インスタンスで使用するサブスクリプションに対して、Log() ルール関数を割り当てないでください。この関数は、デバッグにのみ使用してください。

---

### 引数（入力）

p_subscription_guid	サブスクリプションのグローバル一意識別子。
p_event	イベント・メッセージ。

Error

PL/SQL 構文

```
function Error
(p_subscription_guid in raw,
 p_event in out wf_event_t) return varchar2;
```

説明

ステータス・コード ERROR を返します。また、この関数をサブスクリプションのルール関数として割り当てるときは、エラー・メッセージの内部名を表すテキスト文字列をサブスクリプションの「パラメータ」フィールドに入力する必要があります。サブスクリプションが実行されると、Error() は、setErrorMessage() を使用してそのエラー・メッセージをイベント・メッセージに格納します。8-259 ページの「setErrorMessage」を参照してください。

「パラメータ」フィールドに入力するテキスト文字列は、有効な Oracle Workflow エラー・メッセージ名でなければなりません。Oracle Workflow から提供されるエラー・メッセージの名前は、WFERR タイプのメッセージとして、WF\_RESOURCES 表の NAME 列に格納されています。

特定のイベントが発生するたびに、事前定義済の Workflow エラー・メッセージを含むエラー通知をシステム管理者に送信する場合は、Error() をサブスクリプションのルール関数として使用できます。

たとえば、対象となるイベントへのサブスクリプションを定義し、ルール関数 WF\_RULE.Error を指定し、その「パラメータ」フィールドに WFSQL\_ARGS と入力します。次に、そのイベントを呼び出し、サブスクリプションをトリガーします。Error() がステータス・コード ERROR を返すため、イベント・マネージャはイベント・メッセージを WF\_ERROR キューに格納し、イベントのサブスクリプション処理が停止します。WF\_ERROR キュー上でリスナーが動作しているときは、エラー通知がシステム管理者に送信されます。このエラー通知には、「引数に無効な値が渡されました」というメッセージが含まれます。これが、WFSQL\_ARGS エラー・メッセージの表示名になります。

---

---

**注意：** コール元のアプリケーションが正常に処理を完了した場合、Error() はそのアプリケーションに対して例外を呼び出しません。

---

---

引数（入力）

p_subscription_guid	サブスクリプションのグローバル意識別子。
p_event	イベント・メッセージ。

## Warning

### PL/SQL 構文

```
function Warning
(p_subscription_guid in raw,
 p_event in out wf_event_t) return varchar2;
```

### 説明

ステータス・コード **WARNING** を返します。また、この関数をサブスクリプションのルール関数として割り当てるときは、エラー・メッセージの内部名を表すテキスト文字列をサブスクリプションの「パラメータ」フィールドに入力する必要があります。サブスクリプションが実行されると、**Warning()** は、**setErrorMessage()** を使用してそのエラー・メッセージをイベント・メッセージに格納します。8-259 ページの「[setErrorMessage](#)」を参照してください。

「パラメータ」フィールドに入力するテキスト文字列は、有効な Oracle Workflow エラー・メッセージ名でなければなりません。Oracle Workflow から提供されるエラー・メッセージの名前は、WFERR タイプのメッセージとして、WF\_RESOURCES 表の NAME 列に格納されています。

特定のイベントが発生するたびに、事前定義済の Workflow エラー・メッセージを含む警告通知をシステム管理者に送信する場合は、**Warning()** をサブスクリプションのルール関数として使用できます。

たとえば、対象となるイベントへのサブスクリプションを定義し、ルール関数 **WF\_RULE.Warning** を指定し、その「パラメータ」フィールドに **WFSQL\_ARGS** と入力します。次に、そのイベントを呼び出し、サブスクリプションをトリガーします。**Warning()** がステータス・コード **ERROR** を返すため、イベント・マネージャはイベント・メッセージを **WF\_ERROR** キューに格納します。ただし、イベントのサブスクリプション処理は続行します。**WF\_ERROR** キュー上でリスナーが動作しているときは、警告通知がシステム管理者に送信されます。このエラー通知には、「引数に無効な値が渡されました」というメッセージが含まれます。これが、**WFSQL\_ARGS** エラー・メッセージの表示名になります。

---

---

**注意：** コール元のアプリケーションが正常に処理を完了した場合、**Warning()** はそのアプリケーションに対して例外を呼び出しません。

---

---

### 引数（入力）

<b>p_subscription_guid</b>	サブスクリプションのグローバル一意識別子。
<b>p_event</b>	イベント・メッセージ。

## Success

### PL/SQL 構文

```
function Success
  (p_subscription_guid in raw,
   p_event in out wf_event_t) return varchar2;
```

### 説明

ステータス・コード **SUCCESS** を返します。この関数は、キューからイベント・メッセージを削除しますが、コール側サブスクリプションに **SUCCESS** ステータス・コードを返す以外のコードは実行しません。

ステータス・コード **Success** は、ビジネス・イベント・システムを使用するコードを開発しているときに、テストおよびデバッグの目的で使用できます。たとえば、同一イベントへの複数のサブスクリプションをデバッグする場合に、いずれかのサブスクリプションのルール関数を **WF\_RULE.Success** に置き換え、サブスクリプションのその他の詳細はそのままにします。サブスクリプションを実行すると、**SUCCESS** が返されますが、他のサブスクリプション処理は実行されません。このメソッドを使用すると、問題のあるサブスクリプションを簡単に特定できます。

**Success()** は、標準の「Noop」アクティビティで使用する **WF\_STANDARD.Noop** プロシージャに似ています。

### 引数（入力）

<b>p_subscription_guid</b>	サブスクリプションのグローバル一意識別子。
<b>p_event</b>	イベント・メッセージ。



## Workflow\_Protocol

### PL/SQL 構文

```
function Workflow_Protocol
  (p_subscription_guid in raw,
   p_event in out wf_event_t) return varchar2;
```

### 説明

サブスクリプションに指定されたワークフロー・プロセスにイベント・メッセージを送信します。ワークフロー・プロセスは、サブスクリプションに指定された受信エージェントにイベント・メッセージを送信します。

---

---

**注意：** Workflow\_Protocol() 自体は、イベント・メッセージを受信エージェントに送信しません。この関数は、イベント・メッセージをワークフロー・プロセスに送信するだけです。ワークフロー・プロセスでは、イベント・メッセージを指定されたエージェントに送信する処理をモデル化できます。

---

---

サブスクリプションに送信エージェントも指定されている場合は、ワークフロー・プロセスによってイベント・メッセージが送信エージェントのキューに格納され、受信エージェントに伝播されます。送信エージェントが指定されていない場合は、デフォルトの送信エージェントが選択されます。

サブスクリプションのパラメータにパラメータの名前と値のペア ACKREQ=Y が指定されている場合、ワークフロー・プロセスはイベント・メッセージを送信した後で受信確認の受信を待機します。

ワークフロー・プロセスが例外を発生した場合、Workflow\_Protocol() はエラー情報をイベント・メッセージに格納し、ステータス・コード ERROR を返します。例外が発生しなかった場合は、ステータス・コード SUCCESS を返します。

Workflow\_Protocol() は、ワークフロー送信プロトコルおよびイベント・システムのデモ・イベントに事前定義されているいくつかのサブスクリプションの、ルール関数として使用されます。14-20 ページの「[ワークフロー送信プロトコル](#)」および 15-62 ページの「[イベント・システム・デモンストレーション](#)」を参照してください。

### 引数（入力）

<b>p_subscription_guid</b>	サブスクリプションのグローバル一意識別子。
<b>p_event</b>	イベント・メッセージ。

## Error\_Rule

### PL/SQL 構文

```
function Error_Rule
  (p_subscription_guid in raw,
   p_event in out wf_event_t) return varchar2;
```

### 説明

Default\_Rule() と同じ、次のサブスクリプション処理を実行します。

- ワークフロー・プロセスへのイベント・メッセージの送信（サブスクリプション定義で指定されている場合）
- エージェントへのイベント・メッセージの送信（サブスクリプション定義で指定されている場合）

ただし、これらの処理のどちらかで例外が発生した場合、**Error\_Rule()** は例外を再度呼び出すため、イベントが **WF\_ERROR** キューに戻されることはありません。例外が発生しなかった場合は、ステータス・コード **SUCCESS** を返します。

**Error\_Rule()** は、**Unexpected** イベントおよびソース・タイプが「エラー」の **Any** イベントへの事前定義済サブスクリプションの、ルール関数として使用されます。事前定義済のサブスクリプションには、「システム:エラー」項目タイプの「デフォルト・イベント・エラー・プロセス」にイベントが送信されるように指定されています。

また、独自のエラー・サブスクリプションで、このルール関数を使用することもできます。エラー・サブスクリプションのルール関数として **WF\_RULE.Error** を入力し、そのサブスクリプションを起動させるワークフローの項目タイプとプロセスを指定します。

### 引数（入力）

<b>p_subscription_guid</b>	サブスクリプションのグローバル一意識別子。
<b>p_event</b>	イベント・メッセージ。

#### 関連項目：

14-15 ページ [「Unexpected イベント」](#)

14-13 ページ [「Any イベント」](#)

## SetParametersIntoParameterList

### PL/SQL 構文

```
function SetParametersIntoParameterList
(p_subscription_guid in raw,
 p_event in out wf_event_t) return varchar2;
```

### 説明

サブスクリプション・パラメータのパラメータ名前とパラメータ値の組合せをイベント・メッセージの **PARAMETER\_LIST** 属性に設定します。ただし、**ITEMKEY** および **CORRELATION\_ID** という名前のパラメータを除きます。これらの名前を持つパラメータには、イベント・メッセージの **CORRELATION\_ID** 属性がパラメータ値に設定されます。

これらの操作で例外が発生した場合、**SetParametersIntoParameterList()** はエラー情報をイベント・メッセージに格納し、ステータス・コード **ERROR** を返します。例外が発生しなかった場合は、ステータス・コード **SUCCESS** を返します。

**SetParametersIntoParameterList()** をフェーズ番号が下位のサブスクリプションのルール関数として使用すると、サブスクリプションの事前定義済パラメータをイベント・メッセージに追加できます。これにより、フェーズ番号が上位の後続のサブスクリプションは、イベント・メッセージ内のそれらのパラメータにアクセスできます。

### 引数（入力）

<b>p_subscription_guid</b>	サブスクリプションのグローバル一意識別子。
<b>p_event</b>	イベント・メッセージ。

#### 関連項目：

8-248 ページ [「イベント・メッセージ構造」](#)

## イベント関数の API

イベント関数 API は、アプリケーション・プログラム、イベント・マネージャまたはワークフロー・プロセスによって実行時にコールされるユーティリティ関数で、ビジネス・イベント・システムとの通信およびイベントの管理を行います。イベント関数 API は、WF\_EVENT\_FUNCTIONS\_PKG と呼ばれる PL/SQL パッケージに定義されています。

- 8-297 ページ [「Parameters」](#)
- 8-299 ページ [「SubscriptionParameters」](#)
- 8-300 ページ [「AddCorrelation」](#)
- 8-302 ページ [「Generate」](#)
- 8-304 ページ [「Receive」](#)

## Parameters

### PL/SQL 構文

```
function Parameters
    (p_string in varchar2,
    p_numvalues in number,
    p_separator in varchar2) return t_parameters;
```

### 説明

特定の区切り文字で区切られた、一定数のパラメータが含まれるテキスト文字列を解析します。Parameters() は、T\_PARAMETERS 複合データ型を使用した VARRAY で、解析済パラメータを返します。T\_PARAMETERS 複合データ型は、WF\_EVENT\_FUNCTIONS\_PKG パッケージに定義されています。次の表で、T\_PARAMETERS データ型を説明します。

表 8-11

データ型の名前	要素のデータ型の定義
T_PARAMETERS	VARCHAR2(240)

Parameters() は、ジェネレート関数からコールできる汎用ユーティリティです。イベント・キーが複数の値で構成され、事前定義の文字で連結されているときに使用します。この関数を使用すると、イベント・キーがいくつかの要素値に分解されます。

### 引数（入力）

<b>p_string</b>	連結したパラメータを含むテキスト文字列。
<b>p_numvalues</b>	文字列に含まれているパラメータの数。
<b>p_separator</b>	文字列内のパラメータの区切り文字。

### 例

```
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 構文

```
function SubscriptionParameters
  (p_string in varchar2,
   p_key in varchar2) return varchar2;
```

### 説明

イベント・サブスクリプションに定義されたパラメータを含むテキスト文字列から、指定されたパラメータの値を返します。テキスト文字列内のパラメータの名前と値のペアは、空白で区切られ、次の形式で指定されていなければなりません。

```
<name1>=<value1> <name2>=<value2> ... <nameN>=<valueN>
```

SubscriptionParameters() は、テキスト文字列から指定されたパラメータ名を検索し、その名前に割り当てられている値を返します。たとえば、サブスクリプションのルール関数からこの関数をコールして、サブスクリプションのパラメータの値を取得し、ルール関数に対してその値に基づいた別の処理をコーディングできます。

### 引数（入力）

<b>p_string</b>	イベント・サブスクリプションに定義されたパラメータを含むテキスト文字列。
<b>p_key</b>	値を取得するパラメータの名前。

### 例

次の例では、SubscriptionParameters() を使用して、ITEMKEY サブスクリプション・パラメータの値を l\_function プログラム変数に割り当てます。このコード例は、AddCorrelation 関数の一部で、サブスクリプションを処理しているときに相関 ID をイベント・メッセージに追加します。8-300 ページの「[AddCorrelation](#)」を参照してください。

```
...
--
-- 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 構文

```
function AddCorrelation
(p_subscription_guid in raw,
p_event in out wf_event_t) return varchar2;
```

### 説明

サブスクリプションを処理しているときに、**相関 ID** をイベント・メッセージに追加します。**AddCorrelation()** は、サブスクリプションのパラメータから、**ITEMKEY** という名前のパラメータを検索します。このパラメータには、イベント・メッセージの**相関 ID** を生成するカスタム関数を指定します。この関数は、サブスクリプションの「パラメータ」フィールドに次の形式で指定する必要があります。

**ITEMKEY**=<package\_name.function\_name>

**AddCorrelation()** は、**SubscriptionParameters()** を使用して **ITEMKEY** パラメータの値を検索および取得します。8-299 ページの「**SubscriptionParameters**」を参照してください。

**相関 ID** カスタム関数が **ITEMKEY** パラメータに指定されている場合、**AddCorrelation()** は、その関数を実行し、その関数から返された値に**相関 ID** を設定します。カスタム関数が指定されていない場合、**相関 ID** はシステム日付に設定されます。イベント・メッセージがワークフロー・プロセスに送信されると、ワークフロー・エンジンはその**相関 ID** を項目キーとして使用して、プロセス・インスタンスを識別します。

**AddCorrelation()** で例外が発生した場合は、ステータス・コード **ERROR** が返されます。例外が発生しなかった場合は、ステータス・コード **SUCCESS** が返されます。

**AddCorrelation()** は、イベント・サブスクリプションのルール関数の標準 API に従って定義します。**AddCorrelation()** をサブスクリプションのルール関数として使用して**相関 ID** をイベントに追加するときに、下位のフェーズ番号を割り当てれば、フェーズ番号が上位のサブスクリプションを続けて実行することができます。

たとえば、次の手順で行います。

1. ルール関数 **WF\_EVENT\_FUNCTIONS\_PKG.AddCorrelation** とフェーズ番号 10 を使用して、対象となるイベントへのサブスクリプションを定義します。そのサブスクリプションの「パラメータ」フィールドにパラメータの名前と値のペア **ITEMKEY**=<package\_name.function\_name> を入力します。  
<package\_name.function\_name> は、**相関 ID** を生成するパッケージおよび関数に置き換えます。
2. フェーズ番号 20 を使用して、そのイベントへの別のサブスクリプションを定義します。実行する処理として、カスタム・ルール関数またはワークフローの項目タイプとプロセス、あるいはその両方を入力します。



3. イベントを呼び出して、それらのサブスクリプションをトリガーします。最初に、フェーズ番号が下位のサブスクリプションが実行され、相関 ID がイベント・メッセージに追加されます。イベントが 2 番目のサブスクリプションに渡されると、その相関 ID が項目キーとして使用されます。

また、AddCorrelation() をカスタム・ルール関数からコールして、カスタム関数の処理として相関 ID を追加することもできます。7-24 ページの「[イベント・サブスクリプションのルール関数の標準 API](#)」を参照してください。

---

**注意：** 再利用できない複雑で特別なルール関数を作成するのではなく、再利用できる単純なルール関数を作成して複数のサブスクリプションをイベントに定義することをお薦めします。

---

## 引数（入力）

<code>p_subscription_guid</code>	サブスクリプションのグローバル一意識別子。
<code>p_event</code>	イベント・メッセージ。

## Generate

### PL/SQL 構文

```
function Generate
  (p_event_name in varchar2,
   p_event_key in varchar2) return clob;
```

### 説明

シード・イベント・グループのイベントに対してイベント・データを生成します。このイベント・データには、システム間でオブジェクトをレプリケートするときに使用される、ビジネス・イベント・システムのオブジェクト定義が含まれています。

シード・イベント・グループは、次のイベントで構成されます。

- 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

イベント、イベント・グループ、システム、エージェントおよびサブスクリプションに定義されたイベントの場合、WF\_EVENT\_FUNCTIONS\_PKG.Generate() は、対応する表に関連付けられた Generate API をコールして、イベント・データの XML 文書を生成します。同期イベント・システム・イベントの場合、WF\_EVENT\_FUNCTIONS\_PKG.Generate() は、イベント、イベント・グループ、システム、エージェントおよびサブスクリプションに定義さ

れたイベントがすべて含まれる XML 文書を、ローカル・システム上のイベント・マネージャから生成します。

## 引数（入力）

<b>p_event_name</b>	イベントの内部名。
<b>p_event_key</b>	プログラムまたはアプリケーション内でイベントが発生したときに、生成される文字列。このイベント・キーにより、イベントの特定のインスタンスが一意に識別されます。

### 関連項目：

- 8-309 ページ [「WF\\_EVENTS\\_PKG.Generate」](#)
- 8-312 ページ [「WF\\_EVENT\\_GROUPS\\_PKG.Generate」](#)
- 8-315 ページ [「WF\\_SYSTEMS\\_PKG.Generate」](#)
- 8-318 ページ [「WF\\_AGENTS\\_PKG.Generate」](#)
- 8-321 ページ [「WF\\_EVENT\\_SUBSCRIPTIONS\\_PKG.Generate」](#)
- 14-2 ページ [「事前定義済ワークフロー・イベント」](#)

## Receive

### PL/SQL 構文

```
function Receive
(p_subscription_guid in raw,
 p_event in out wf_event_t) return varchar2;
```

### 説明

サブスクリプションを処理しているときにビジネス・イベント・システムのオブジェクト定義を受信し、該当するビジネス・イベント・システムの表にロードします。この関数は、システム間でオブジェクトのレプリケーションを行います。

WF\_EVENT\_FUNCTIONS\_PKG.Receive() は、イベント・サブスクリプションのルール関数の標準 API に従って定義されています。WF\_EVENT\_FUNCTIONS\_PKG.Receive() は、2 つの事前定義済サブスクリプションのルール関数として使用されます。一方のサブスクリプションは、システムのサインアップ・イベントがローカルで発生したときにトリガーされます。もう一方のサブスクリプションは、シード・イベント・グループのいずれかのイベントが外部ソースから着信したときにトリガーされます。

シード・イベント・グループは、次のイベントで構成されます。

- 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() は、着信したイベント・メッセージのイベント・データに含まれる XML 文書を解析し、ビジネス・イベント・システムのオブジェクト定義をロードして該当する表に格納します。

---

**注意：** イベント、イベント・グループ、システム、エージェントおよびサブスクリプションに定義されたイベントの場合、WF\_EVENT\_FUNCTIONS\_PKG.Receive() は、対応する表に関連付けられた Receive API をコールし、その XML 文書を解析し、その定義を表にロードします。

---

## 引数（入力）

p_subscription_guid	サブスクリプションのグローバル一意識別子。
p_event	イベント・メッセージ。

### 関連項目：

8-310 ページ [「WF\\_EVENTS\\_PKG.Receive」](#)  
8-313 ページ [「WF\\_EVENT\\_GROUPS\\_PKG.Receive」](#)  
8-316 ページ [「WF\\_SYSTEMS\\_PKG.Receive」](#)  
8-319 ページ [「WF\\_AGENTS\\_PKG.Receive」](#)  
8-322 ページ [「WF\\_EVENT\\_SUBSCRIPTIONS\\_PKG.Receive」](#)  
14-2 ページ [「事前定義済ワークフロー・イベント」](#)

## ビジネス・イベント・システムのレプリケーションの API

次の API をコールすると、システム間でビジネス・イベント・システムのデータをレプリケートできます。レプリケーション API は、次の PL/SQL パッケージに格納されています。各パッケージは、ビジネス・イベント・システムの表に対応しています。Oracle Workflow には、表ごとにジェネレート関数および Receive 関数が提供されます。

- WF\_EVENTS\_PKG
  - 8-309 ページ [「WF\\_EVENTS\\_PKG.Generate」](#)
  - 8-310 ページ [「WF\\_EVENTS\\_PKG.Receive」](#)
- WF\_EVENT\_GROUPS\_PKG
  - 8-312 ページ [「WF\\_EVENT\\_GROUPS\\_PKG.Generate」](#)
  - 8-313 ページ [「WF\\_EVENT\\_GROUPS\\_PKG.Receive」](#)
- WF\_SYSTEMS\_PKG
  - 8-315 ページ [「WF\\_SYSTEMS\\_PKG.Generate」](#)
  - 8-316 ページ [「WF\\_SYSTEMS\\_PKG.Receive」](#)
- WF\_AGENTS\_PKG
  - 8-318 ページ [「WF\\_AGENTS\\_PKG.Generate」](#)
  - 8-319 ページ [「WF\\_AGENTS\\_PKG.Receive」](#)
- WF\_EVENT\_SUBSCRIPTIONS\_PKG
  - 8-321 ページ [「WF\\_EVENT\\_SUBSCRIPTIONS\\_PKG.Generate」](#)
  - 8-322 ページ [「WF\\_EVENT\\_SUBSCRIPTIONS\\_PKG.Receive」](#)

各 Generate API は、指定されたビジネス・イベント・システムのオブジェクト定義に対応する表から、必要な情報がすべて含まれる XML メッセージを生成します。対応する Receive API は、その XML メッセージを解析し、その行をロードして該当する表に格納します。

これらの API は、ビジネス・イベント・システムのデータが自動レプリケートされるときに使用されます。Generate API は WF\_EVENT\_FUNCTIONS\_PKG.Generate() からコールされ、Receive API は WF\_EVENT\_FUNCTIONS\_PKG.Receive() からコールされます。8-302 ページの [「Generate」](#) および 8-304 ページの [「Receive」](#) を参照してください。

### ドキュメント・タイプ定義

ワークフロー表の XML メッセージのドキュメント・タイプ定義 (DTD) は、マスター・タグ WF\_TABLE\_DATA の下に定義されています。各 DTD のこのマスター・タグの下には、DTD が適用されるワークフロー表の名前を識別するタグがあり、そのタグの下にバージョン・タグと表の各列のタグがあります。DTD の構造の例を示します。

```
<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>
```

ビジネス・イベント・システムのレプリケーションの API では、次の DTD が使用されます。

- 8-308 ページ [「WF\\_EVENTS ドキュメント・タイプ定義」](#)
- 8-311 ページ [「WF\\_EVENT\\_GROUPS ドキュメント・タイプ定義」](#)
- 8-314 ページ [「WF\\_SYSTEMS ドキュメント・タイプ定義」](#)
- 8-317 ページ [「WF\\_AGENTS ドキュメント・タイプ定義」](#)
- 8-320 ページ [「WF\\_EVENT\\_SUBSCRIPTIONS ドキュメント・タイプ定義」](#)

## WF\_EVENTS ドキュメント・タイプ定義

次のドキュメント・タイプ定義（DTD）は、XML メッセージに必要な構造を示しており、WF\_EVENTS 表のイベント定義に関するすべての情報を含んでいます。

```
<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.Generate

### PL/SQL 構文

```
function Generate  
  (x_guid in raw)  
  return varchar2;
```

### 説明

指定されたイベント定義の WF\_EVENTS 表から、すべての情報が含まれる XML メッセージを生成します。

### 引数（入力）

<b>x_guid</b>	イベントのグローバル一意識別子。
---------------	------------------

## WF\_EVENTS\_PKG.Receive

### PL/SQL 構文

```
procedure Receive  
  (x_message in varchar2);
```

### 説明

イベント定義に関するすべての情報が含まれる XML メッセージを受信し、その情報をロードして WF\_EVENTS 表に格納します。

### 引数（入力）

<b>x_message</b>	イベント定義に関するすべての情報を含む XML メッセージ。
------------------	--------------------------------

## WF\_EVENT\_GROUPS ドキュメント・タイプ定義

次のドキュメント・タイプ定義（DTD）は、XML メッセージに必要な構造を示しており、WF\_EVENT\_GROUPS 表のイベント・グループ・メンバー定義に関するすべての情報を含んでいます。

---

---

**注意：** イベント・グループのヘッダー情報には、個別のイベントと同様に、WF\_EVENTS 表に定義します。ただし、イベント・グループ・メンバーの定義は、WF\_EVENT\_GROUPS 表に格納します。

---

---

```
<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 構文

```
function Generate
  (x_group_guid in raw,
   x_member_guid in raw)
  return varchar2;
```

### 説明

指定されたイベント・グループ・メンバー定義の WF\_EVENT\_GROUPS 表から、すべての情報が含まれる XML メッセージを生成します。

### 引数（入力）

<b>x_group_guid</b>	イベント・グループのグローバル一意識別子。
<b>x_member_guid</b>	個々のメンバー・イベントのグローバル一意識別子。

## WF\_EVENT\_GROUPS\_PKG.Receive

### PL/SQL 構文

```
procedure Receive  
  (x_message in varchar2);
```

### 説明

イベント・グループ・メンバー定義に関するすべての情報が含まれる XML メッセージを受信し、その情報をロードして WF\_EVENT\_GROUPS 表に格納します。

### 引数（入力）

<b>x_message</b>	イベント・グループ・メンバー定義に関するすべての情報が含まれる XML メッセージ。
------------------	--

## WF\_SYSTEMS ドキュメント・タイプ定義

次のドキュメント・タイプ定義（DTD）は、XML メッセージに必要な構造を示しており、WF\_SYSTEMS 表のシステム定義に関するすべての情報を含んでいます。

```
<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\_SYSTEMS\_PKG.Generate

### PL/SQL 構文

```
function Generate  
    (x_guid in raw)  
    return varchar2;
```

### 説明

指定されたシステム定義の WF\_SYSTEMS 表から、すべての情報が含まれる XML メッセージを生成します。

### 引数（入力）

<b>x_guid</b>	システムのグローバル一意識別子。
---------------	------------------

## WF\_SYSTEMS\_PKG.Receive

### PL/SQL 構文

```
procedure Receive  
  (x_message in varchar2);
```

### 説明

システム定義に関するすべての情報が含まれる XML メッセージを受信し、その情報をロードして WF\_SYSTEMS 表に格納します。

### 引数（入力）

<b>x_message</b>	システム定義に関するすべての情報が含まれる XML メッセージ。
------------------	----------------------------------



## WF\_AGENTS ドキュメント・タイプ定義

次のドキュメント・タイプ定義（DTD）は、XML メッセージに必要な構造を示しており、WF\_AGENTS 表のエージェント定義に関するすべての情報を含んでいます。

```
<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\_AGENTS\_PKG.Generate

### PL/SQL 構文

```
function Generate  
  (x_guid in raw)  
  return varchar2;
```

### 説明

指定されたエージェント定義の WF\_AGENTS 表から、すべての情報が含まれる XML メッセージを生成します。

### 引数（入力）

<b>x_guid</b>	エージェントのグローバル一意識別子。
---------------	--------------------

## WF\_AGENTS\_PKG.Receive

### PL/SQL 構文

```
procedure Receive  
    (x_message in varchar2);
```

### 説明

エージェント定義に関するすべての情報が含まれる XML メッセージを受信し、その情報をロードして WF\_AGENTS 表に格納します。

### 引数（入力）

<b>x_message</b>	エージェント定義に関するすべての情報が含まれる XML メッセージ。
------------------	------------------------------------

## WF\_EVENT\_SUBSCRIPTIONS ドキュメント・タイプ定義

次のドキュメント・タイプ定義（DTD）は、XML メッセージに必要な構造を示しており、WF\_EVENT\_SUBSCRIPTIONS 表のイベント・サブスクリプション定義に関するすべての情報を含んでいます。

```
<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>
```

## WF\_EVENT\_SUBSCRIPTIONS\_PKG.Generate

### PL/SQL 構文

```
function Generate  
  (x_guid in raw)  
  return varchar2;
```

### 説明

指定されたイベント・サブスクリプション定義の WF\_EVENT\_SUBSCRIPTIONS 表から、すべての情報が含まれる XML メッセージを生成します。

### 引数（入力）

<b>x_guid</b>	イベント・サブスクリプションのグローバル一意識別子。
---------------	----------------------------

## WF\_EVENT\_SUBSCRIPTIONS\_PKG.Receive

### PL/SQL 構文

```
procedure Receive  
  (x_message in varchar2);
```

### 説明

イベント・サブスクリプション定義に関するすべての情報が含まれる XML メッセージを受信し、その情報をロードして WF\_EVENT\_SUBSCRIPTIONS 表に格納します。

### 引数（入力）

<b>x_message</b>	イベント・サブスクリプション定義に関するすべての情報が含まれる XML メッセージ。
------------------	--

---

## Oracle Workflow ホーム・ページ

この章では、Oracle Workflow ホーム・ページについて説明します。ユーザーと管理者は、このホーム・ページから Oracle Workflow のすべての Web ベース機能に一元的にアクセスできます。

## Oracle Workflow ホーム・ページへのアクセス

Oracle Workflow ホーム・ページを使用して、Oracle Workflow のすべての Web ベース機能にリンクします。各機能へのアクセスはこのページに集約されているため、個々の URL を覚える必要はありません。

---

---

**注意：** Oracle Workflow のインストールに Oracle Internet Directory/Single Sign-On 統合が実装されている場合は、Oracle Workflow の Web ベース機能へのアクセス時にシングル・サインオンを使用できます。シングル・サインオンでは、他の参加コンポーネントにアクセスしており、再度ログインする必要がない場合、参加中の Oracle9iAS コンポーネントにログインしているユーザーは自動的に認証されます。2-29 ページの「[手順 WF-4 Workflow ディレクトリ・サービスと Oracle Internet Directory の同期](#)」を参照してください。

---

---

### ► Oracle Workflow ホーム・ページへのアクセス

1. Web ブラウザを使用して、次の URL でホーム・ページに接続します。

`<webagent>/wfa_html.home`

`<webagent>` は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

---

---

**注意：** このページにはセキュリティが適用されるため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。

---

---





2. Web 画面に、Oracle Workflow の現行バージョンが示されます。
3. Oracle Workflow の他の Web 画面と同様に、Oracle Workflow ホーム・ページの左上隅にはツールバーが表示されます。「ホーム」アイコンを選択すると、Oracle Workflow ホーム・ページに戻ります。現行ページの名前がツールバーの中央に表示されます。「ログアウト」アイコンを選択すると、Oracle Workflow の現行 Web セッションからログアウトし、「ヘルプ」アイコンを選択すると、現行画面のオンライン・ヘルプが表示されます。一部の「イベント・マネージャ」Web 画面には「問合せ」アイコンもあります。このアイコンを選択して問合せの詳細を入力すれば、イベント・マネージャ・オブジェクトを検索できます。
4. 「ワークリスト」リンクを選択し、自分のワークフロー通知リストを再表示します。自分の通知は「ワークリスト」から直接閉じたり再割当てすることができます。また、個々の通知の詳細にドリルダウンして、各通知を閉じるか再割当てしたり、各通知に対して応答することもできます。10-17 ページの「ワークリストの通知の表示」を参照してください。
5. 「通知の検索」リンクを選択し、特定の基準と一致する通知を検索して処理します。10-15 ページの「通知の検索」を参照してください。

6. 「通知ルール」リンクを選択し、自動通知ルーティング・ルールを表示して定義します。ワークフロー管理者権限を持つロールでログインしている場合は、「自動通知処理ルールの検索」Web 画面が表示され、指定したロールのルーティング・ルールを最初に表示できます。10-26 ページの「[自動通知処理ルールの定義](#)」を参照してください。
7. 「プロセス検索」リンクを選択して、特定の検索基準と一致するワークフロー・プロセス・インスタンスのリストを問い合わせます。特定のプロセス・インスタンスが見つかり、そのステータスの詳細をワークフロー・モニターに表示できます。11-10 ページの「[プロセス検索 Web 画面の使用](#)」を参照してください。
8. 「ユーザー設定項目」リンクを選択し、Oracle Workflow との対話方法を制御する作業環境を設定します。9-6 ページの「[ユーザー設定項目の設定](#)」を参照してください。
9. ワークフロー管理者権限を持つロールでログインしている場合は、「グローバル・ワークフロー・プリファレンス」リンクを選択すると、ユーザーと Oracle Workflow との対話方法を制御するグローバル設定項目を設定できます。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。
10. ワークフロー管理者権限を持つロールでログインしている場合は、「文書ノード」リンクが表示されます。この機能は今後使用する目的で確保されています。このリンクを使用して処理を実行する必要はありません。
11. 「項目タイプの定義」リンクを選択し、「項目タイプの検索」Web 画面にアクセスします。「項目タイプの検索」Web 画面を使用して、特定の項目タイプ定義を問い合わせ、「項目タイプの定義」ページに表示します。3-25 ページの「[項目タイプの定義 Web 画面](#)」を参照してください。
12. ワークフロー管理者権限を持つロールでログインしている場合は、「プロセスの開始」リンクを選択すると、特定のワークフロー・プロセス定義をテストできます。12-2 ページの「[ワークフロー定義のテスト](#)」を参照してください。
13. ワークフロー管理者権限を持つロールでログインしている場合は、「デモンストレーション・ページ」リンクを選択すると、「デモンストレーション」ホーム・ページにアクセスできます。「デモンストレーション」ホーム・ページを使用すると、Oracle Workflow によって提供されるデモンストレーション・ワークフロー・プロセスをどれでも開始できます。15-2 ページの「[サンプル・ワークフロー・プロセス](#)」を参照してください。
14. ワークフロー管理者権限を持つロールでログインしている場合は、「イベント」リンクを選択すると、ビジネス・イベント・システム・イベントを表示および定義できます。13-6 ページの「[イベントの定義](#)」を参照してください。
15. ワークフロー管理者権限を持つロールでログインしている場合は、「イベント / グループの検索」リンクを選択すると、特定の検索基準に一致するイベントおよびイベント・グループを問い合わせることができます。13-15 ページの「[イベントの検索](#)」を参照してください。
16. ワークフロー管理者権限を持つロールでログインしている場合は、「システム」リンクを選択すると、ビジネス・イベント・システム・システムを表示および定義できます。13-19 ページの「[システムの定義](#)」を参照してください。

17. ワークフロー管理者権限を持つロールでログインしている場合は、「システムの検索」リンクを選択すると、特定の検索基準に一致するシステムを問い合わせることができます。13-21 ページの「[システムの検索](#)」を参照してください。
18. ワークフロー管理者権限を持つロールでログインしている場合は、「エージェント」リンクを選択すると、ビジネス・イベント・システム・エージェントを表示および定義できます。13-30 ページの「[エージェントの定義](#)」を参照してください。
19. ワークフロー管理者権限を持つロールでログインしている場合は、「エージェントの検索」リンクを選択すると、特定の検索基準に一致するエージェントを問い合わせることができます。13-34 ページの「[エージェントの検索](#)」を参照してください。
20. ワークフロー管理者権限を持つロールでログインしている場合は、「イベント・サブスクリプション」リンクを選択すると、ビジネス・イベント・システム・サブスクリプションを表示および定義できます。13-48 ページの「[イベント・サブスクリプションの定義](#)」を参照してください。
21. ワークフロー管理者権限を持つロールでログインしている場合は、「サブスクリプションの検索」リンクを選択すると、特定の検索基準に一致するサブスクリプションを問い合わせることができます。13-53 ページの「[イベント・サブスクリプションの検索](#)」を参照してください。
22. ワークフロー管理者権限を持つロールでログインしている場合は、「セットアップのチェック」リンクを選択すると、ローカル・エージェントについて、ビジネス・イベント・システム・セットアップおよびスケジュールのリスナーと伝播をチェックできます。13-56 ページの「[メッセージ伝播の設定](#)」を参照してください。
23. ワークフロー管理者権限を持つロールでログインしている場合は、「イベントの呼出し」リンクを選択すると、ビジネス・イベントをイベント・マネージャに対して発生させることができます。13-71 ページの「[イベントの呼出し](#)」を参照してください。
24. ワークフロー管理者権限を持つロールでログインしている場合は、「システムのサインアップ」リンクを選択すると、システムを別のシステムにサインアップしてビジネス・イベントを受信することができます。13-73 ページの「[システムのサインアップ](#)」を参照してください。
25. ワークフロー管理者権限を持つロールでログインしている場合は、「システム識別子」リンクを選択すると、システムのサインアップに必要なシステム識別子情報を取り出すことができます。13-74 ページの「[システム識別子情報の取得](#)」を参照してください。
26. ワークフロー管理者権限を持つロールでログインしている場合は、「イベント・キュー・サマリー」リンクを選択すると、ビジネス・イベント・システムによって使用されるローカル・キューを確認できます。13-80 ページの「[ローカル・キューの確認](#)」を参照してください。

## ユーザー設定項目の設定

「ユーザー設定項目」Web 画面から設定できるユーザー設定項目を指定し、Oracle Workflow との対話方法を制御できます。「ユーザー設定項目」Web 画面で値を指定すると、「グローバル・ユーザー設定項目」Web 画面でワークフロー管理者によって設定されたデフォルトのグローバル値が上書きされます。

### ▶ ユーザー設定項目の設定

1. Web ブラウザを使用して、次の URL で Oracle Workflow ホーム・ページに接続します。

```
<webagent>/wfa_html.home
```

Oracle Workflow ホーム・ページから「ユーザー設定項目」リンクを選択します。

または、次のように「ユーザー設定項目」Web 画面に直接接続します。

```
<webagent>/wf_pref.edit
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

---

---

**注意：** これらのページにはセキュリティが適用されるため、現行の Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。

---

---



2. 「ユーザー設定項目」Web 画面に、現在のユーザー設定項目の要約が表示されます。「更新」を選択し、この設定項目を変更します。



3. 「言語」フィールドと「地域」フィールドで、値リストを使用して NLS\_LANGUAGE と NLS\_TERRITORY の組合せを選択し、通知セッションのデフォルトの言語依存動作と地域依存書式設定を定義します。
4. 「日付書式」フィールドで、自分のデータベース・セッションで使用する Oracle8i 準拠の日付書式を指定します。Oracle8i 準拠の日付書式の例としては、DD-Mon-RRRR があります。日付書式を指定しなかった場合、日付書式のデフォルトは DD-MON-YYYY になります。

---



---

**注意：** Oracle Workflow では、日付書式に時間書式を指定しなくても、表示される日付によっては時間要素が含まれる場合があります。日付書式とともに時間書式を指定すると、Oracle Workflow で時間要素が表示される場合に、日付の後に 2 つの時間要素が表示されます。

---



---

5. 「文書ホーム・ノード」フィールドは空白のままにしてください。この機能は今後使用する目的で確保されています。
6. 「電子メール通知を送信してください」セクションで、次の通知設定項目を選択します。
  - HTML メール： 通知が HTML 電子メールとして送られてきます。HTML 形式のメッセージ本文を表示できるメール・リーダーを使用する必要があります。
  - HTML 添付ファイル付きのプレーン・テキスト・メール： 通知がプレーン・テキスト電子メールとして送信されますが、添付ファイルとして HTML 形式版の通知が添付されます。
  - プレーン・テキスト・メール： 通知がプレーン・テキストの電子メールとして送られてきます。
  - プレーン・テキスト要約メール： すべての通知の要約がプレーン・テキストの電子メールとして送られてきます。個々の通知を処理するには、「通知」Web 画面を使用する必要があります。
  - メールを送信しないでください： 通知は電子メールとして送信されません。通知の表示と処理には、「通知」Web 画面を使用する必要があります。
7. 変更の終了後に「OK」をクリックします。

**関連項目：**

2-46 ページ [「通知環境設定」](#)

---

## 通知の表示と応答の処理

この章では、ワークフロー・プロセスに関与するユーザーがワークフロー通知を表示して応答するための様々な方法について説明します。また、Oracle Workflow で通知が自動的に処理されるようにルールを定義する方法についても説明します。

## 通知処理の概要

Oracle Workflow では、ワークフロー・エンジンがワークフロー・プロセスで通知アクティビティを実行すると、ロールに通知が送信されます。通知アクティビティでは、一部の手動処理を担当するロールを指定する場合や、単にプロセス関連情報をロールに中継する場合があります。通知をロールに正常に配信するには、そのロールを Oracle Workflow ディレクトリ・サービスで定義する必要があります。

ロールのメンバーは、Oracle Workflow ディレクトリ・サービス内での自分のロールの通知環境設定に応じて、3 つのインタフェースのいずれかを使用して通知を表示します。個別の通知ごとに電子メールを受信するか、すべての通知が要約された 1 件の電子メールを受信するか、ワークフローの「通知」Web 画面で通知を問合せできます。2-20 ページの「[手順 WF-3 Oracle Workflow のディレクトリ・サービスの設定](#)」および 9-6 ページの「[ユーザー設定項目の設定](#)」を参照してください。

各通知メッセージには、プロセスに関する状況依存情報と、応答が必要であれば通知への応答方法を含めることができます。また、Web URL のポインタおよび Oracle Applications フォームへの参照を含めると、通知関連の追加情報を取得できます。

通知受信者は、通知を適切なタイミングで表示したり応答できない場合があります。ワークフロー・プロセスにボトルネックが発生しないように、自動通知処理を使用して、通知を自動的に処理するように Oracle Workflow に指示するルールを定義できます。

## 電子メールによる通知の閲覧

「ユーザー設定項目」Web 画面で、通知環境設定が「プレーン・テキスト・メール」、「HTML メール」または「HTML 添付ファイル付きのプレーン・テキスト・メール」に設定されており、ワークフロー管理者が通知メーラーを実行するよう設定している場合は、ワークフローの通知を電子メールのメッセージとして配布できます。

電子メール・ソフトウェアでサポートされるのが、添付なしのプレーン・テキスト・メッセージのみの場合は、通知環境設定を「プレーン・テキスト・メール」に設定してください。

電子メール・ソフトウェアでメッセージ本文内の HTML 形式を解釈して表示できる場合は、通知環境設定で「HTML メール」を選択してください。HTML メールでは、(通知を完了するためにアクセスする必要のある) サポート対象の情報ソースに直接リンクできます。

電子メール・ソフトウェアで、メッセージ本文のプレーン・テキストの表示のみでなく、メッセージの添付ファイルも表示できる場合は、通知環境設定を「HTML 添付ファイル付きのプレーン・テキスト・メール」に設定してください。

応答が必要な電子メール通知は、ユーザーが通知に応答するまでステータスが「オープン」となります。FYI (参考用: For your information) 通知など、応答の必要がない電子メール通知の場合は、ワークフロー管理者が通知メーラーの設定時に、通知ステータスの更新方法を決定します。Oracle Workflow では、設定時の構成に応じて、電子メールで通知を送信した後に FYI 通知のステータスが自動的に「完了」に更新されるか、「通知ワークリスト」Web 画面で手動でクローズするまで通知が「オープン」ステータスのままになります。



読み終わった FYI メッセージは、受信ボックスから削除できます。ただし、組織の通知メーラーが電子メールで送信後も FYI 通知をオープンにしておくように設定されている場合は、すでに通知メッセージを電子メールの受信ボックスから削除していても、「通知ワークリスト」も使用して通知を手動でクローズする必要があります。10-17 ページの「[ワークリストの通知の表示](#)」を参照してください。

プレーン・テキストの電子メール通知には、テンプレートによる応答と直接応答という 2 つの応答方法があります。組織の応答方法は、ワークフロー管理者が通知メーラーの設定時に決定します。テンプレートによる応答方法の場合は、通知で提供される応答プロンプトのテンプレートを使用して返信し、各プロンプトの後に応答値を引用符で囲んで入力します。直接応答方法の場合は、応答値を返信の 1 行目として直接入力します。

テンプレートによる応答の場合も、直接応答の場合も、電子メール通知は Oracle Workflow Builder で定義されている標準メッセージ・テンプレートに基づいています。どちらのテンプレートでも、返信に必要な構文が記述され、通知の確認に必要な情報が表示されます。どちらのタイプのメッセージにも、カスタム・サイト情報、通知の期日、応答処理に必要なすべての情報なども指定されます。2-65 ページの「[手順 WF-10 メッセージ・テンプレートの変更](#)」を参照してください。

通知に電子メールで応答する場合、返信メッセージには元の通知メッセージからの通知 ID (NID) とアクセス・キーを挿入する必要があります。通知メーラーで応答を正常に処理できるのは、応答に正しい NID とアクセス・キーの組合せを挿入している場合のみです。返信に NID とアクセス・キーが確実に含まれるようにするには、元のメッセージ全体を返信に挿入する方法と、NID 行を含む応答テンプレートを使用する方法があります。

---

---

**注意：** 通知のアクセス・キーは、通知システムにより NID ごとに生成される個別のランダム・キーです。このアクセス・キーは、実際にそのキーを含む通知を受け取ったユーザーのみがその通知に応答できるようにする、パスワードとして機能します。

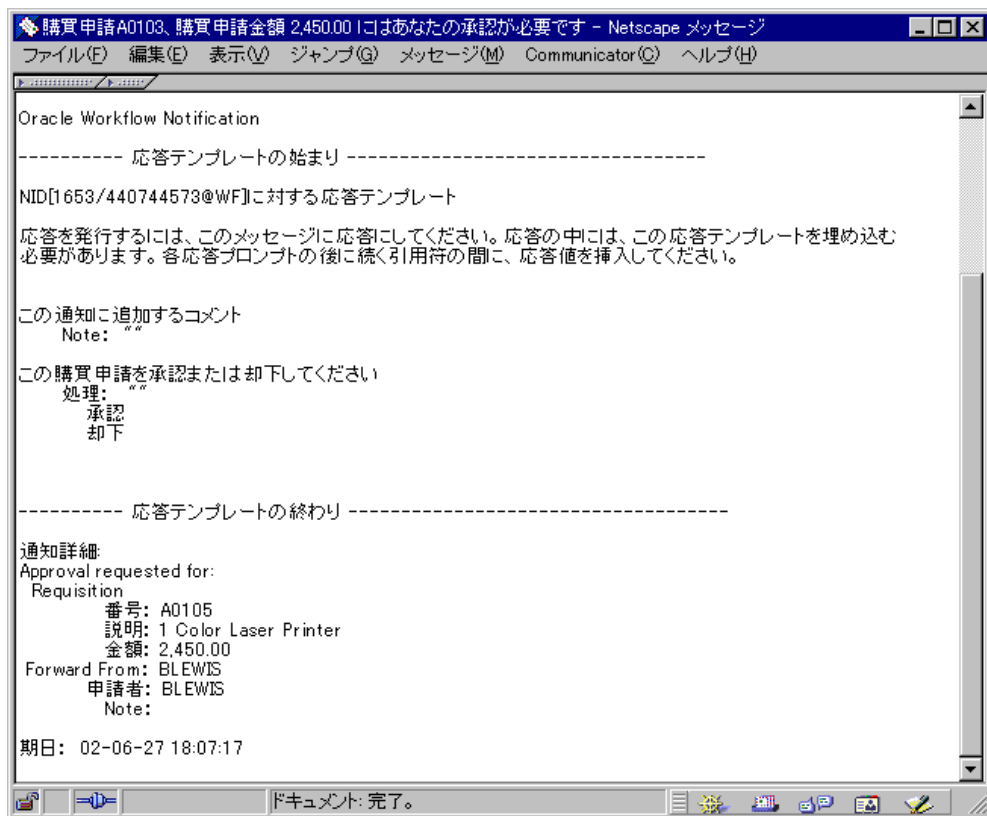
---

---

#### 関連項目：

2-52 ページ「[通知メーラーの起動](#)」

▶ テンプレートによる応答を使用したプレーン・テキストの電子メール通知への応答



1. プレーン・テキストの電子メール通知には、通知への応答時に役立つ情報が含まれています。通知によっては、この情報が他のソースへの参照や添付ファイルとして表示されることもあります。添付ファイルによっては、「通知」Web 画面を使用して通知を表示した場合にのみ表示できるものもあります。10-13 ページの「[Web ブラウザによる通知の表示](#)」を参照してください。
2. 通知に応答するには、メール・アプリケーションで返信コマンドを使用し、元の電子メール通知に返信します。
3. 返信に、元の通知からの応答テンプレートを挿入します。応答テンプレートには、応答プロンプトの他に、特別な通知 ID およびアクセス・キーが含まれています。この情報は、通知メーカーで応答対象の通知を識別するために必要になります。メール・アプリケーションにより、返信メッセージの生成時に元のメッセージの編集可能なコピーが挿入される場合は、そのコピーを使用して応答値を入力できます。それ以外の場合は、元

のメッセージからコピーし、貼り付けて、編集できる応答テンプレートのコピーを作成します。

4. 応答テンプレートの指示に従って、各応答プロンプトの後の引用符 (" ") に応答値を挿入します。通知システムは、応答値を文字として解釈するため、大文字と小文字は区別されます。
5. 応答に問題がなければ、メール・アプリケーションの送信コマンドを使用して、応答を送信します。

---

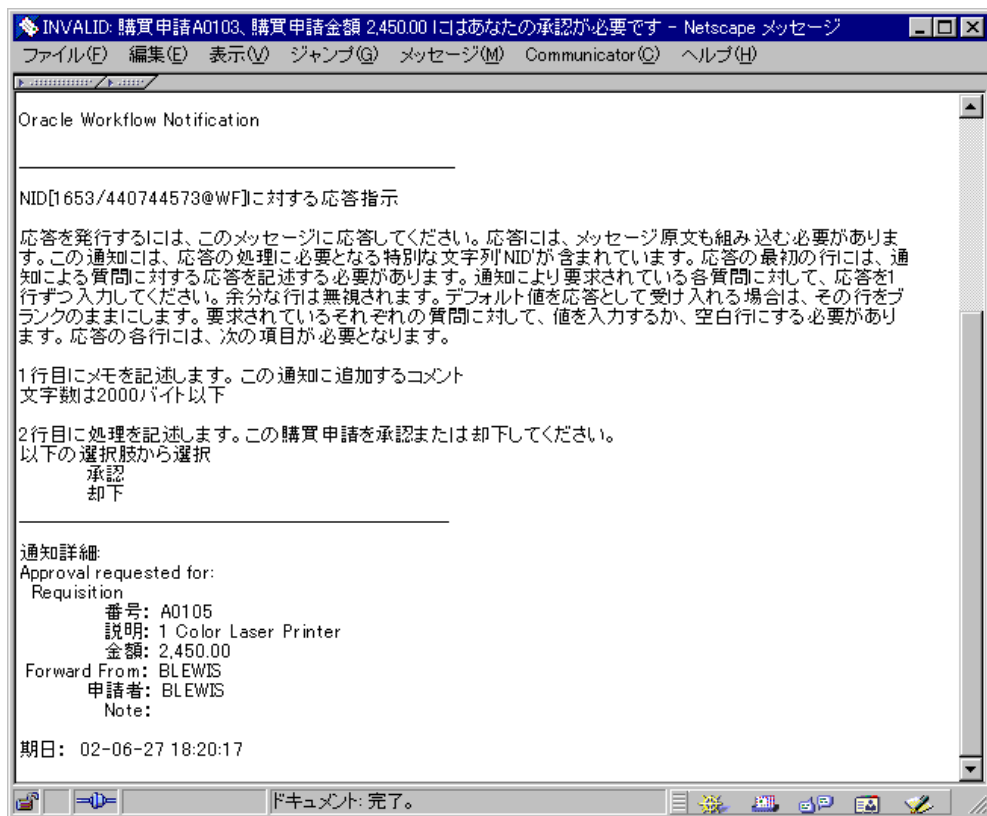
**注意：** 無効な応答を送信すると、通知システムから「無効な応答」のメッセージが送信されます。取消し済の通知に応答すると、その通知が取り消されたことを示すメッセージが送信されます。同様に、以前に応答済の通知に応答すると、その通知はすでに完了していることを示すメッセージが送信されます。

---

**関連項目：**

2-47 ページ「[プレーン・テキスト電子メール](#)」

➤ 直接応答を使用したプレーン・テキストの電子メール通知への応答



1. プレーン・テキストの電子メール通知には、通知への応答時に役立つ情報が含まれています。通知によっては、この情報が他のソースへの参照や添付ファイルとして表示されることもあります。添付ファイルによっては、「通知」Web 画面を使用して通知を表示した場合にのみ表示できるものもあります。10-13 ページの「[Web ブラウザによる通知の表示](#)」を参照してください。
2. 通知に応答するには、メール・アプリケーションで返信コマンドを使用し、元の電子メール通知に返信します。
3. 返信に、元の通知メッセージのテキストを挿入します。このテキストには、特別な通知 ID とアクセス・キーが含まれています。この情報は、通知メーカーで応答対象の通知を識別するために必要です。
4. 通知メッセージに含まれる構文指示に従って返信をフォーマットします。応答値は返信の 1 行目に挿入し、各行で個別の応答値を表します。

応答値が複数行にまたがる場合は、応答値全体を二重引用符 (") で囲む必要があります。二重引用符で囲まれた値全体が 1 行と見なされます。

通知システムは、応答値を文字として解釈するため、大文字と小文字は区別されます。

応答プロンプトでデフォルトの応答値が提供される場合は、該当する応答行を空白にしてデフォルト値を受け入れることができます。

5. 応答に問題がなければ、メール・アプリケーションの送信コマンドを使用して、応答を送信します。

---

---

**注意：** 無効な応答を送信すると、通知システムから「無効な応答」のメッセージが送信されます。取消し済の通知に応答すると、その通知が取り消されたことを示すメッセージが送信されます。同様に、以前に応答済の通知に応答すると、その通知はすでに完了していることを示すメッセージが送信されます。

---

---

## 例

次の例は、一連の応答インストラクションと 3 つの有効な応答を示しています。

---

### 応答インストラクション

---

1 行目にアクションを入力してください。承認しますか？ 値は次のどちらかです（デフォルトは「却下」）。

承認する

却下する

2 行目にレビュー・コメントを入力してください。値は 2000 バイト以内にします。

3 行目に必須の日付を入力してください。必須の日付がない場合は空白のままにします。値は「DD-MON-YYYY」形式の日付にします。

4 行目に最大金額を入力してください。これは承認額の最大値です。値は数値にします。デフォルトは 1500 です。

---

---

**有効な応答 A: 承認**

---

承認する

この項目が予想と一致しているかどうかをお知らせください。

1998 年 1 月 1 日

1000.00

---

---

**有効な応答 B: 否認**

---

却下する

高額すぎます。

---

**注意：** 空白行で応答した場合は、デフォルトの応答値が使用されます。

---

---

**有効な応答 C: 否認**

---

却下する

「この品目は高額すぎます。低コストの交換品を探すか、  
この品目の承認を裏付ける理由を追加してください。」

1998 年 1 月 1 日

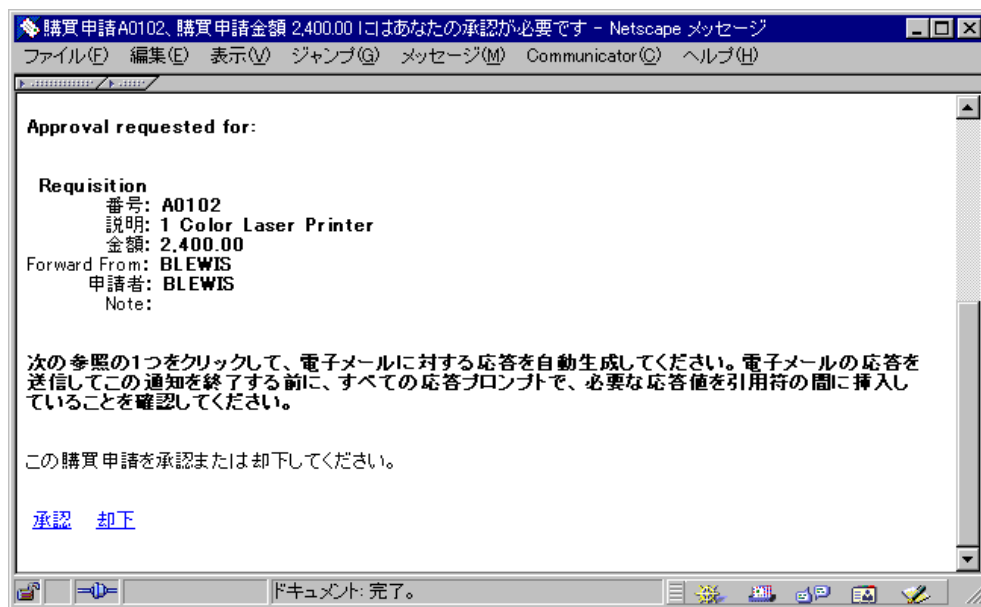
1000.00

---

**関連項目：**

2-47 ページ [「プレーン・テキスト電子メール」](#)

▶ HTML 形式の電子メール通知への応答



1. HTML 形式の電子メール通知には、通知への応答時に役立つ情報が含まれています。通知によっては、この情報が他のソースへのリンクや添付ファイルとして表示されることもあります。

---

---

**注意：** HTML 形式の電子メール通知には、常に 1 つの添付ファイルが含まれています。この添付ファイルは「通知の詳細」リンクと呼ばれるもので、ここから「通知の詳細」Web 画面の通知に直接リンクできます。

この添付ファイルをオープンするには、Web ブラウザで JavaScript とフレームをサポートしている必要があるため注意してください。「通知の詳細」リンクの添付文書をオープンすると、自動的に Web サーバーとの Web セッションを確立しようとします。セッションを確立するために、この添付ファイルではユーザーのアクセスが認証され、通知がまだオープンされているかどうかを確認され、通知がすでに完了している場合は、メッセージが表示されます。

ワークフロー管理者が、「通知の詳細」リンクの選択時にログインを要求するように通知メーラーを構成している場合は、「通知の詳細」Web 画面にアクセスする前にログインを求めるプロンプトが表示されます。

この「通知の詳細」ページからは、通知メーラーを介して応答を処理する必要がなく、通知に直接応答できます。「通知の詳細」ページから応答する場合は、残りの手順を省略してください。

**参照：** 10-19 ページの「[通知の詳細の表示](#)」を参照してください。

---

---

2. 通知に関するすべての情報を確認してから、通知の最後に表示される応答リンクを 1 つクリックします。

---

---

**注意：** ワークフロー管理者が、Outlook Express のメッセージ・テンプレートとしてワークフロー・オープン・メールを使用するように通知メーラーを構成している場合は、個々の応答へのリンクのかわりに、「Click here to respond」というリンクが表示されます。「Click here to respond」リンクから、「通知の詳細」Web ページの通知にアクセスできます。

このリンクをオープンするには、Web ブラウザで JavaScript とフレームをサポートしている必要があるため注意してください。「Click here to respond」リンクを選択すると、Web サーバーとの Web セッションが自動的に確立されます。セッションを確立するために、この添付ファイルではユーザーのアクセスが認証され、通知がまだオープンされているかどうかを確認され、通知がすでに完了している場合は、メッセージが表示されます。ただし、ネットワークに接続していない場合は、このリンクは使用できません。

「通知の詳細」ページにアクセスすると、このページから通知に直接応答できます。この場合、残りの手順は省略してください。

---

---



3. 各応答リンクにより、プレーン・テキストの返信メールが自動的に生成されます。この返信には、適切な「To: 電子メール・アドレス」と、メッセージ本文内の応答テンプレートが含まれています。応答テンプレートは、応答対象となる通知の識別に必要な通知 ID とアクセス・キー、および選択した応答で編集される応答プロンプトからなっています。

---

---

**警告：** 電子メールによる応答には、HTML 形式を挿入しないでください。

---

---

4. 通知によっては、自動的に生成された電子メールの応答テンプレートで、選択した応答の他にも情報が要求される場合があります。各応答プロンプトの後ろに引用符 (') で囲まれている応答値のテキストを編集し、応答を指定してください。

---

---

**注意：** 応答値を囲む引用符として、プレーン・テキストの応答テンプレートでは二重引用符が使用されますが、HTML 形式の電子メール通知の応答リンクから生成される応答テンプレートでは一重引用符が使用されます。一重引用符を使用することで、応答リンクの  
<A HREF="mailto:"> タグに含まれる二重引用符は処理できないが、一重引用符は処理できる電子メール・アプリケーションにも対応できます。

---

---

5. 応答に問題がなければ、メール・アプリケーションの送信コマンドを使用して、応答を送信します。

---

---

**注意：** 無効な応答を送信すると、通知システムから「無効な応答」のメッセージが送信されます。取消し済の通知に応答すると、その通知が取り消されたことを示すメッセージが送信されます。同様に、以前に応答済の通知に応答すると、その通知はすでに完了していることを示すメッセージが送信されます。

---

---

#### 関連項目：

2-48 ページ「[HTML 形式の電子メール](#)」

► **HTML 添付ファイル付きのプレーン・テキスト電子メール通知への応答**

1. 添付ファイル付きのプレーン・テキストの電子メール通知には、通知への応答時に役立つ情報が含まれています。通知によっては、この情報が他のソースへのリンクやメッセージの添付ファイルとして、メッセージ本文中に表示されることもあります。また、この通知には、常に少なくとも次の 2 つの添付ファイルが含まれています。
  - HTML メッセージ本文： 通知メッセージの HTML 形式のバージョンです。
  - 通知の詳細リンク： 「通知の詳細」 Web ページに表示される通知に直接リンクします。
2. 通知に対するすべての情報を確認してから、次の 3 つの方法のいずれかを使用して通知に応答します。
  - メール・リーダーで返信コマンドを使用し、プレーン・テキスト・メッセージの本文に示されるインストラクションに従って応答します。10-4 ページの「[テンプレートによる応答を使用したプレーン・テキストの電子メール通知への応答](#)」および 10-6 ページの「[直接応答を使用したプレーン・テキストの電子メール通知への応答](#)」を参照してください。
  - HTML メッセージ本文の添付ファイルを表示し、HTML メッセージ本文の最後にある応答リンクのいずれかを選択して応答します。10-9 ページの「[HTML 形式の電子メール通知への応答](#)」を参照してください。
  - 「通知の詳細リンク」添付ファイルを選択し、「通知の詳細」 Web 画面を表示します。10-19 ページの「[通知の詳細の表示](#)」を参照してください。

---

**注意：** この添付ファイルをオープンするには、Web ブラウザで JavaScript とフレームをサポートしている必要があります。「通知の詳細」リンクの添付文書をオープンすると、自動的に Web サーバーとの Web セッションを確立しようとします。セッションを確立するために、この添付ファイルではユーザーのアクセスが認証され、通知がまだオープンされているかどうかを確認され、通知がすでに完了している場合は、メッセージが表示されます。

ワークフロー管理者が、「通知の詳細」リンクの選択時にログインを求めると、通知メーカーを構成している場合は、「通知の詳細」 Web 画面にアクセスする前にログインを求めるプロンプトが表示されます。

---

**関連項目：**

2-50 ページ「[HTML 添付ファイル付きのプレーン・テキスト電子メール](#)」

### ▶ 他のユーザーへの通知の再割当て

- メール・リーダーの転送機能を使用して、電子メール通知を他のユーザーに転送または再割当てします。HTML 添付ファイルには「再割当て」ボタンを使用しないでください。

---

**注意：** 電子メールで通知を他のユーザーに転送する場合は、対象ユーザーに自分のかわりに通知に応答するように依頼します。通知の所有権は、まだ元のユーザーに定義されていることに注意してください。通知とその所有権を他のユーザーに譲渡する場合は、必ず「通知」Web 画面から処理を行います。10-13 ページの「[Web ブラウザによる通知へのアクセス](#)」を参照してください。

---

## Web ブラウザによる通知の表示

JavaScript とフレームがサポートされる Web ブラウザを使用し、「通知」Web 画面に通知を表示して応答できます。

### ▶ Web ブラウザによる通知へのアクセス

1. Oracle Self-Service Web Applications を使用している場合は、Oracle Self-Service Web Applications のログイン・ページを使用してログオンし、適切なリンクを選択して「通知ワークリスト」ページを表示します。手順 1 にスキップしてください。
2. Oracle Self-Service Web Applications を使用していない場合は、次の方法でワークリストにアクセスできます。
  - 現在処理中の通知のワークリストに直接ナビゲートするには、次のように入力します。

```
<webagent>/wfa_html.worklist[?orderkey=<orderkey>
&status=<status>&user=<user>]
```

ワークリストの URL のうち、カッコ ([]) で囲まれた部分は、オプションで渡すことのできる引数を表します（カッコは省略してください）。

<> 内のテキストを次のように置き換えます。

- <webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。
- <orderkey> は、通知リストの順序を決定するキーを表します。有効な値は、PRIORITY、MESSAGE\_TYPE、SUBJECT、BEGIN\_DATE、DUE\_DATE、END\_DATE および STATUS です。<orderkey> を NULL にすると、ワークリストは最初に PRIORITY、次に BEGIN\_DATE の降順に設定されます（最高の優先度と最新日付を持つ通知が最初に表示されます）。

- `<status>` は、表示する通知のステータスを表します。有効な値は、OPEN、CLOSED、CANCELED および ERROR です。`<status>` を NULL にすると、OPEN ステータスの通知が URL によって表示されます。
  - `<user>` は、通知を問い合わせるロールの内部名を表します。この引数を指定できるのは、現行の Web セッションにワークフロー管理者権限を持つロールでログインしている場合のみです。ロールに管理者権限が付与されていない場合や、`<user>` を空白にした場合は、現行ロールに対する通知が URL によって表示されます。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。
  - 特定の検索基準と一致する通知のワークリストを表示するには、「通知の検索」Web 画面にジャンプして、次のように入力します。

```
<webagent>/wfa_html.find
```
  - 「通知ワークリスト」または「通知の検索」Web 画面には、Oracle Workflow ホーム・ページからもナビゲートできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。
  - Oracle Applications ユーザーの場合にのみ、システム管理者は Oracle Applications の関数 FND\_FNDWFNOT を使用して、アプリケーションに「通知ワークリスト」Web 画面を追加できます。この関数は、Web 画面の `wfa_html.worklist` をコールします。Oracle Applications のシステム管理者または開発者は、この関数をユーザーの職責の「ナビゲート」メニューに追加するか、この関数を Oracle Applications フォームからコールする必要があります。これにより、変更済のメニューまたはフォームを使用して、「通知ワークリスト」Web 画面にナビゲートできます。『Oracle Applications System Administrator's Guide』の「Overview of Menus and Function Security」、『Oracle Applications Developer's Guide』の「Overview of Menus and Function Security」、「Menus Window」および「Overview of Form Development Steps」を参照してください。
3. Oracle Self-Service Web Applications を使用していない場合、および Web ブラウザ・セッションから最初に Oracle Workflow URL にアクセスした場合は、ログオン用の有効なユーザー名とパスワードの入力を求めるプロンプトが表示されます。
  4. ユーザー名とパスワードを入力します。
  5. 「OK」を選択します。間違えた場合は、値を消去して再入力できます。

`wfa_html.worklist` の URL を使用した場合は、10-17 ページの「[ワークリストの通知の表示](#)」に進んでください。

► 通知の検索

通知の検索 - CDOUGLAS - Netscape

ファイル(E) 編集(E) 表示(V) ジャンプ(G) Communicator(C) ヘルプ(H)

通知の検索

送信元

状態 オープン

タイプ すべて

件名

送信日 -

期限 -

優先度 すべて

☐ 通知の委任先

検索

ドキュメント: 完了。

1. 「通知の検索」ウィンドウを使用して、特定の通知を検出するための検索基準を入力できます。現行のセッションに通常のワークフロー・ユーザーとしてログオンしている場合は、自分が所有している通知の検索基準を指定できます。検索基準として、次の項目を使用できます。
  - 送信元: ロールを指定して、そのロールから送信された通知をすべて検索します。通知の「送信元」ロールは、`#FROM_ROLE` メッセージ属性によって決定されます。4-24 ページの「[#FROM\\_ROLE 属性](#)」を参照してください。
  - 状態: 通知の進行状況として、「取消」、「クローズ」、「無効な応答」または「オープン」を選択します。任意のステータスの通知を表示するには、「すべて」を選択します。
  - タイプ: 通知の項目タイプを選択します。任意の項目タイプの通知を表示するには、「すべて」を選択します。
  - 件名: 検索する通知の件名を入力します。このフィールドでは、テキスト文字列の大文字 / 小文字が区別されず、パーセントの記号 (%) をワイルドカードとして解釈します。

- 送信日： 通知が送信された日付、または送信してからの日数を入力します。データベースのデフォルトの日付書式を使用してください。
  - 期限： 通知を完了する日付、または完了するまでの日数を入力します。データベースのデフォルトの日付書式を使用してください。
  - 優先度： 検索する通知の優先度として「高」、「標準」または「低」を選択します。任意の優先度の通知を表示するには、「すべて」を選択します。
  - 通知の委任先： 所有権は譲渡せずに、指定したロールに対して転送した通知を検索する場合は、この基準をオンにします。このフィールドの隣の上矢印のアイコンをクリックすると、有効なロールのリストが表示されます。10-24 ページの「[値リストの使用](#)」を参照してください。
2. ワークフローの管理者権限が付与されているユーザーは、自分が所有権を持っていない通知も検索できます。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

前述の標準の検索基準（ただし「通知の委任先」は除く）の他にも、次の選択基準オプションを指定できます。

- 通知 ID: 特定の通知 ID を指定します。通知 ID を指定すると、他の選択基準はすべて無効になるため注意してください。
- 所有者: ロールを指定し、そのロールが所有している通知をすべて識別します。このフィールドの隣の上矢印のアイコンをクリックすると、有効なロールのリストが表示されます。10-24 ページの「値リストの使用」を参照してください。
- 宛先: ロールを指定し、そのロールに送信された通知をすべて識別します。このフィールドの隣の上矢印のアイコンをクリックすると、有効なロールのリストが表示されます。10-24 ページの「値リストの使用」を参照してください。

**注意:** 元の所有者が（所有権は譲渡せずに）作業を別のロールに委任した通知を識別するには、「所有者」フィールドと「宛先」フィールドで異なるロールを指定します。このように 2 つの基準を組み合わせると、標準の「通知の検索」画面で「通知の委任先」の基準を選択した場合と同じように検索されます。

3. 「検索」ボタンを選択して「ワークリスト」ウィンドウを開きます。

▶ ワークリストの通知の表示



1. 「通知ワークリスト」に「通知の検索」ページからナビゲートした場合は、指定した検索基準と一致する通知が表示されます。このページに直接ナビゲートした場合は、オープンしている通知のすべてのリストが表示されます。

「ワークリスト」には、通知ごとに次の情報が表示されます。

- 優先度： 通知の優先度が「高」または「低」の優先度のアイコンで表されます。
- タイプ： ワークフロー・プロセスと通知に関連付けられている項目タイプ。
- 送信元： 通知の送信元のロール。通知の「送信元」ロールは、**#FROM\_ROLE** メッセージ属性によって決定されます。4-24 ページの「**#FROM\_ROLE 属性**」を参照してください。
- 件名： 通知の説明。

---

**注意：** 通知が「オープン」で応答が必要な場合は、「件名」リンクの隣に応答要求アイコンが表示されます。

---

- 送信日： 通知の配信日。
  - 期限： 通知を完了する必要がある日付。
2. 列ヘッダーのどれかをクリックし、その列の昇順に通知をソートします。
  3. ツールバーの「検索」アイコンを使用すると、いつでも「通知の検索」画面に戻って検索基準を指定し、「ワークリスト」で通知の検索対象をさらに限定できます。
  4. 「ワークリスト」では、応答が不要な FYI タイプの複数の通知を同時にクローズできます。完了する FYI タイプの通知ごとに「選択」をオンにして、「完了」を選択します。
  5. 通知を 1 つのグループとしてまとめて再割当てすることもできます。再割当てする通知の「選択」をオンにして「再割当て ...」を選択します。「通知の再割当て」ページが表示され、通知をどのグループに対してどのように割り当てるかを指定できます。10-13 ページの「**他のユーザーへの通知の再割当て**」を参照してください。
  6. 通知の詳細にナビゲートし、通知の「件名」リンクをクリックして、通知を処理できます。



## 通知の詳細の表示



1. 「通知の詳細」ページでは、通知のすべての詳細が上のフレームに、通知の応答セクションが下のフレームに表示されます。どちらのフレームもスクロールとサイズ変更が可能です。
2. 上のフレームには、メッセージ本文に埋め込まれているリンクが表示されることもあります。これは、通知の追加情報のソースへのリンクです。参照 URL リンクは、別の Web ブラウザ・ウィンドウをオープンし、指定された URL に接続します。
3. 上のフレームには、メッセージ本文の後に添付アイコンが表示されることもあります。これらのアイコンも、通知に関する追加情報のソースへのリンクです。添付リンクには、次の 3 種類があります。
  - 参照 URL リンク：別の Web ブラウザ・ウィンドウをオープンし、指定された URL に接続します。
  - PL/SQL または PL/SQL CLOB 文書リンク：PL/SQL 関数から生成された文書の内容を表示します。
  - Oracle Applications フォーム・リンク：Oracle Applications embedded Workflow を使用している場合は、このリンクを使用して、基礎となるメッセージ属性で参照



される Oracle Applications フォームにドリルダウンできます。メッセージ属性の定義によっては、Oracle Applications フォームに適切なコンテキスト情報を自動的に表示できます。

---

**注意：** 添付フォーム・アイコンが通知メッセージに表示されるのは、「ワークリスト」 Web 画面を最初に Oracle Applications のメニューから起動した場合のみです。添付フォーム・アイコンを使用するには、Oracle Applications のソケット・リスナー・ポートを有効化し、ソケット・リスナー・ポートをフォームが起動されるポートに設定する必要があります。2-38 ページの「[手順 WF-6 ソケット・リスナー・プロファイル・オプションの設定](#)」を参照してください。

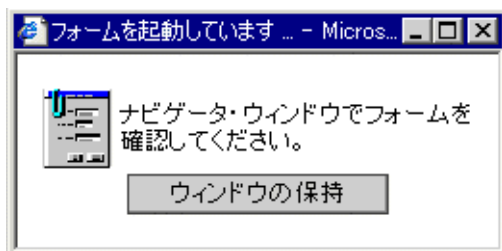
---

---

**注意：** 通知システムでは、最初に Oracle Applications で、受信者の職責にリンク・フォームをオープンするための適切なセキュリティが適用されるかどうかを検証されます。職責がフォームのオープンを許可されていない場合は、添付フォーム・アイコンは使用不可になり、その旨を示すメッセージが表示されます。また、上のフレームに添付されている参照専用フォーム内の情報を更新することはできません。

---

添付フォーム・アイコンを選択すると、起動ウィンドウが表示され、ソケットの接続が確立されます。「ウィンドウの保持」ボタンを選択すると、起動ウィンドウを引き続き表示できます。「ウィンドウの保持」ボタンを選択しない場合、このウィンドウは 3 秒後に自動的に閉じます。



起動されたフォームは、通知が表示されているブラウザ・ウィンドウ上部の Oracle Applications 「ナビゲータ」ウィンドウに表示されます。

---

**注意：** フォームが表示されない場合は、JInitiator のコンソールでエラー・メッセージを確認してください。

---

4. 次のような「応答」セクションが表示されます。

- 通知に応答する必要があるが、その応答が通知アクティビティの結果に影響しない場合、応答プロンプトはすべてフィールドやドロップ・ダウン・リストとして表示されます。応答値を入力した後に、「送信」ボタンを選択して応答を送信します。
- 通知に応答する必要があり、応答の1つが通知アクティビティの結果になる場合は、その応答が前述のように応答ボタン・セットの最後に表示され、選択できます。各ボタンは、応答プロンプトについて選択が可能です。その他の応答プロンプトがある場合は、すべてプロンプトの上にフィールドまたはドロップ・ダウン・リストとして表示されます。最後の応答プロンプトのボタンをクリックすると同時に、通知の応答が送信されます。
- 通知に応答する必要がなければ、応答セクションにその旨が表示されます。「応答」セクションで「完了」を選択して通知を閉じると、次回にオープン通知を問い合わせるときに、その通知は通知要約リストに表示されなくなります。

---

**注意：** 任意の応答プロンプトをクリックすると、応答属性の詳細情報を表示できます。

---

---

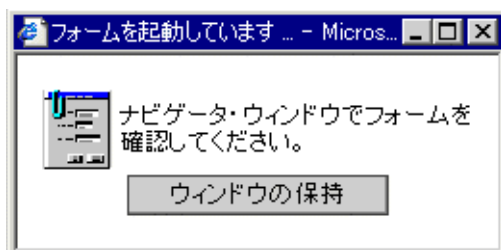
**注意：** ワークリストを Oracle Applications から起動すると、「応答」セクションに添付フォーム・アイコンが表示される場合があります。このアイコンをクリックすると、Oracle Applications フォームにドリルダウンして応答を完了できます。

---



添付フォーム・アイコンを使用するには、Oracle Applications のソケット・リスナー・ポートを有効化し、ソケット・リスナー・ポートをフォームが起動されるポートに設定する必要があります。2-38 ページの「[手順 WF-6 ソケット・リスナー・プロファイル・オプションの設定](#)」を参照してください。

添付フォーム・アイコンを選択すると、起動ウィンドウが表示され、ソケットの接続が確立されます。「ウィンドウの保持」ボタンを選択すると、起動ウィンドウを引き続き表示できます。「ウィンドウの保持」ボタンを選択しない場合、このウィンドウは3秒後に自動的に閉じます。



起動されたフォームは、通知が表示されているブラウザ・ウィンドウ上部の Oracle Applications 「ナビゲータ」ウィンドウに表示されます。

---

**注意：** フォームが表示されない場合は、JInitiator のコンソールでエラー・メッセージを確認してください。

---

5. 応答をいったん送信すると、「通知の詳細」ページから「ワークリスト」に戻し、応答した通知のステータスが「完了」になります。

---

**注意：** クローズした通知をもう一度オープンすると、「応答」セクションでは、対象の応答がすでに送信済であることが示され、応答として送信された値が表示されます。

---

6. 「通知の詳細」ページのツールバーの「検索」アイコンを使用すると、いつでも「通知の検索」画面に戻り、他の通知を検索して表示できます。

### ► 他のユーザーへの通知の再割当て

1. 通知を再割当てにするには、次の 2 通りの方法があります。
  - 「ワークリスト」ページで、1 つ以上の通知の「選択」をオンにして、「再割当て ...」を選択します。
  - 「ワークリスト」ページで、再割当てにする通知の件名リンクをクリックします。「詳細通知」ページが表示されるので、通知の「応答」フレームで「再割当て ...」を選択します。

---

**注意：** ワークフローには、通知の再割当てを制限する特別な #HIDE\_REASSIGN 属性を組み込むことができます。その場合、「再割当て」ボタンは「応答」フレームに表示されないため、通知を再割当てすることはできません。4-23 ページの「[#HIDE\\_REASSIGN 属性](#)」を参照してください。

---



2. 「通知の再割当て」ページが表示され、画面の下部に再割当ての対象として選択された通知の要約が表示されます。

**注意：** 特定の通知の「応答」フレームで「再割当て」ボタンをクリックすると、選択した通知の要約は表示されません。

3. 「再割当て先」フィールドで上矢印のアイコンをクリックすると、ウィンドウが表示され、選択するロールのリストを検索できます。10-24 ページの「[値リストの使用](#)」を参照してください。
4. ロールを選択した後、通知をどのように再割当てするかを指定します。この通知に回答する権限を新しいロールに付与する場合は、「通知に回答する許可を委任」を選択します。このオプションを選択すると、Oracle Workflow では、元の所有者が通知を所有したままになります。また、通知の所有権と職責を完全に他のロールに与える場合は、「通知の所有権を譲渡」を選択します。
5. 新しいロールに渡すコメントがあれば入力します。「OK」を選択します。通知が再割当てされると、Web ブラウザは「ワークリスト」ページに戻ります。このページでは、再割当てされた通知は「ワークリスト」には表示されなくなります。

---

---

**注意：** 通知後関数という特別なロジックを組み込んで、通知の委任先または譲渡先のロールが適切であるかどうかを検証したり、通知の再割当てを制限できます。その場合、通知を再割当てするときに役立つように警告メッセージを受け取ることもできます。8-14 ページの「[通知後関数](#)」を参照してください。

---

---

6. 「通知の再割当て」ページのツールバーの「検索」アイコンを使用すると、いつでも「通知の検索」画面に戻り、他の通知を検索して表示できます。

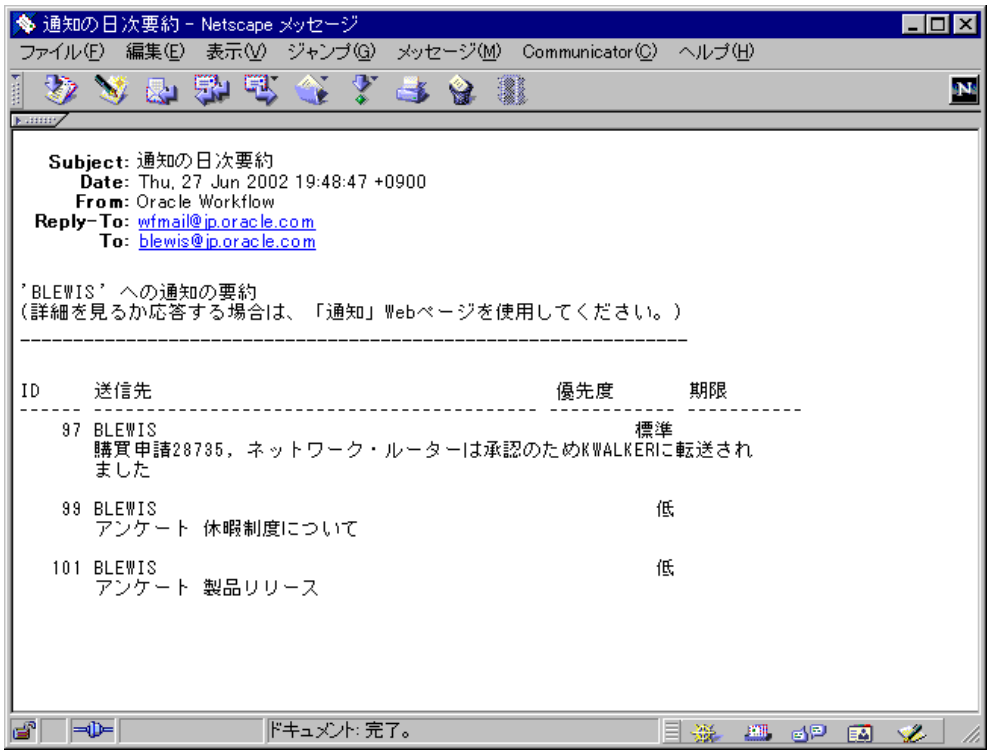
➤ **値リストの使用**

1. 値リストをサポートしているフィールドには、フィールドの上矢印のアイコンをクリックすると、値リスト・ウィンドウが表示されます。
2. 「検索」フィールドに検索基準を入力し、「検索」ボタンを選択して、基準と一致する値を検索します。「消去」ボタンを選択して、「検索」フィールドの内容を消去することもできます。検索基準を指定せずに「検索」のみを選択すると、完全な値リストが表示されます。
3. リストの値をクリックして選択し、値リスト・ウィンドウをクローズします。選択した値が元のフィールドに挿入されます。

## 電子メールによる通知要約の確認

「ユーザー設定項目」Web 画面で通知環境設定が「プレーン・テキスト要約メール」に設定されている場合は、ワークフロー通知の要約を 1 つの電子メール・メッセージとして受信できます。通知要約を受信する頻度は、通知要約に使用する通知メーラーに設定されている実行頻度によって決まります。2-52 ページの「[通知メーラーの起動](#)」を参照してください。

電子メール・ソフトウェアを使用して、電子メール通知の要約を受信できます。次の例では、Netscape Messenger をメール・クライアントとして、通知の要約を受信しています。



電子メール通知の要約は、Oracle Workflow Builder で定義されている標準テンプレートに基づいています。要約では、各通知の受信者、通知 ID、件名、優先度および期日が識別されます。2-65 ページの「[手順 WF-10 メッセージ・テンプレートの変更](#)」を参照してください。

また、通知の詳細を表示する場合や、通知に応答したり完了する場合は、「通知」Web 画面を使用する必要があることも表示されています。

## 自動通知処理ルールの定義

Oracle Workflow の自動通知処理を使用すると、休暇中のように通知を直接管理できない場合に、通知を他のロールに自動的に転送したり、事前定義済の応答を使用して着信通知に自動的に応答できます。

自動通知処理のルールは、「自動通知処理」Web 画面で定義できます。各ルールはロールに固有のものであり、特定の項目タイプやメッセージ名を持つ一部またはすべてのメッセージに適用できます。ルールの結果は、通知を他のユーザーに再割当てする、通知に応答するか通知をクローズする、元の宛先に通知を送信するのみでそれ以上のアクションはとらない、という 3 つのアクションのうちのいずれかになります。

通知システムで通知をロールに送信するか、再割当てを行うたびに、Oracle Workflow によって通知がそのロールのルール・リストと比較テストされ、基準に最も近いものが次の順序で検索されます。

ROLE = <role> で

1. MESSAGE\_TYPE = <type> および MESSAGE\_NAME = <name>
2. MESSAGE\_TYPE = <type> および MESSAGE\_NAME が NULL
3. MESSAGE\_TYPE が NULL および MESSAGE\_NAME が NULL

一致が見つかり、Oracle Workflow によってルールが適用され、それ以上ルールの照合は行われません。

ルールによって通知が再割当てされた場合は、新規受信者のロールのルール・リストに対してルール照合が再度実行されます。Oracle Workflow では、通知の転送回数が保存され、継続的な転送サイクルが検出されます。通知の自動転送回数が 10 回を超えると、転送サイクルが発生したものと見なされ、それ以上転送ルールは実行されず、通知にはエラー・マークが設定されます。

### ➤ 自動通知処理ルールを定義

1. Web ブラウザを使用して 2 つの URL の一方に接続します。

現行ロールのルーティング・ルール・リストを表示するには、次のように入力します。

```
<webagent>/wf_route.list[?user=<rolename>]
```

この URL では、大カッコ ([]) で示したようにオプションの引数を指定できます。オプションの引数を渡すときは、大カッコを省略してください。

<> 内のテキストを次のように置き換えます。

- <webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。
- <rolename> は、ルーティング・ルールを問い合わせる内部ロール名を表します。ただし、現行ロール以外のロールを問合せできるのは、現行ロールにワークフロー管



理者権限がある場合に限られるため注意してください。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

ワークフロー管理者権限がある場合は、Web 画面を表示して特定のロールのルーティング・ルールを検索できます。次のように入力します。

```
<webagent>/wf_route.find
```

ロールのユーザー ID を入力して「検索」を選択します。



---

**注意：** この2つの URL はセキュリティが適用されるページにアクセスするため、まだ現行の Web セッションに有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効ユーザーとしてのログオンを求めるプロンプトが表示されます。

---

---

**注意：** 「通知ルール」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---

2. ロールの「通知ルール」ページが表示され、現行ロールの既存ルールがすべて表示されます。「ルールの作成」を選択します。



3. 「項目タイプ」ドロップ・ダウン・フィールドで、このルールを適用する項目タイプを選択するか、このルールを任意の項目タイプに関連した通知に適用する場合は「<すべて>」を選択します。



4. 「次」を選択して先に進みます。このルールを取り消して前のページに戻る場合は「取消」を選択します。
5. ルールを適用する項目タイプとして「<すべて>」を選択した場合は、手順8にスキップします。特定の項目タイプを選択した場合は、次の手順に進み、ルールの適用対象となる項目タイプの通知を選択します。

6. 「通知」フィールドで、このルールを適用する通知メッセージを選択するか、このルールを項目タイプのすべての通知に適用する場合は、「<すべて>」を選択します。



7. 「次」を選択して先に進みます。このルールを取り消して前のページに戻る場合は「取消」を選択します。
8. 最後の「新規ルール作成」ページが表示されます。このページのフィールドは、作成したルールを適用する項目タイプおよび通知に応じて異なります。たとえば、ルールがすべての項目タイプに関連する場合は、すべての通知を別のユーザーに自動的に再割当てできますが、通知が異なれば応答属性も異なるため、すべての通知に対して自動応答を定義することはできません。

9. 「開始日付」および「終了日付」フィールドに値を入力して、このルールの有効期間を指定します。データベースのデフォルトの日付書式を使用して日付を指定し、デフォルトの日付書式に時間コンポーネントがない場合は、HH24:MI:SS の書式を使用して時刻を指定します。

「開始日付」を空白にすると、ルールは即時に有効になります。「終了日付」を空白にすると、ルールは失効日がないことになります。

---

**警告：** 同じ通知に対して、発効日の異なる複数のルールを定義できるため、「自動通知処理」Web 画面では、同じ通知に対して複数のルールを定義できます。同じ通知に関するルールの有効期間がオーバーラップしないように注意する必要があります。同じ通知に対して複数のルールが有効であれば、Oracle Workflow ではルールが 1 つランダムに選択されて適用されます。

---

10. 「通知に含めるコメント」フィールドに、ルール適用時に通知に追加するテキストを入力します。コメントは、通知の再割当てや自動的な応答が行われたときに、特別な「前のコメント」フィールドに表示されます。
11. このルールで実行する処理を選択します。
  - 「再割当て先」： 指定したロールに通知が転送されます。
  - 「応答」： 事前定義済の一連の応答値でメッセージに応答します。
  - 「一般的なルールに関係なく、通知を自分に送信」： 通知は受信ボックスに残り、処理は実行されません。この処理を指定してルールを定義すると、より包括的なルールから特定の通知のサブセットを除外できます。たとえば、すべての通知メッセージを別のロールに転送するルールがあり、そのルールから通知のサブセットを除外する場合を考えます。そのためには、その通知サブセットにのみ適用する新しいルールを定義し、処理として「一般的なルールに関係なく、通知を自分に送信」を指定します。
12. ルールの処理が「再割当て先」に設定されている場合は、上矢印のアイコンをクリックするとウィンドウが表示され、割当て先のロールを検索できます。10-24 ページの「[値リストの使用](#)」を参照してください。
13. ルールを選択した後、通知をどのように再割当てするかを指定します。この通知に応答する権限を新しいロールに付与する場合は、「通知に応答する許可を委任」を選択します。このオプションを選択すると、Oracle Workflow では、元の所有者が通知を所有したままですが、通知受信者のロールが通知の再割当て先となります。通知の所有権と職責を新規のロールに完全に与える場合は、「通知の所有権を譲渡」を選択します。

**注意：** ワークフロー管理者は、特別なロジックを実装し、通知の委任先または譲渡先のロールが適切であるかどうかを検証したり、通知の再割当てを制限できます。その場合、再割当てルールの作成時に警告メッセージを受け取ることもできます。
14. ルールの処理が「応答」である場合は、自動的に返信を行うための応答値を設定します。
15. 「OK」を選択し、このルールを保存して「通知ルール」Web 画面に戻り、ロールのルーティング・ルールを更新済リストを表示します。また、このルールを取り消して前のページに戻る場合は、いつでも「取消」を選択できます。

➤ **自動通知処理ルールの更新または削除**

1. 「自動通知処理」 Web 画面の URL に接続します。

`<webagent>/wf_route.list`

`<webagent>` は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

2. ロールの「自動通知処理」ページが表示されます。「ルールの適用結果」列で、更新するルールをクリックします。
3. 「ルールの変更」ページで、ルールを変更し、「OK」を選択して変更を保存します。  
「取消」を選択すると、変更を取り消して前のページに戻ることができます。
4. ルールを削除する場合は、「ルールの削除」列で、削除するルールの「X」を選択します。

---

## ワークフロー・プロセス監視

この章では、ワークフロー・プロセスのインスタンスを監視する方法について説明します。

## ワークフロー監視の概要

ある作業項目のワークフローを開始後に、そのステータスをチェックして先に進んでいるかどうかを確認したり、その作業項目に対して現在実行されているアクティビティを識別する必要があります。Oracle Workflow には、Java ベースのワークフロー・モニター・ツールと、WF\_ITEM\_ACTIVITY\_STATUSES\_V というビューが用意されており、ワークフロー・プロセスのインスタンスに関するステータス情報にアクセスできます。

---

---

**注意：** Oracle Applications Manager を実装している環境で Oracle Applications embedded Workflow を使用している場合は、Oracle Workflow Manager を追加管理ツールとして使用して作業項目を確認および管理することができます。詳細は、Oracle Applications Manager のオンライン・ヘルプを参照してください。

また、Oracle9i リリース 2 で利用可能なスタンドアロン版の Oracle Workflow を使用している場合は、Oracle Enterprise Manager から利用可能なスタンドアロン版の Oracle Workflow Manager コンポーネントを追加管理ツールとして使用して、作業項目を確認および管理できます。詳細は、Oracle Workflow Manager のオンライン・ヘルプを参照してください。

---

---

### 関連項目：

8-161 ページ [「Oracle Workflow のビュー」](#)

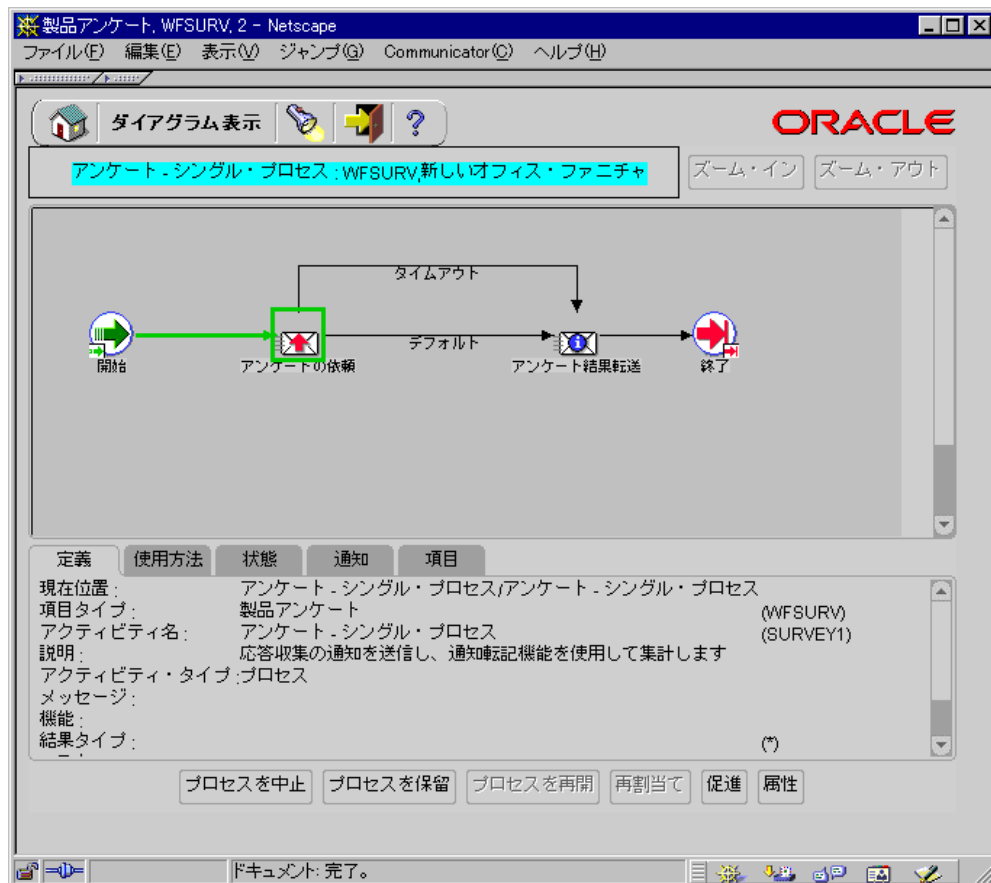
## ワークフロー・モニター

ワークフロー・モニターは、ワークフロー・プロセスの特定インスタンスのステータスを表示し、管理するためのツールです。ポイント・アンド・クリックのインタフェースを使用して、プロセスのアクティビティおよびプロセス全体についてステータスの詳細情報を表示できます。ワークフロー・モニターは、USER モードまたは ADMIN モードで実行できます。ADMIN モードでは、ワークフロー管理者にのみ関係のある追加の詳細情報や機能が提供されます。11-8 ページの [「ワークフロー・モニターへのアクセス」](#) を参照してください。

ワークフロー・モニターは、次のセクションで構成されています。

- プロセス・タイトル
- プロセス・ダイアグラム・ウィンドウ
- 詳細タブ・ウィンドウ
- 管理ボタン





## プロセス・タイトル

プロセス・タイトルは、ワークフロー・モニターの左上に表示され、ワークフロー・プロセス名と、そのプロセスの実行中のインスタンスをプロセス・ダイアグラム・ウィンドウで一意に識別する項目タイプ名およびユーザー・キーが表示されます。ユーザー・キーが設定されていない場合は、かわりに項目キーが表示されます。プロセス・ダイアグラム・ウィンドウでサブプロセスにドリルダウンすると、プロセス・タイトルが更新されてサブプロセス名が表示されます。

プロセス・ダイアグラム・ウィンドウで選択済のアクティビティを選択解除したり、詳細タブ・ウィンドウでそのプロセスやサブプロセス全体の情報を表示するには、プロセス・ダイアグラム・ウィンドウの余白をクリックします。

プロセス・ダイアグラム・ウィンドウ

プロセス・ダイアグラム・ウィンドウは、プロセス・タイトルに現在表示されているワークフロー・プロセスやサブプロセスのダイアグラムを表示する、スクロール可能なキャンバスです。このダイアグラムは、Oracle Workflow Builder で作成するダイアグラムとまったく同じです。ただし、ワークフロー・モニターを使用してこのダイアグラムを編集することはできないため注意してください。

プロセス・ダイアグラム・ウィンドウには、プロセスやそのアクティビティのステータスがグラフィカルに示されます。

- アクティビティ・アイコンは、それが要注意状態にあることを示すために、色付きのボックスでハイライト表示される場合があります。次の表は、それぞれの色が表す状態を示しています。

表 11-1

ボックスの色	状態	考えられるステータス・コード
赤	エラー	ERROR
緑	アクティブ / 処理中	ACTIVE、NOTIFIED、DEFERRED
黄	保留中	HOLD
< なし >	正常	COMPLETE、WAITING、NULL

- すでに通過したトランジション（矢印）は緑の太線で表示され、まだ通過していないトランジションは黒の細線で表示されます。
- アクティビティを選択して、そのアクティビティに関する情報を表示する詳細タブ・ウィンドウを更新するには、ダイアグラムのアクティビティ・アイコンをクリックします。
- 現在選択されているアクティビティ・アイコンを選択解除し、現在のプロセス全体の情報を表示する詳細タブ・ウィンドウの表示を更新するには、プロセス・ダイアグラム上の空白部分をクリックします。
- サブプロセスのダイアグラムにドリルダウンするには、サブプロセスを示すアクティビティ・アイコンをダブルクリックします。この操作により、プロセス・タイトルが自動的に更新されてサブプロセス名が反映され、そのサブプロセス全体の情報を表示する詳細タブ・ウィンドウが更新されます。

かわりに、サブプロセス・アクティビティを選択し、「ズーム・イン」を選択して、サブプロセスのダイアグラムにドリルダウンすることもできます。直前レベルのプロセスに戻るには、「ズーム・アウト」を選択します。

## 詳細タブ・ウィンドウ

詳細タブ・ウィンドウは、プロセス・ダイアグラムの下に表示される上下にスクロール可能な表示領域で、選択されたプロセスやアクティビティに関する情報が表示されます。

各タブには、次のような情報が表示されます。アスタリスク (\*) で始まる行や、太字体のカッコ () 内の値は、モニターが ADMIN モードで実行されている場合のみ表示されます。

### 「定義」タブ

現在位置:	プロセスの表示名 / アクティビティの表示名	
項目タイプ:	項目タイプの表示名	(内部名)
アクティビティ名:	アクティビティの表示名	(内部名)
説明:	アクティビティの説明	
アクティビティ・タイプ:	プロセス、通知、イベントまたは関数	
メッセージ:	メッセージの内部名	
関数:	アクティビティによってコールされる PL/SQL プロシージャ名	
結果タイプ:	結果タイプの表示名	(内部名)
* コスト:	関数アクティビティの秒単位のコスト	
* 再開封時:	無視、ループまたはリセット	
* エラー・プロセス:	アクティビティに割り当てられたエラー項目タイプおよびエラー・プロセスの内部名 (存在する場合)	

### 「使用方法」タブ

現在位置:	アクティビティの表示名	
開始 / 終了:	いいえ、開始または終了	(プロセスの結果)
実行者:	ロール名または項目属性の内部名	
* コメント:	プロセス・アクティビティ・ノードのコメント	
タイムアウト:	N 分または項目属性の内部名	

### 「状態」タブ

現在位置:	アクティビティの表示名	
ステータス:	アクティビティのステータス	
結果:	アクティビティの結果	(結果コード)
開始日:	アクティビティが開始する日	

終了日:	アクティビティが終了する日
期日:	アクティビティの期限が切れる日
* 通知:	通知 ID
割り当てられたユーザー:	ロール名または項目属性の内部名（「アクティビティ・ステータス」が「エラー」の場合にのみ表示）
* エラー名:	エラーの名前
エラー・メッセージ:	エラー・メッセージ
* エラー・スタック:	エラー・スタック

「通知」タブ

現在位置:	アクティビティの表示名
* ID:	通知 ID
宛先:	通知の宛先
ステータス:	通知のステータス
開始日:	通知が配信される日
終了日:	通知が完了する日
期日:	アクティビティの期限が切れる日

(選択したアクティビティが応答の必要な通知である場合、このタブには前述の情報が表示されるかわりに、メッセージ応答属性が、  
<message\_attribute> <type (Format)> <value> のように表示されます。選択したアクティビティがポーリング・タイプの通知アクティビティで、「ロールの拡張」がオンで、応答が必要である場合、このタブには宛先ごとに前述のメッセージ応答属性が表示されます。選択したアクティビティが通知アクティビティで、「ロールの拡張」はオンになっているが、応答は不要な場合、ロール内の個々のユーザーではなく、ロールのみが宛先として表示されます。)

「項目」タブ

プロセス表示名:	項目タイプ、項目キー（または、設定されている場合はユーザー・キー）
所有者:	まだ実装されていない項目の所有者
開始日:	ワークフロー・プロセス・インスタンスが作成される日
終了日:	ワークフロー・プロセス・インスタンスが完了する日
<Item Attribute>:	<type (format)> <value>
...	

## 管理ボタン

管理ボタンは、ワークフロー・モニターが ADMIN モードで実行されている場合に限り、詳細タブ・ウィンドウの下に表示されます。各ボタンを使用して該当する Workflow Engine API をコールし、異なる管理操作を実行できます。ボタンとその動作は次のとおりです。

- プロセスを中止： プロセス・タイトルまたはプロセス・アクティビティを選択した場合にのみ使用できます。WF\_ENGINE.AbortProcess をコールして選択したプロセスを中止し、未処理の通知を取り消します。中止するプロセスに割り当てる結果の入力を求めるプロンプトが表示されます。そのプロセスは、結果が指定どおりになると、ステータスが「完了」になります。8-39 ページの「[AbortProcess](#)」を参照してください。
- プロセスを保留： プロセス・タイトルまたはプロセス・アクティビティを選択した場合にのみ使用できます。それ以上アクティビティが進行しないように、WF\_ENGINE.SuspendProcess をコールし、選択したプロセスを保留します。8-35 ページの「[SuspendProcess](#)」を参照してください。
- プロセスを再開： 保留プロセスを選択した場合にのみ使用できます。WF\_ENGINE.ResumeProcess をコールし、保留したプロセスを再開して正常な実行ステータスにします。プロセスが保留になった時点で移動したアクティビティが実行されます。8-37 ページの「[ResumeProcess](#)」を参照してください。
- 再割当て： 通知アクティビティを選択した場合にのみ使用できます。WF\_ENGINE.AssignActivity をコールして、通知アクティビティを異なる実行者に再割当てします。ロール名の入力を求めるプロンプトが表示されます。8-76 ページの「[AssignActivity](#)」を参照してください。
- 促進： プロセス・タイトル、つまりアクティビティを選択した場合にのみ使用できます。WF\_ENGINE.HandleError をコールし、エラーのあるアクティビティの状態を変更するか、または選択したアクティビティおよび後続の他のすべてのアクティビティを取り消して、プロセスの一部をロールバックします。「スキップ」を選択して該当アクティビティをスキップし、それを指定した結果に割り当てるか、または「再試行」を選択してアクティビティを再実行するように求めるプロンプトが表示されます。8-79 ページの「[HandleError](#)」を参照してください。
- 属性： 項目タイプ属性の値を変更できるように、常時使用可能です。各項目タイプ属性の現行値が表示されます。値の変更後に、「OK」を選択して変更を適用します。

## ワークフロー・モニターへのアクセス

ワークフロー・モニターへのユーザー・アクセスを制御するには、2通りの方法があります。ワークフロー・モニターへのアクセスを制御するワークフロー対応アプリケーションを使用する方法と、「プロセス検索」Web 画面に直接アクセスする方法です。

### アプリケーションによるワークフロー・モニターへのアクセス制御

アプリケーション・コードのロジック内で、ユーザーが表示できるワークフロー・プロセスのインスタンスと、そのユーザーに対してモニターを ADMIN モードで実行するか、USER モードで実行するかを識別します。また、アプリケーションのユーザー・インタフェースには、Java 1.1.8 と AWT をサポートする Web ブラウザ・アプリケーションをコールし、WF\_MONITOR.GetDiagramURL() 関数からの戻り値を取得するワークフロー・モニターの URL を渡す手段を用意する必要があります。戻される URL には、ADMIN または USER モードのどちらかでワークフロー・モニターにアクセスできるように、非表示のパスワードが添付されています。8-155 ページの「[GetDiagramURL](#)」を参照してください。

### 「プロセス検索」Web 画面へのアクセスの提供

ワークフロー・モニターにアクセスするには、Java 1.1.8 と AWT をサポートする Web ブラウザに、「プロセス検索」の URL を渡す方法もあります。「プロセス検索」ページにアクセスするには、ユーザー認証が必要です。Oracle Workflow が Oracle Self-Service Web Applications と Oracle HTTP Server のどちらのセキュリティを使用するように構成されているかに応じて、ユーザーが現行ブラウザ・セッションで有効なユーザーとしてログインしていない場合は、適切なユーザー名とパスワードを使用してログインする必要があります。ユーザーがワークフロー管理者権限を持っている場合は、「プロセス検索」Web 画面を使用してワークフロー・プロセスのインスタンスを検索できます。ワークフロー管理者権限を持っていないユーザーの場合、「プロセス検索」Web 画面を使用して検索できるのは、ワークフロー・エンジンの SetProcessOwner API のコールによって設定されたユーザー自身のプロセスに限られます。ユーザーは、そのプロセス・インスタンスの通知またはプロセス・ダイアグラムをワークフロー・モニターで表示できます。

「プロセス検索」の URL は、次のようになります。

```
<webagent>/wf_monitor.find_instance
```

<webagent> は、WF\_CORE.TRANSLATE() をコールして、WF\_RESOURCES 表の WF\_WEB\_AGENT トークンから取り出すことができる Web エージェントの文字列です。8-113 ページの「[TRANSLATE](#)」を参照してください。

---

**注意：**「プロセス検索」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---

## Oracle Applications からワークフロー・モニターへのアクセス

Oracle Applications の場合にのみ、Oracle Applications の関数 FND\_FNDWFIAS を使用して、アプリケーションにワークフロー・モニターへのアクセスを追加できます。この関数は、Web 画面の wf\_monitor.show をコールします。

この関数は、ユーザーの職責の「ナビゲート」メニューに追加する方法と、Oracle Applications フォームからコールする方法があります。『Oracle Applications Developer's Guide』の「Menus Window」および「Overview of Form Development Steps」を参照してください。

パラメータを渡さずに FND\_FNDWFIAS をコールすると、wf\_monitor.show により「プロセス検索」Web 画面が表示されます。

表示するプロセス・インスタンスを指定する場合は、この関数に次のパラメータを渡す必要があります。

- ITEM\_TYPE: 有効な項目タイプ。
- ITEM\_KEY: アプリケーション・オブジェクトの主キーから導出される文字列。この文字列により、項目タイプの項目が一意に識別されます。項目タイプと項目キーにより、プロセスが識別されます。
- ADMIN\_MODE: 「YES」の場合はモニターが ADMIN モードで実行され、「NO」の場合は USER モードで実行されます。
- ACCESS\_KEY: 選択したモードでモニターを実行するためのアクセス・キー・パスワード。適切なアクセス・キーを取り出すには、ワークフロー・モニター API の GetAccessKey() を使用します。8-154 ページの「GetAccessKey」を参照してください。

これらのパラメータを渡すと、wf\_monitor.show では、指定したプロセス・インスタンスについてワークフロー・モニターの「通知リスト」Web 画面が表示されます。

関数 FND\_FUNCTION.EXECUTE をコールすると、パラメータを指定して FND\_FNDWFIAS を実行できます。『Oracle Applications Developer's Guide』の「FND\_FUNCTION.EXECUTE」を参照してください。構文は、次のとおりです。

```
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>' ));
```

関数 FND\_FNDWFIAS では、Oracle Applications の機能セキュリティを使用してユーザー・アクセスが制御されます。『Oracle Applications System Administrator's Guide』の「Overview of Function Security」および『Oracle Applications Developer's Guide』の「Overview of Menus and Function Security」を参照してください。

---

---

**注意：** Oracle Applications では、関数 FND\_UTILITIES.OPEN\_URL をコールして Web ブラウザを開き、「プロセス検索」の URL やワークフロー・モニターのダイアグラムの URL など、指定した URL に接続することもできます。ただし、この関数では Oracle Applications の機能セキュリティは使用されません。『Oracle Applications Developer's Guide』の「FND\_UTILITIES:Utility Routine」を参照してください。

---

---

### 「プロセス検索」Web 画面の使用

Oracle Workflow の「プロセス検索」Web 画面を使用すると、特定の検索基準と一致するワークフロー・プロセス・インスタンスのリストを問合せできます。表示される「プロセス・リスト」からプロセス・インスタンスの 1 つを選択し、より詳細に検討できます。

表示される「通知リスト」Web 画面で、完了した通知アクティビティの詳細を確認できます。また「拡張オプション」を選択し、その他のアクティビティの詳細を「アクティビティ・リスト」Web 画面に表示できます。「通知リスト」からワークフロー・モニターにナビゲートすると、プロセスの管理や、プロセスとそのステータスのグラフィカルな表示も可能です。



▶ 「プロセス検索」 ページでの検索基準の指定

1. 次のフィールドを任意に組み合わせて検索基準を入力し、ワークフロー・プロセスのインスタンスのサブセットを検索できます。
  - プロセス・ステータス：「すべての状態」、「有効」または「完了」を指定します。
  - 項目タイプ： 検索するワークフロー・プロセスのインスタンスの項目タイプを選択します。すべての項目タイプのワークフロー・プロセス・インスタンスを検索するには、「すべて」を選択します。
  - 項目キーまたはユーザー・キー： 項目キーまたはユーザー・キーを指定します。項目キーは項目の内部識別子を表し、ユーザー・キーは項目に割り当てられたエンド・ユーザー識別子を表します。ユーザー・キーは項目に対して一意でなくてもかまいません。8-25 ページの「[SetItemUserKey](#)」を参照してください。
  - プロセス名： プロセス・アクティビティの表示名を指定します。
2. ワークフロー管理者権限を持つユーザーとしてログオンした場合は、プロセスを所有していなくても、任意のプロセスのインスタンスを検索して表示できます。「プロセス所

有者」フィールドに WF\_ROLES で定義されているロールの内部名を入力すると、そのロールが所有するプロセスのみが表示されます。また、このフィールドを空白のままにすると、プロセスの所有者に関係なく、検索基準と一致するすべてのプロセスのインスタンスが表示されます。

ワークフロー管理者権限を持っていない場合、「プロセス所有者」フィールドには、現行の Web セッションへのログイン時に使用したロールの内部名が表示されます。この場合に検索して表示できるのは、ユーザー自身が実行したプロセス、または最初の参加者に限られます。

**注意：** WF\_ENGINE.SetItemOwner API をコールすると、プロセスの所有者を設定できます。プロセスの所有者は、そのプロセスを実行したユーザー、またはそのプロセスの最初の参加者です。

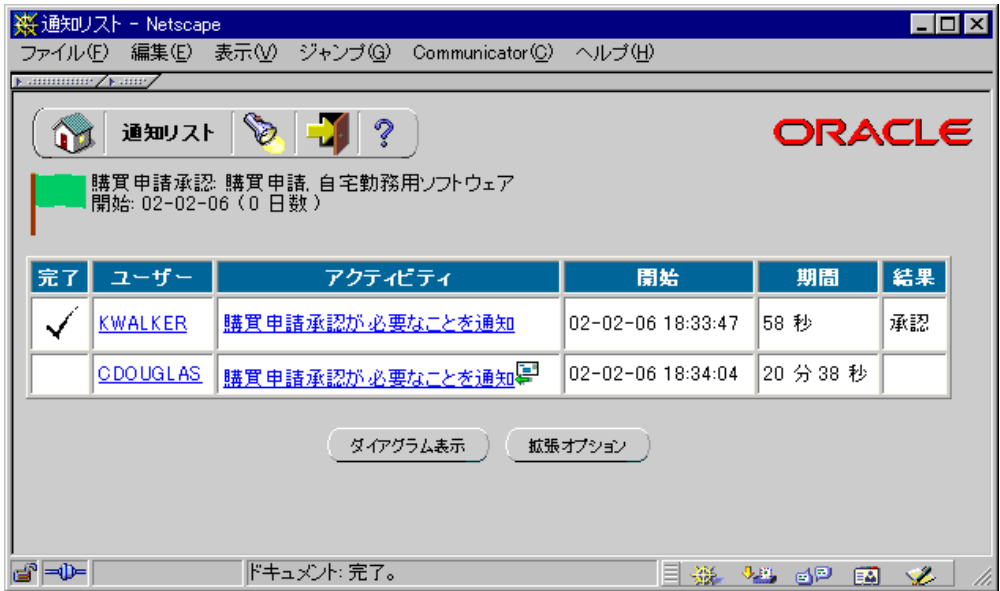
- 3. オプションで、アクティビティが「中止」、「エラー」または「すべての状態」になっているワークフロー・プロセス・インスタンスも検索できます。
- 4. 特定のユーザーまたはロールからの応答待ちになっているアクティビティを含むワークフロー・プロセス・インスタンスを検索できます。
- 5. 指定した日数の間進行していないワークフロー・プロセス・インスタンスも識別できます。
- 6. 検索基準の入力終了後に「検索」を選択します。基準と一致するすべてのプロセス・インスタンスが「プロセス・リスト」Web 画面に表示されます。

➤ 「プロセス・リスト」のレビュー



- 1. 「プロセス・リスト」に、「プロセス検索」 Web 画面で指定した検索基準と一致するすべてのワークフロー・プロセス・インスタンスの要約が表示されます。
- 2. プロセス・インスタンスは、最初に項目タイプ別、次に項目キー別に昇順で表示されます。
- 3. 「プロセス・リスト」には、各プロセス・インスタンスのステータスが要約されて、「完了」、「エラー」および「中止」の各列に示されます。
- 4. 「プロセス名」列でプロセス・リンクを選択し、そのプロセス・インスタンスの通知リストを表示します。

▶ 「通知リスト」のレビュー



- 1. 「通知リスト」には、選択したプロセス・インスタンスの現在の送信済通知で、特殊な結果応答が必要なものがすべて表示されます。つまり、これらはプロセスが宛先の応答に基づいて分岐できる通知アクティビティです。
- 2. 「通知リスト」には、各通知アクティビティの内容、それが割り当てられている宛先、送信された日時、完了の有無、完了までに経過した日数、その結果が要約されています。

**注意：** プロセス自体がエラー状態であり、エラーの原因が通知であった場合は、その通知の結果が「結果」列にリンクとして表示されることがあります。そのリンクを選択するとエラーの原因が表示されます。

- 通知が割り当てられているユーザーに電子メールを送信する場合は、「ユーザー」列でユーザー・リンクを選択します。

---

**注意：**「通知リスト」Web 画面のリンクにカーソルをあわせると、リンクについて役に立つヒントが表示されます。ヒントは Web ブラウザのステータス・バーに表示されます。

---

- 通知アクティビティの完全な定義を参照する場合は、「アクティビティ」列にある通知アクティビティ・リンクを選択します。
- 通知アクティビティがまだ開いていて、応答が必要な場合は、ワークフロー管理者権限でログオンしていると、「アクティビティ」列の通知アクティビティ名の後ろにアイコンが表示されます。このアイコンをクリックすると、「通知の詳細」ページにジャンプし、この通知に直接応答できます。応答の完了後に、ブラウザの「戻る」ボタンを選択して、「通知リスト」に戻ります。
- 「拡張オプション」を選択して「アクティビティ・リスト」Web 画面にジャンプすると、拡張基準を指定し、プロセスの対象となる特定アクティビティを検索して表示できます。11-15 ページの「[「アクティビティ・リスト」のアクティビティのフィルタ](#)」のアクティビティのフィルタ」を参照してください。
- 「ダイアグラム表示」ボタンを選択して、選んだプロセス・インスタンスをワークフロー・モニターに表示し、プロセス・ステータスをグラフィカルに表示できます。現在の Web セッションにワークフロー管理者権限を持つユーザーとして接続している場合、ワークフロー・モニターにはプロセスが ADMIN モードで表示され、それ以外の場合は USER モードで表示されます。11-2 ページの「[ワークフロー・モニター](#)」を参照してください。

---

**注意：** 選択したプロセスが親 / 子プロセスのメンバーの場合は、左側に親 / 子階層リストが表示されます。この階層リストには、現行のプロセス・インスタンスに対応する親および子インスタンスへのリンクが表示されます。リンクをクリックすると、選択した親または子インスタンスの「通知リスト」が起動します。

---

▶ 「アクティビティ・リスト」のアクティビティのフィルタ



1. 「アクティビティ・リスト」Web 画面では、対象の特定アクティビティをフィルタするために、様々な基準を指定できます。
2. 「アクティビティ状態オプション」チェック・ボックスで、対象のアクティビティ・ステータスを指定します。「アクティブ」のステータスには、「通知済」、「遅延」、「待機」ステータスのアクティビティも含まれます。
3. 「アクティビティ・タイプ」チェック・ボックスで、表示するアクティビティのタイプを指定します。応答の必要な通知アクティビティ、応答の不要な通知アクティビティ、機能アクティビティ、標準項目タイプに属するアクティビティ、イベント・アクティビティのうち、いずれかまたはすべてを表示するように選択できます。
4. 基準の選択の完了後に「アクティビティのフィルタ」を選択し、基準と一致するアクティビティを表示します。

5. 結果のアクティビティ要約リストには、次の情報の列があります。

- 状態： アクティビティのステータス。「有効」、「完了」、「エラー」、「中止」のいずれかです。
- ユーザー： アクティビティの実行者。アクティビティが関数アクティビティの場合は、ワークフロー・エンジンが実行者です。実行者がユーザーの場合は、そのユーザー名へのリンクをクリックすると、そのユーザーにメールを送信できます。アクティビティが通知アクティビティで、「ロールの拡張」がオンになっている場合は、要約にそれと同じアクティビティが複数行表示され、「ユーザー」列には、ロールの各メンバーが表示されます。
- 上位アクティビティ： このアクティビティが属するプロセス・アクティビティ（このアクティビティ自体が最上位レベルのプロセスである場合を除きます）。上位アクティビティは、その定義の詳細にリンクしています。
- アクティビティ： アクティビティの名前。このアクティビティは、その定義の詳細にリンクしています。
- 開始： アクティビティが開始した日時。
- 期間： アクティビティの完了にかかった時間。かかった時間の最上位単位より 1 つ下の単位で示されます。アクティビティの完了に 1 分からなかった場合は、秒数のみが示されます。
- 結果： アクティビティの結果。アクティビティのステータスが「エラー」の場合は、結果がそのエラーに関連するエラー名、エラー・メッセージおよびエラー・スタックにリンクします。

---

---

**注意：**「アクティビティ・リスト」Web 画面のリンクにカーソルをあわせると、リンクについて役に立つヒントが表示されます。ヒントは Web ブラウザのステータス・バーに表示されます。

---

---

6. 任意の列のヘッダーをクリックすると、その列を基準にしてアクティビティ要約リストをソートできます。列タイトルの横に、ソートに使用されていることを示すアスタリスク (\*) が表示されます。アスタリスクが列タイトルの左にある場合、ソート順は昇順です。アスタリスクが列タイトルの右にある場合、ソート順は降順です。同じ列タイトルを複数回クリックすると、ソート順が逆になります。
7. 「ダイアグラム表示」ボタンを選択し、プロセス・インスタンスをワークフロー・モニターに表示し、プロセス・ステータスをグラフィカルに表示することもできます。現在の Web セッションにワークフロー管理者権限を持つユーザーとして接続している場合、ワークフロー・モニターにはプロセスが ADMIN モードで表示され、それ以外の場合は USER モードで表示されます。11-2 ページの「ワークフロー・モニター」を参照してください。

**関連項目：**

2-20 ページ「[手順 WF-3 Oracle Workflow のディレクトリ・サービスの設定](#)」

2-13 ページ「[グローバル・ワークフロー・プリファレンスの設定](#)」





---

## ワークフロー定義のテスト

この章では、Oracle Workflow の「プロセスの開始」Web 画面を使用してワークフロー定義をテストする方法について説明します。

## ワークフロー定義のテスト

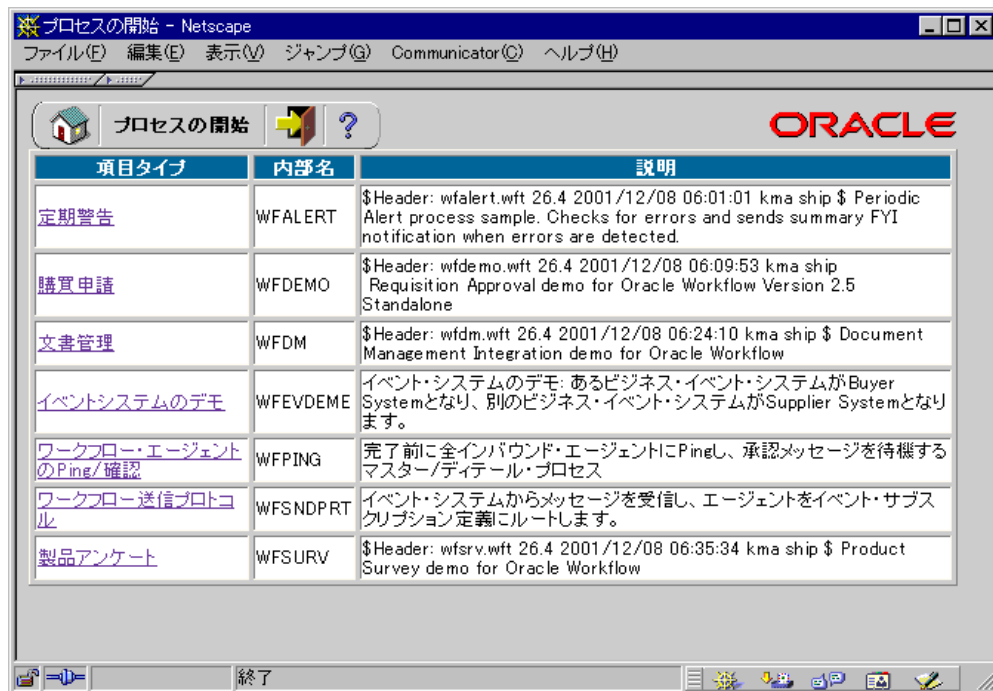
Oracle Workflow には「プロセスの開始」と呼ばれる Web ベースのインタフェースがあり、これを使用して、データベースに定義し保存したあらゆるワークフロー定義をテストします。「プロセスの開始」にアクセスできるのは、ワークフロー管理者のロールに属しているユーザーのみです。

あらゆる Oracle Workflow データベースに対して「プロセスの開始」Web 画面を実行できますが、テスト目的の場合は別の環境を作成することをお勧めします。ワークフロー定義をテストするには、テスト環境で次の項目を設定する必要があります。

- テストするユーザーとロールを定義します。Oracle Workflow のデモンストレーション・データ・モデルで事前定義されているユーザーとロールに対してテストできます。15-6 ページの「[購買申請データ・モデルのインストール](#)」を参照してください。
- Oracle Workflow のスタンドアロン版を使用していて、通知の表示に「通知」Web 画面を使用する場合は、テストするユーザーとロールを Web セキュリティ・システムで定義する必要があります。詳細は、使用している Web サーバーのドキュメントを参照してください。
- 通知の表示に電子メールを使用する場合は、TEST\_ADDRESS パラメータを通知メーラーの構成ファイルに設定し、すべての通知を 1 つのテスト用電子メール・アドレスに送ります。2-55 ページの「[通知メーラーの構成ファイルの作成](#)」を参照してください。

### ▶ ワークフロー定義のテスト

1. Web ブラウザを使用して、次の URL で Oracle Workflow ホーム・ページに接続します。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。
2. 「プロセスの開始」リンクを選択し、「プロセスの開始」Web 画面を表示します。



**注意：** 保護されている次の URL を使用し、直接このページに接続する方法もあるので注意してください。

`<webagent>/wf_initiate.ItemType`

`<webagent>` は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。2-13 ページの「グローバル・ワークフロー・プリファレンスの設定」を参照してください。

**注意：** このページにはセキュリティが適用されるため、現行 Web セッションで有効なワークフロー管理者としてログオンしていない場合は、ページが表示される前に有効なワークフロー管理者としてのログオンを求めるプロンプトが表示されます。

- 「プロセスの開始」ページには、「システム：エラー」、「システム：メーラー」および「標準」など、Oracle Workflow がシードした項目タイプを除き、データベースに格納されている項目タイプ定義がすべて表示されます。各項目タイプの内部名と説明も表示

されます。テストするワークフロー・プロセスの定義を所有する項目タイプを選択します。

ワークフローの開始 - Netscape

ファイル(F) 編集(E) 表示(V) ジャンプ(G) Communicator(C) ヘルプ(H)

ワークフローの開始 - WFSURV

ORACLE

項目キー

ユーザー・キー

プロセス名

プロセス所有者

アンケートの参加者

タイムアウト(分)

文書ID

アンケート名

個別の参加者

OK 取消

ドキュメント: 完了。

4. 「ワークフローの開始」Web 画面を使用して、開始するプロセスの詳細を指定します。ワークフロー・プロセスのインスタンスを実行するには、次の項目を指定する必要があります。

- プロセス・インスタンスの一意の項目キー
- プロセスの識別に使用するユーザー定義のキー
- テストするプロセスの名前
- オプションのプロセス所有者
- プロセスの項目タイプに関連した、あらゆる項目タイプ属性の値

「OK」を選択します。ワークフロー・プロセスを開始するために、「ワークフローの開始」Web 画面では、指定した項目タイプや項目キーに対して、ワークフロー・エンジンの `CreateProcess` や `Startprocess` などの API がコールされます。ワークフロー・エンジンの `SetItemOwner` や `SetItemAttr` などの API もコールされ、プロセス所有者とすべての項目タイプ属性が指定の値に設定されます。

5. 実行したプロセス・インスタンスに対するワークフロー・モニターのアクティビティ・リストが表示されます。アクティビティ・リストには、実行されているアクティビティのステータスが示されます。「ダイアグラム表示」ボタンを選択し、ワークフロー・モニターにプロセスのステータスをグラフィカルに表示することもできます。11-15 ページの「[「アクティビティ・リスト」のアクティビティのフィルタ](#)」を参照してください。
6. テストしているプロセスに通知が含まれている場合は、Workflow ホーム・ページに戻り、「通知の検索」リンクを選択して、プロセスを完了するために応答を必要とする未処理の通知を検索します。また、通知の応答を電子メールでテストする場合は、通知メーラーで指定したテスト用電子メール・アカウントに接続し、プロセスの未処理の通知に応答します。



---

## ビジネス・イベントの管理

この章では、Oracle Workflow の「イベント・マネージャ」Web 画面を使用してビジネス・イベントを管理する方法について説明します。

## ビジネス・イベントの管理

Oracle Workflow Business Event System は、Oracle Advanced Queuing (AQ) のインフラストラクチャを利用して、システム間でビジネス・イベントを交換するためのアプリケーション・サービスです。ビジネス・イベント・システムは、イベント・マネージャとワークフロー・プロセス・イベント・アクティビティで構成されます。

イベント・マネージャには、ビジネス・イベント、システム、システム内の名前付き通信エージェント、およびイベントと特定のシステムの関連付けを示すサブスクリプションが登録されています。イベントはローカルで呼び出すか、または AQ を介して外部システムまたはローカル・システムから受信します。ローカル・イベントが発生すると、サブスクリプションが遅延されていない限り、イベントを呼び出したコードと同じトランザクション内でサブスクライバのコードが実行されます。

サブスクリプションには、次のタイプの処理を指定できます。

- イベント情報に対するカスタム・コードの実行
- ワークフロー・プロセスへのイベント情報の送信
- 他のキューまたはシステムへのイベント情報の送信

ビジネス・イベントは、ワークフロー・プロセス内ではイベント・アクティビティとして表されます。イベント・アクティビティをワークフロー・プロセスに含めることにより、事前定義済の関数を直接実行したり、事前定義済の受信者にイベントを送信するだけでなく、ビジネス・イベントに関する複雑な処理またはルーティング・ロジックをモデル化できます。4-42 ページの「[イベント・アクティビティ](#)」を参照してください。

ビジネス・イベント・システムには、次のような機能があります。

- **システム統合メッセージ・ハブ：** Oracle Workflow とビジネス・イベント・システムを統合すれば、複雑なシステム統合シナリオのメッセージ・ハブとして機能します。イベント・マネージャは、イベントおよび呼び出し元に基づいてシステム間のルーティングを密接に連結するために使用できます。ワークフロー・プロセス・イベント・アクティビティは、より高度なルーティング、コンテンツベースのルーティング、変換、エラー処理などのモデル化に使用できます。
- **分散アプリケーション・メッセージ機能：** アプリケーションは、ビジネス・エンティティに対して Generate および Receive イベント・メッセージ・ハンドラを提供できます。たとえば、メッセージ・ハンドラを使用して、マスター / コピーのレプリケーションを分散アプリケーションに実装できます。
- **メッセージ・ベースのシステム統合：** ビジネス・イベントの発生時に、システム間にメッセージが送信されるようにサブスクリプションを設定できます。このように、イベント・マネージャを使用してポイントツーポイントのメッセージ統合を実現できます。
- **ビジネス・イベント・ベースのワークフロー・プロセス：** ビジネス・イベントの内容に基づいた拡張ルーティングまたはプロセスを組み込んで、高度なワークフロー・プロセスを開発できます。



- **パッケージ・アプリケーションの非侵入性カスタマイズ:** 分析者は、インターネットまたはイントラネット・アプリケーションに対して、興味あるビジネス・イベントを登録できます。アプリケーションのユーザーは、それらのイベントへのサブスクリプションを登録して、カスタム・コードまたはワークフロー・プロセスをトリガーできます。

## イベント・マネージャ

Oracle Workflow のイベント・マネージャには、アプリケーションで発生する興味あるビジネス・イベント、イベントを交換するシステム、それらのシステム内の名前付き通信エージェント、およびイベントと特定のシステムの関連付けを示すサブスクリプションを登録できます。「イベント・マネージャ」Web 画面を使用して、これらのイベント、システム、エージェントおよびサブスクリプションを定義およびメンテナンスできます。

---

---

**注意:** 「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

---

---

Workflow XML Loader を使用して、フラット・ファイルとデータベース間でビジネス・イベント・システムオブジェクトの XML 定義をアップロードおよびダウンロードすることができます。2-106 ページの「[Workflow XML Loader の使用](#)」を参照してください。

ローカル・アプリケーションからイベントが呼び出されたとき、またはローカル・システムあるいは外部システムからイベントが着信したときは、イベント・マネージャがそのイベントへのサブスクリプションを実行します。イベント・マネージャは、サブスクリプションに定義されているアクションに応じて、カスタム・コードのコール、ワークフロー・プロセスへのイベント情報の送信、またはエージェントへのイベント情報の送信を行います。

また、イベント・マネージャを使用して、インバウンド・イベント・メッセージのリスナーおよびアウトバウンド・イベント・メッセージの伝播のスケジュールなど、メッセージ伝播を設定することができます。

ビジネス・イベント・システムの設定が完了したら、イベント・マネージャを使用して、イベントの手動呼出し、ビジネス・イベントを交換するためのシステムのサインアップ、システム間の同期およびローカル・キューの確認を行うことができます。「ワークフロー・エージェントの Ping/ 確認」を使用して、設定をテストできます。

## イベント

ビジネス・イベントは、システム内の他のオブジェクトまたは外部エージェントに関連付けられた、インターネットまたはイントラネット内の状態変化です。たとえば、発注書の作成は、購買アプリケーションにおけるビジネス・イベントの例です。イベントの関連付けは、イベント・マネージャで定義できます。

Oracle Workflow には、ビジネス・イベント・システム内で関連付けられた状態変化に対して、いくつかの事前定義済イベントが用意されています。14-2 ページの「[事前定義済ワークフロー・イベント](#)」を参照してください。

ローカル・システム上のアプリケーションでイベントが発生する場合は、イベントの特定のインスタンスを一意に識別するイベント・キーを割り当ててから、イベント・マネージャに送信する必要があります。

イベントは次の方法で呼び出すことができます。

- イベントを呼び出すアプリケーションから `WF_EVENT.Raise()` API を使用して、イベントを呼び出します。8-268 ページの「[Raise](#)」を参照してください。
- ワークフロー・プロセスから `Raise` イベント・アクティビティを使用して、イベントを呼び出します。4-42 ページの「[イベント・アクティビティ](#)」を参照してください。
- 「[Raise Events](#)」画面から、イベントを手動で呼び出します。13-71 ページの「[イベントの呼出し](#)」を参照してください。

また、イベント・マネージャでは、ローカル・システムまたはリモート・システムから送信されたイベントを受信できます。

イベント・マネージャにイベントを定義する際に、一意の内部名をイベントに割り当てる必要があります。この名前は大文字 / 小文字が区別されます。イベントの内部名を記述するときには、次のように複数の識別子をピリオドで区切った書式を使用することをお勧めします。

`<company>.<family>.<product>.<component>.<object>.<event>`

この書式では、定義するイベントを分類階層に体系化することができます。

また、イベント・グループを定義して、イベントを相互に関連付け、それらのイベントをイベント・サブスクリプションのグループとして参照することもできます。イベント・グループはイベントの一種で、個別のメンバー・イベントの集合で構成されます。イベント・グループの内部名には、個別のイベント名と同じ書式を使用する必要があります。イベント・グループを定義すると、グループに対してサブスクリプションを登録できます。グループ内のイベントに対して個別にサブスクリプションを作成する必要はありません。サブスクリプションは、グループのいずれかのメンバー・イベントが発生したときに実行されます。

---

**注意：** イベント・グループを使用して、イベントを呼び出すことはできません。各イベントは個別に呼び出す必要があります。

---

イベントの内容を説明するために必要な詳細情報は、イベント名およびイベント・キーとともに、イベント・データと呼ばれます。たとえば、発注イベントのイベント・データには、項目番号、説明およびコストなどが含まれます。イベント・データは、XML 文書として作成できます。

イベントが発生するアプリケーションには、イベントをイベント・マネージャに呼び出す際に、イベント・データを組み込むことができます。アプリケーションからイベント・データが渡されない場合は、イベントにジェネレート関数を指定して、イベント名、イベント・キーおよびパラメータ・リスト（オプション）から完全なイベント・データを生成する必要があります。ジェネレート関数は、標準 API に従ってください。8-268 ページの「[Raise](#)」および 7-19 ページの「[イベント・データ・ジェネレート関数の標準 API](#)」を参照してください。

イベント・マネージャはサブスクリプションを実行する前に、各サブスクリプションにイベント・データが必要かどうかをチェックします。必要なイベント・データが渡されていない場合、イベント・マネージャはそのイベントに対してジェネレート関数をコールし、イベント・データを生成します。イベント・データを必要とするイベントにジェネレート関数が定義されていない場合、イベント名およびイベント・キーを使用してデフォルトのイベント・データが作成されます。

---

**注意：** ジェネレート関数の負荷が大きいため、イベントの呼び出し後に制御をできるだけ早くコール側アプリケーションに戻したい場合は、完全なイベント・データを必要とするすべてのサブスクリプションを遅延させることができます。後でそれらのサブスクリプションを実行するまで、イベント・マネージャはジェネレート関数を実行しません。13-43 ページの「[遅延サブスクリプション処理](#)」を参照してください。

---

イベント定義を自動的に作成するプログラムを使用する場合は、プログラム独自の名前と簡単な識別子をイベントの所有者名および所有者タグとして設定できます。この識別情報を使用して、所有するイベントをプログラムから検索することができます。所有者名および所有者タグは、「イベントの編集」および「グループの編集」画面から必要に応じて手動で更新できます。

▶ イベントの定義

1. Web ブラウザを使用して、次の URL に接続します。

```
<webagent>/wf_event_html.listevents
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

**注意：** 「イベント」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。



2. 「イベント」画面に、既存のイベントのリストが表示されます。「イベント」画面には、各イベントの内部名、表示名、タイプおよび状態の一覧が表示されます。

「イベントの追加」ボタンを選択して「イベントの編集」画面を開きます。

The screenshot shows a Netscape Communicator window titled 'イベントの編集 - Netscape'. The menu bar includes 'ファイル(F)', '編集(E)', '表示(V)', 'ジャンプ(G)', 'Communicator(C)', and 'ヘルプ(H)'. The address bar shows a URL starting with 'http://'. The main content area is titled 'イベントの編集' and features the Oracle logo in the top right. The form contains the following fields and controls:

- 名前 (Name): A text input field.
- 表示名 (Display Name): A text input field.
- 説明 (Description): A text area with scrollbars.
- 状態 (Status): A dropdown menu currently showing '使用可能' (Available).
- 機能の生成 (Function Generation): A text input field.
- 所有者名 (Owner Name): A text input field.
- 所有者タグ (Owner Tag): A text input field.
- 送信 (Send) and 取消 (Cancel) buttons at the bottom center.

The status bar at the bottom indicates '問合せ詳細の入力' (Input of inquiry details).

3. イベントの内部名を「名前」フィールドに入力します。すべての Oracle Workflow API、SQL スクリプトおよび PL/SQL プロシージャでは、イベントの識別時に内部名が参照されます。内部名は大文字 / 小文字が区別されます。次のように、複数の識別子をピリオドで区切った書式で記述することをお勧めします。

`<company>.<family>.<product>.<component>.<object>.<event>`

4. イベントの表示名を入力します。この名前は、「イベント」リストに表示されます。
5. オプションで、イベントの説明を入力します。
6. 「状態」フィールドで、イベントのステータスとして「使用可能」または「使用不能」を選択します。イベントを使用不能にしても、そのイベントは「イベント」リストに表示され、参照できます。ただし、有効なサブスクリプションでは使用できません。

7. ローカル・システムで発生するイベントを定義する場合は、イベントのジェネレート関数を入力します。ジェネレート関数は、イベント名、イベント・キーおよびパラメータ・リスト（オプション）から完全なイベント・データを生成する PL/SQL プロシージャです。7-19 ページの「[イベント・データ・ジェネレート関数の標準 API](#)」を参照してください。
8. オプションで、イベントを所有するプログラムまたはアプリケーションを識別することもできます。プログラム名を「所有者名」フィールドに、プログラム ID を「所有者タグ」フィールドに入力してください。「イベントの編集」画面でイベントを手動で定義する場合は、「所有者名」および「所有者タグ」を入力する必要はありません。ただし、イベント定義を自動的に作成するプログラムを使用する場合、これらのフィールドにはそのプログラムで設定された所有者情報が表示されます。この情報は、「イベントの編集」画面から必要に応じて手動で更新できます。
9. 「送信」ボタンを選択してイベントを保存し、「イベント」画面に戻ります。「イベント」画面には、更新されたイベントのリストが表示されます。  
  
「取消」ボタンを選択し、イベントを保存しないで「イベント」画面に戻ることもできます。

### ▶ イベント・グループの定義

1. Web ブラウザを使用して、次の URL に接続します。

```
<webagent>/wf_event_html.listevents
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・ブリファレンスの設定](#)」を参照してください。

---

---

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

---

---

---

---

**注意：** 「イベント」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---

---



2. 「イベント」画面に、既存のイベントのリストが表示されます。「グループの追加」ボタンを選択して「イベントの編集」画面を開きます。

The screenshot shows a Netscape browser window titled 'グループの編集 - Netscape'. The address bar is empty. The menu bar includes 'ファイル(F)', '編集(E)', '表示(V)', 'ジャンプ(G)', 'Communicator(C)', and 'ヘルプ(H)'. The toolbar contains icons for home, back, forward, and help. The main content area is titled 'グループの編集' and features the Oracle logo in the top right. The form fields are as follows:

- 名前: A text input field.
- 表示名: A text input field.
- 説明: A text area with a vertical scrollbar.
- 状態: A dropdown menu with '使用可能' selected.
- 所有者名: A text input field.
- 所有者タグ: A text input field.

At the bottom of the form are two buttons: '送信' (Send) and '取消' (Cancel). The status bar at the bottom of the browser window shows 'ドキュメント: 完了。' (Document: Complete).

3. イベント・グループの内部名を「名前」フィールドに入力します。すべての Oracle Workflow API、SQL スクリプトおよび PL/SQL プロシージャでは、イベント・グループの識別時に内部名が参照されます。内部名は大文字 / 小文字が区別されます。内部名を記述するときは、次のように複数の識別子をピリオドで区切った書式を使用してください。  
  
`<company>.<family>.<product>.<component>.<object>.<event>`
4. イベント・グループの表示名を入力します。この名前は、「イベント」リストに表示されます。
5. オプションで、イベント・グループの説明を入力します。
6. 「状態」フィールドで、イベント・グループのステータスとして「使用可能」または「使用不能」を選択します。イベント・グループを使用不能にしても、そのイベント・グループは「イベント」リストに表示され、参照できます。ただし、有効なサブスクリプションでは使用できません。
7. オプションで、イベント・グループを所有するプログラムまたはアプリケーションを識別することもできます。プログラム名を「所有者名」フィールドに、プログラム ID を



「所有者タグ」フィールドに入力してください。「グループの編集」画面でイベント・グループを手動で定義する場合は、「所有者名」および「所有者タグ」を入力する必要はありません。ただし、イベント・グループ定義を自動的に作成するプログラムを使用する場合、イベント・マネージャのこれらのフィールドにはそのプログラムで設定された所有者情報が表示されます。この情報は、「グループの編集」画面から必要に応じて手動で更新できます。

8. 「送信」ボタンを選択してイベント・グループを保存します。

---

**注意：**「取消」ボタンを選択し、イベント・グループを保存しないで「イベント」画面に戻ることもできます。

---

イベント・グループ定義を保存すると、「グループの編集」画面にそのグループのメンバー・イベントのリストが表示され、各イベントの名前、表示名およびステータスがされます。

グループの編集 - Netscape

ファイル(F) 編集(E) 表示(V) ジャンプ(Q) Communicator(C) ヘルプ(H)

グループの編集

名前

表示名

説明

状態

所有者名

所有者タグ

グループ内のイベント

名前	表示名	状態	サブスクリプション	編集
----	-----	----	-----------	----

イベントの追加

送信 取消

問合せ詳細の入力

9. メンバー・イベントをグループに追加するには、「イベントの追加」ボタンを選択します。

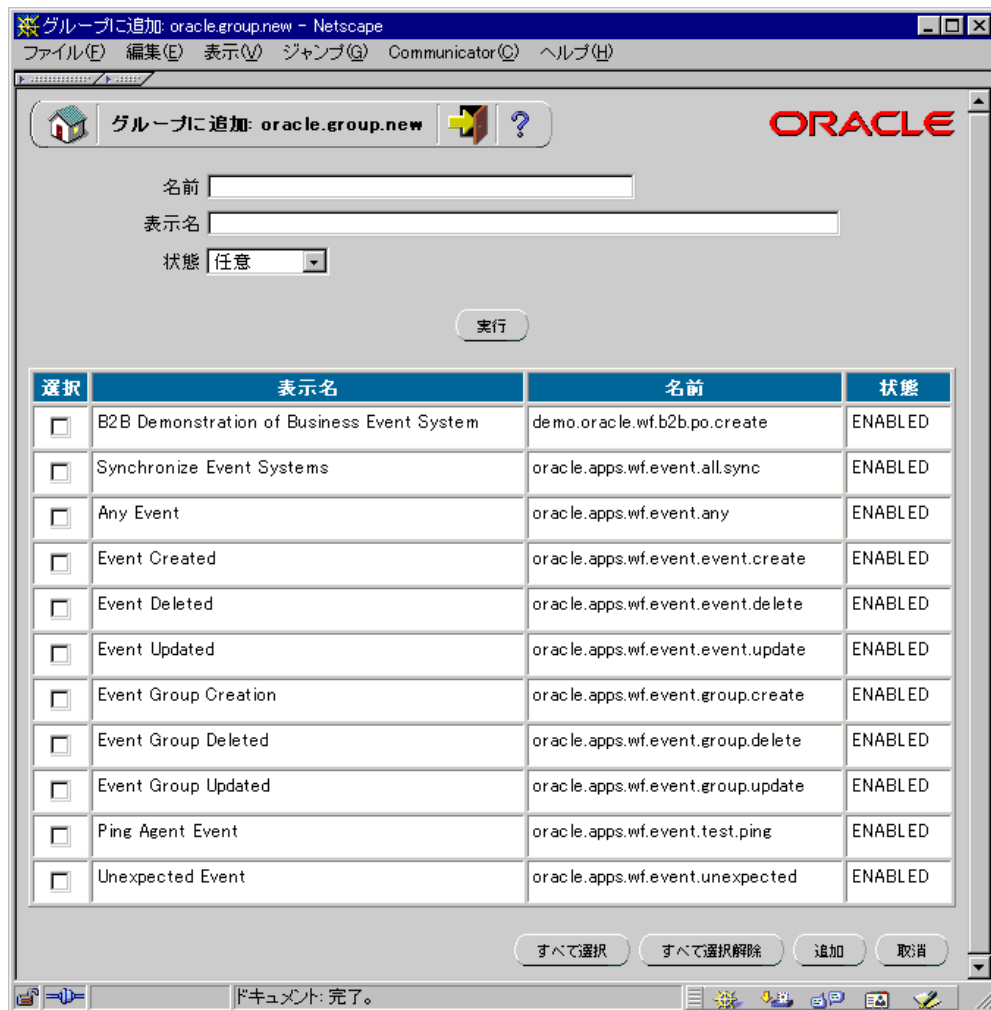
---

**注意：** イベント・グループには、個別のイベントのみを登録できます。  
イベント・グループには、別のグループを登録できません。

---



10. 表示された「グループに追加」画面で、検索基準を入力し、追加するイベントを検索します。検索基準として、次の項目を使用できます。
- 名前： 追加するイベントの内部名を入力します。
  - 表示名： 追加するイベントの表示名を入力します。
  - 状態： 追加するイベントのステータスとして「使用可能」または「使用不能」を選択します。すべてのステータスからイベントを検索するには、「任意」を選択します。
11. 「実行」を選択します。「グループに追加」画面に、検索基準と一致したイベントのリストが表示されます。



12. イベント・グループに追加するイベントを選択します。リスト内のイベントをすべて選択するときは、「すべて選択」ボタンを選択します。リスト内のイベントをすべて選択解除するときは、「すべて選択解除」ボタンを選択します。

別のイベントを検索する場合は、新しい検索基準を入力して「実行」ボタンを選択します。「グループに追加」画面に、新しい検索基準と一致したイベントのリストが表示されます。

「取消」ボタンを選択し、現行の検索を取り消して前の検索結果に戻ることもできます。

13. 追加するイベントの選択が終了したら、「追加」ボタンを選択して、選択したイベントをイベント・グループに追加します。「グループの編集」画面に、更新されたイベント・グループ・メンバーのリストが表示されます。



14. 「送信」ボタンを選択してイベント・グループ定義を保存します。

**注意：**「取消」ボタンを選択して、最新の検索結果が表示された「グループに追加」画面に戻ることもできます。

15. グループからメンバー・イベントを削除する場合は、「グループの編集」画面で、削除するイベントを選択します。リスト内のイベントをすべて選択するときは、「すべて選択」を選択します。リスト内のイベントをすべて選択解除するときは、「すべて選択解除」を選択します。

16. 「削除」ボタンを選択して、選択したイベントをイベント・グループから削除します。「グループの編集」画面に、更新されたイベント・グループ・メンバーのリストが表示されます。

---

**注意：** メンバー・イベントをイベント・グループから個別に削除しても、各イベントのイベント定義は削除されません。各イベントは「イベント」リストに残ります。

---

17. イベントを参照するサブスクリプションを表示するには、そのイベントの「サブスクリプション」列にあるスケジュール・アイコンを選択します。「イベント・サブスクリプション」画面に、イベントのサブスクリプションのリストが表示されます。

---

**注意：** イベントにサブスクリプションを指定していない場合は、空のスケジュール・アイコンが表示されます。サブスクリプションから参照されているイベントの場合は、スケジュール・アイコンが正式に表示されます。

---

「イベント・サブスクリプション」画面の「サブスクリプションの追加」ボタンを選択して、イベントに対して新しいサブスクリプションの定義を開始できます。「サブスクリプションの編集」画面が表示され、イベント名が「イベント・フィルタ」フィールドに自動的に入力されます。13-48 ページの「[イベント・サブスクリプションの定義](#)」を参照してください。

18. イベントを更新するには、そのイベントの「編集」列にある鉛筆アイコンを選択します。「イベントの編集」画面が表示されます。イベント定義に変更を加え、変更内容を保存します。13-6 ページの「[イベントの定義](#)」を参照してください。

## ► イベントの検索

1. Web ブラウザを使用して、次の URL に接続します。

```
<webagent>/wf_event_html.findevent
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

---

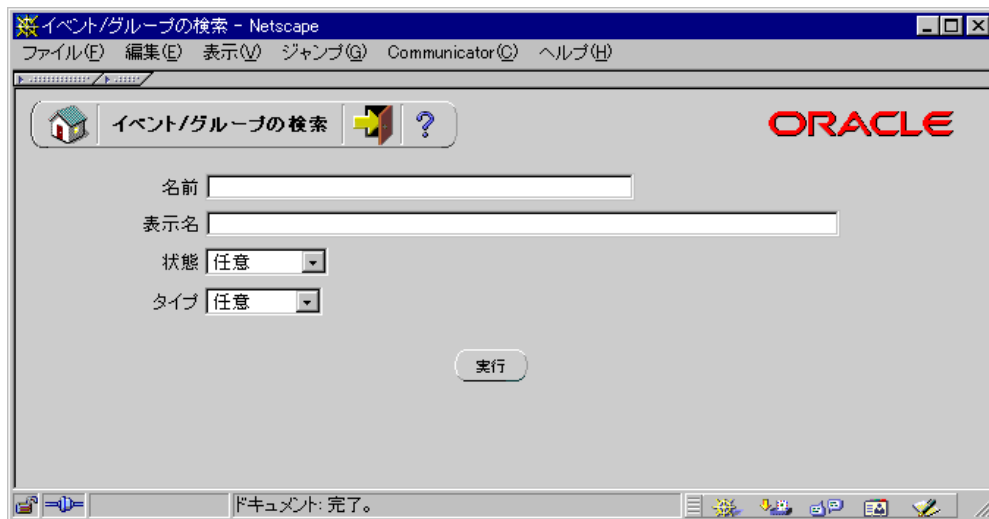
**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

---

---

**注意：**「イベント / グループの検索」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---



2. 「イベント / グループの検索」画面が表示されます。「イベント / グループの検索」画面では、検索基準を入力して特定のイベントを検索できます。検索基準として、次の項目を使用できます。
  - 名前： 表示するイベントの内部名を入力します。
  - 表示名： 表示するイベントの表示名を入力します。
  - 状態： 表示するイベントのステータスとして「使用可能」または「使用不能」を選択するか、「任意」を選択してすべてのステータスのイベントを表示します。
  - タイプ： 表示するイベントのタイプとして「イベント」または「グループ」を選択するか、「任意」を選択してすべてのタイプのイベントを表示します。
3. 「実行」を選択します。「イベント」画面に、検索基準と一致したイベントのリストが表示されます。

▶ イベントの更新または削除

1. 「イベント」画面で、更新または削除するイベントを選択します。「イベント / グループの検索」画面を使用して、必要なイベントを検索し、「イベント」画面に表示することができます。13-15 ページの「[イベントの検索](#)」を参照してください。



2. イベントを参照するサブスクリプションを表示するには、そのイベントの「サブスクリプション」列にあるスケジュール・アイコンを選択します。「イベント・サブスクリプション」画面に、イベントのサブスクリプションのリストが表示されます。

**注意：** イベントにサブスクリプションを指定していない場合は、空のスケジュール・アイコンが表示されます。サブスクリプションから参照されているイベントの場合は、スケジュール・アイコンが正式に表示されます。

「イベント・サブスクリプション」画面の「サブスクリプションの追加」ボタンを選択して、イベントに対して新しいサブスクリプションの定義を開始できます。「サブスクリプションの編集」画面が表示され、イベント名が「イベント・フィルタ」フィールドに自動的に入力されます。13-48 ページの「[イベント・サブスクリプションの定義](#)」を参照してください。

- 3. イベントを更新するには、そのイベントの「編集」列にある鉛筆アイコンを選択します。「イベントの編集」画面が表示されます。イベント定義に変更を加え、変更内容を保存します。13-6 ページの「[イベントの定義](#)」を参照してください。
- 4. イベントを削除するには、そのイベントの「削除」列にあるごみ箱アイコンを選択し、表示された確認ウィンドウで「OK」を選択します。確認ウィンドウで「取消」ボタンを選択し、イベントを削除しないで「イベント」画面に戻ることもできます。

**注意：** 削除できるイベントは、サブスクリプションから参照されていないイベントおよびイベント・グループに属していないイベントのみです。

システム

システムとは、ホスト・マシンやデータベース・インスタンスなど、論理的に独立したソフトウェア環境のことです。イベント・マネージャ内でイベントを交換するシステムを定義する必要があります。

システムを定義する際に、イベント・マネージャのオブジェクト定義の更新の送信元となるマスター・システムに関連付けるかどうかを指定できます。

各システムでは、アドレス指定可能な通信ポイント（エージェント）を1つ以上公開できます。システムを定義したら、ビジネス・イベントの交換に使用するエージェントをシステム内に定義する必要があります。13-24 ページの「[エージェント](#)」を参照してください。

ローカル・システム

Oracle Workflow をデータベース内にインストールすると、そのデータベースはイベント・マネージャのシステムとして自動的に定義され、「グローバル・ワークフロー・プリファレンス」画面のローカル・システムとして設定されます。次の表は、ローカル・システム定義のデフォルト・プロパティの一覧です。

表 13-1

システム・プロパティ	値
名前	< データベース・グローバル名 >
表示名	< データベース・グローバル名 >
説明	Oracle Workflow Configuration Assistant によって作成されたローカル・システム
マスター	(空白)

ローカル・システム定義は、必要に応じて更新できます。

ローカル・システムのステータスは、デフォルトでは「使用可能」に設定されます。ビジネス・イベント・システムの設定が終了したら、「グローバル・ワークフロー・プリファレン



ス」画面を使用して、イベントの処理に必要なシステム・ステータスを設定できます。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

## ► システムの定義

1. Web ブラウザを使用して、次の URL に接続します。

```
<webagent>/wf_event_html.listsystems
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

---

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

---



---

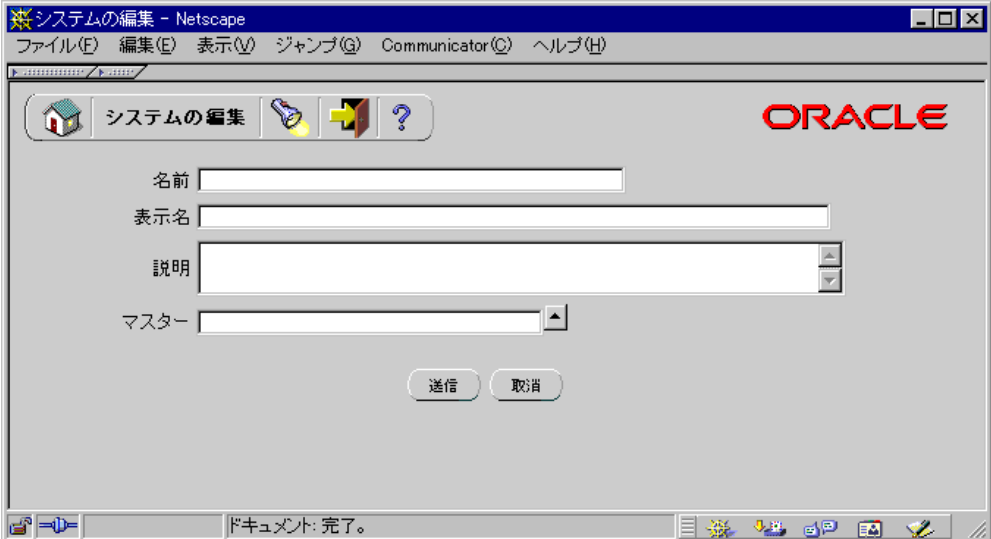
**注意：** 「システム」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---



2. 「システム」画面に、既存のシステムのリストが表示されます。「システム」画面には、各システムの内部名、表示名およびマスター・システムの一覧が表示されます。アスタリスク (\*) はローカル・システムを示します。

「システムの追加」ボタンを選択して「システムの編集」画面を開きます。



The screenshot shows a Netscape browser window titled 'システムの編集 - Netscape'. The address bar shows a URL starting with 'http://'. The page has a header with the Oracle logo and a navigation bar with icons for home, edit, add, and help. The main form contains four input fields: '名前' (Name), '表示名' (Display Name), '説明' (Description), and 'マスター' (Master). Below the fields are two buttons: '送信' (Send) and '取消' (Cancel). The status bar at the bottom indicates 'ドキュメント: 完了。' (Document: Complete).

3. システムの内部名を「名前」フィールドに入力します。すべての Oracle Workflow API、SQL スクリプトおよび PL/SQL プロシージャでは、システムの識別時に内部名が参照されます。

---

**注意：** 内部名には大文字のみを使用する必要があります。一重引用符 (')、二重引用符 (") またはスペースを含めないでください。

---

4. システムの表示名を入力します。この名前は、「システム」リストに表示されます。
5. オプションで、システムの説明を入力します。
6. オプションで、イベント・マネージャのオブジェクト定義の更新の送信元となる、マスター・システムを入力します。「マスター」フィールドの上矢印のアイコンをクリックすると、選択可能なシステムのリストが表示されます。13-23 ページの「[値リストの使用](#)」を参照してください。
7. 「送信」ボタンを選択してシステムを保存し、「システム」画面に戻ります。「システム」画面に、更新されたシステムのリストが表示されます。

「取消」ボタンを選択し、システムを保存しないで「システム」画面に戻ることができます。

## ► システムの検索

1. Web ブラウザを使用して、次の URL に接続します。

```
<webagent>/wf_event_html.findsystem
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・ブリファレンスの設定](#)」を参照してください。

---

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

---

---

**注意：** 「システムの検索」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---



2. 「システムの検索」画面が表示されます。「システムの検索」画面では、検索基準を入力して特定のシステムを検索できます。検索基準として、次の項目を使用できます。
  - 名前： 表示するシステムの内部名を入力します。
  - 表示名： 表示するシステムの表示名を入力します。
  - マスター： 表示するシステムのマスター・システムを入力します。このフィールドの上矢印のアイコンをクリックすると、選択可能なシステムのリストが表示されます。13-23 ページの「[値リストの使用](#)」を参照してください。
3. 「検索」ボタンを選択します。「システム」画面に、検索基準と一致したシステムのリストが表示されます。アスタリスク (\*) はローカル・システムを示します。

### ► システムの更新または削除

1. 「システム」画面で、更新または削除するシステムを選択します。「システムの検索」画面を使用して、必要なシステムを検索し、「システム」画面に表示することができます。13-21 ページの「[システムの検索](#)」を参照してください。



2. 特定のシステムのサブスクリプションを表示するには、そのシステムの「サブスクリプション」列にあるスケジュール・アイコンを選択します。「イベント・サブスクリプション」画面に、サブスクリプションのリストがシステム単位に表示されます。

---

**注意：** システムにサブスクリプションを指定していない場合は、空のスケジュール・アイコンが表示されます。サブスクリプションが指定されているイベントの場合は、スケジュール・アイコンが正式に表示されます。

---

「イベント・サブスクリプション」画面の「サブスクリプションの追加」ボタンを選択して、システムに対して新しいサブスクリプションの定義を開始できます。「サブスクリプションの編集」画面が表示され、システム名が「システム」フィールドに自動的に入力されます。13-48 ページの「[イベント・サブスクリプションの定義](#)」を参照してください。

3. システムを更新するには、そのシステムの「編集」列にある鉛筆アイコンを選択します。「システムの編集」画面が表示されます。システム定義に変更を加え、変更内容を保存します。13-19 ページの「[システムの定義](#)」を参照してください。
4. システムを削除するには、そのシステムの「削除」列にあるごみ箱アイコンを選択して、表示される確認ウィンドウで「OK」を選択します。確認ウィンドウで「取消」ボタンを選択し、システムを削除しないで「システム」画面に戻ることもできます。

---

**注意：** 削除できるシステムは、エージェントが定義されていないシステムまたはサブスクリプションから参照されていないシステムのみです。

---

## ► 値リストの使用

1. 値リストをサポートしているフィールドには、フィールドの上矢印のアイコンをクリックすると、値リスト・ウィンドウが表示されます。
2. 「検索」フィールドに検索基準を入力し、「検索」ボタンを選択して、基準と一致する値を検索します。「消去」ボタンを選択して、「検索」フィールドの内容を消去することもできます。検索基準を指定せずに「検索」のみを選択すると、完全な値リストが表示されます。
3. リストの値をクリックして選択し、値リスト・ウィンドウをクローズします。選択した値が元のフィールドに挿入されます。

## エージェント

エージェントは、システム内の名前付き通信ポイントです。システム内およびシステム間の通信は、エージェント間のメッセージの送信によって実行されます。1つのシステムに対して複数のエージェントを設定し、それぞれ異なる通信代替を指定できます。たとえば、1つのシステムに対して複数のエージェントを定義し、インバウンド通信とアウトバウンド通信、異なるプロトコルによる通信、異なる伝播頻度など、様々な通信代替に対応しています。

イベント・マネージャ内でイベントの交換に使用するエージェントを定義する必要があります。各エージェントの名前は、システム内で一意でなければなりません。エージェントは、次のような書式の複合名を使用して、Oracle Workflow 内のコードから参照できます。

```
<agent_name>@<system_name>
```

たとえば、システム HUB 内のエージェント WF\_IN は、WF\_IN@HUB として参照できます。

ローカル・システムにエージェントを定義したら、ローカル・インバウンド・エージェントのリスナーおよびローカル・アウトバウンド・エージェントの伝播をスケジュールして、エージェントをイベント・メッセージの伝播用に設定する必要があります。13-62 ページの「ローカル・インバウンド・エージェントのリスナーのスケジュール」および 13-67 ページの「ローカル・アウトバウンド・エージェントの伝播のスケジュール」を参照してください。

### エージェントへの方向の割当て

イベント・マネージャでエージェントを定義する際に、エージェントがローカル・システム内でサポートする通信方向を指定する必要があります。

- 受信： システムへのインバウンド通信。エージェントはメッセージを特定のプロトコルで受信して、標準の書式でシステムに渡します。
- 送信： システムからのアウトバウンド通信。エージェントはシステムから標準の書式でメッセージを受け取り、指定されたプロトコルを使用して送信します。

### エージェントへのプロトコルの割当て

各エージェントは、メッセージの通信に使用するプロトコルに関連付ける必要があります。プロトコルには、メッセージのエンコードおよび送信方法が指定されています。メッセージを正しく通信するには、エージェントの送受信で同じプロトコルを使用する必要があります。

プロトコルは、SQLNET などのネットワーク標準を表します。また、ビジネス間標準も表します。つまり、システム間の上位レベルのメッセージ書式およびハンドシェイク手順を定義します。

ビジネス・イベント・システムは、AQ キューを介してエージェントと対話します。AQ を使用すれば、AQ でサポートされている SQLNET プロトコルを介してメッセージを伝播できます。Oracle9i の AQ には、インターネット・アクセス機能も組み込まれています。AQ の

Internet Data Access Presentation (IDAP) を使用してメッセージを作成し、それらの IDAP メッセージを HTTP や HTTPS などのトランスポート・プロトコルを使用してインターネットを介して送信すれば、AQ 操作をインターネットを介して行うことができます。また、Oracle9i では、AQ のメッセージ・ゲートウェイ機能を使用して、Oracle 以外のメッセージ・システムを使用するアプリケーションと AQ との間で通信することができ、サードパーティのメッセージ・ソリューションとの統合が可能になります。さらに、他のサービスでサポートされているプロトコルを使用して、メッセージを伝播することもできます。次の表は、様々なプロトコルに使用できるサービスを、データベースのバージョン別に示しています。

表 13-2

データベースのバージョン	SQLNET プロトコル	HTTP/HTTPS プロトコル	サードパーティのメッセージ・ソリューションとの統合
Oracle8i	Oracle Advanced Queuing	Oracle Message Broker	Oracle Message Broker
Oracle9i	Oracle Advanced Queuing	Oracle Advanced Queuing	Oracle Advanced Queuing メッセージ・ゲートウェイ機能

カスタム・プロトコルを実装するには、次の手順を実行する必要があります。

1. 保留中のインバウンドおよびアウトバウンド・メッセージを保持するように、AQ キューを定義します。
2. AQ キューとの間でメッセージを伝播するコードを指定します。
3. 「イベント・プロトコル・タイプ (WF\_AQ\_PROTOCOLS)」選択肢タイプの新しいプロトコルに対して、選択肢コードを定義します。この選択肢コードは、「標準」項目タイプに格納されます。イベント・マネージャでは、エージェントのプロトコルを検証するときに、「イベント・プロトコル・タイプ」選択肢タイプが使用されます。4-20 ページの「[選択肢タイプの選択肢コードの作成](#)」を参照してください。
4. 新しいプロトコルに基づいてエージェントを定義します。13-30 ページの「[エージェントの定義](#)」を参照してください。

エージェントがインバウンド通信を使用する場合は、システムがエージェントとの通信に使用するアドレスを指定する必要があります。アドレスの書式は、エージェントのプロトコルによって異なります。SQLNET プロトコルを使用するエージェントの場合、AQ 伝播を有効にするには、次の書式でアドレスを指定する必要があります。

```
<schema>.<queue>@<database link>
```

この書式の <schema> はキューを所有するスキーマ、<queue> はキューの名前、<database link> はキューが配置されているインスタンスへのデータベース・リンクの名前を表します。

---

---

**注意：** データベース・リンク名には、データベース・リンクの作成時に指定された名前を正確に入力する必要があります。たとえば、データベース・リンク名が ORA816.US.ORACLE.COM の場合は、データベース上のエージェントのアドレスに対してその名前を正確に入力する必要があります。ORA816 のように名前を省略することはできません。

ビジネス・イベント・システムで使用するデータベース・リンクの名前は、ドメイン名を含む完全修飾名でなければなりません。データベース・リンクの名前を確認するには、次の構文を使用します。

```
SELECT db_link FROM all_db_links
```

2-90 ページの「[データベース・リンクの作成](#)」を参照してください。

---

---

## エージェントへのキューの割当て

各エージェントは、AQ キューに関連付ける必要があります。ローカル・システムは、このキューを使用してエージェントと対話します。メッセージを送信する場合、システムはキューにあるメッセージをエンキューして、宛先アドレスを設定します。メッセージを受信する場合、システムはキューにあるキュー・リスナーを実行します。

Oracle Workflow Business Event System 内のイベント・メッセージは、データ型 WF\_EVENT\_T で定義された標準の書式でエンコードされます。各エージェントは、キュー・ハンドラと呼ばれる PL/SQL パッケージに割り当て、Workflow 標準書式とエージェントのキューに必要な書式との間で変換を行う必要があります。8-248 ページの「[イベント・メッセージ構造](#)」を参照してください。

---

---

**注意：** エージェントのキューのペイロード・タイプが WF\_EVENT\_T の場合でも、システム固有の AQ メッセージ・プロパティを設定するために、キュー・ハンドラに割り当てる必要があります。

---

---

Oracle Workflow には、WF\_EVENT\_QH および WF\_ERROR\_QH という 2 つの標準キュー・ハンドラが用意されています。これらのキュー・ハンドラは、SQLNET 伝播を使用し、WF\_EVENT\_T データ型をペイロードとして使用するキューに対応しています。WF\_EVENT\_QH は、通常のビジネス・イベント・システム処理を扱うキューで使います。WF\_ERROR\_QH は、エラー・キューでのみ使用します。

Oracle Workflow には、WF\_EVENT\_OMB\_QH というキュー・ハンドラも用意されています。Oracle8i に Oracle Message Broker を実装して、HTTP などの別のプロトコルを使用してシステム間でメッセージを伝播するときは、このキュー・ハンドラを使用します。2-94 ページの「[手順 WF-15 WF\\_EVENT\\_OMB\\_QH キュー・ハンドラの設定](#)」および 8-264 ページの「[WF\\_EVENT\\_T および OMBAQ\\_TEXT\\_MSG 間のマッピング](#)」を参照してください。

他の書式を必要とするキューの場合は、その書式用のカスタム・キュー・ハンドラを作成します。カスタム・キュー・ハンドラには、カスタム書式のメッセージをエンキューおよびデ



キューするための標準 API を組み込む必要があります。7-21 ページの「[キュー・ハンドラの標準 API](#)」を参照してください。

標準エージェント

Oracle Workflow をインストールすると、ビジネス・イベント・システムに対して 4 つの標準エージェントが自動的に定義されます。

- WF\_IN: 標準インバウンド・エージェント
- WF\_OUT: 標準アウトバウンド・エージェント
- WF\_DEFERRED: 遅延サブスクリプション処理用標準エージェント
- WF\_ERROR: エラー処理用標準エージェント

これらのエージェントでは、Oracle Workflow のインストール時に自動的に定義される標準のキューが使用されます。2-91 ページの「[キューの設定](#)」を参照してください。

WF\_IN および WF\_OUT エージェントは、使用可能 / 使用不可を切り替えることができます。ただし、エージェントの定義は変更しないでください。WF\_DEFERRED エージェントおよび WF\_ERROR エージェントの定義も変更しないでください。

ただし、ビジネス・イベント・システムの遅延サブスクリプション処理およびエラー処理を有効にするには、WF\_DEFERRED エージェントおよび WF\_ERROR エージェントのリスナーをそれぞれスケジュールする必要があります。また、WF\_IN および WF\_OUT エージェントをイベント・メッセージの伝播に使用するときは、WF\_IN のリスナーおよび WF\_OUT の伝播もスケジュールしてください。13-62 ページの「[ローカル・インバウンド・エージェントのリスナーのスケジュール](#)」および 13-67 ページの「[ローカル・アウトバウンド・エージェントの伝播のスケジュール](#)」を参照してください。

通知メーラー SMTP キューには、WF SMTP\_O\_1\_QUEUE という標準エージェントが定義されます。このエージェントは、「セットアップのチェック」画面と「イベント・システム・ローカル・キュー」画面に表示されます。これらの画面を使用して、通知メーラーのキューにある通知メッセージの数をチェックできます。ただし、WF SMTP\_O\_1\_QUEUE エージェントは、ビジネス・イベント・システムでは使用されません。このため、ステータスは「使用不能」になり、このエージェントに対してキュー・ハンドラは定義されません。このエージェントに対してエージェント・リスナーを実行しないでください。2-45 ページの「[手順 WF-9 通知メーラーの導入](#)」、13-57 ページの「[ビジネス・イベント・システムのセットアップのチェック](#)」および 13-80 ページの「[ローカル・キューの確認](#)」を参照してください。

次の表は、標準 WF\_IN エージェントのデフォルトのプロパティを示しています。

表 13-3

エージェント・プロパティ 値	
名前	WF_IN
表示名	WF_IN

表 13-3 (続き)

エージェント・プロパティ	値
説明	WF_IN
プロトコル	SQLNET
アドレス	<workflow schema>.WF_IN@<local database>
システム	<local system>
キュー・ハンドラ	WF_EVENT_QH
キュー名	<workflow schema>.WF_IN
方向	受信
ステータス	使用可能

次の表は、標準 WF\_OUT エージェントのデフォルトのプロパティを示しています。

表 13-4

エージェント・プロパティ	値
名前	WF_OUT
表示名	WF_OUT
説明	WF_OUT
プロトコル	SQLNET
アドレス	<workflow schema>.WF_OUT@<local database>
システム	<local system>
キュー・ハンドラ	WF_EVENT_QH
キュー名	<workflow schema>.WF_OUT
方向	送信
ステータス	使用可能

次の表は、標準 WF\_DEFERRED エージェントのデフォルトのプロパティを示しています。

表 13-5

エージェント・プロパティ	値
名前	WF_DEFERRED

表 13-5（続き）

エージェント・プロパティ	値
表示名	WF_DEFERRED
説明	WF_DEFERRED
プロトコル	SQLNET
アドレス	<workflow schema>.WF_DEFERRED@<local database>
システム	<local system>
キュー・ハンドラ	WF_EVENT_QH
キュー名	<workflow schema>.WF_DEFERRED
方向	受信
ステータス	使用可能

次の表は、標準 WF\_ERROR エージェントのデフォルトのプロパティを示しています。

表 13-6

エージェント・プロパティ	値
名前	WF_ERROR
表示名	WF_ERROR
説明	WF_ERROR
プロトコル	SQLNET
アドレス	<workflow schema>.WF_ERROR@<local database>
システム	<local system>
キュー・ハンドラ	WF_ERROR_QH
キュー名	<workflow schema>.WF_ERROR
方向	受信
ステータス	使用可能

次の表は、標準 WF\_SMTP\_O\_1\_QUEUE エージェントのデフォルトのプロパティを示しています。

表 13-7

エージェント・プロパティ	値
名前	WF_SMTP_O_1_QUEUE
表示名	WF_SMTP_O_1_QUEUE
説明	メーラーのキューです。このエージェントに対してエージェント・リスナー・コンカレント・プログラムを送信しないでください。
プロトコル	SQLNET
アドレス	<workflow schema>.WF_SMTP_O_1_QUEUE@ <local database>
システム	<local system>
キュー・ハンドラ	
キュー名	<workflow schema>.WF_SMTP_O_1_QUEUE
方向	受信
ステータス	使用不能

➤ エージェントの定義

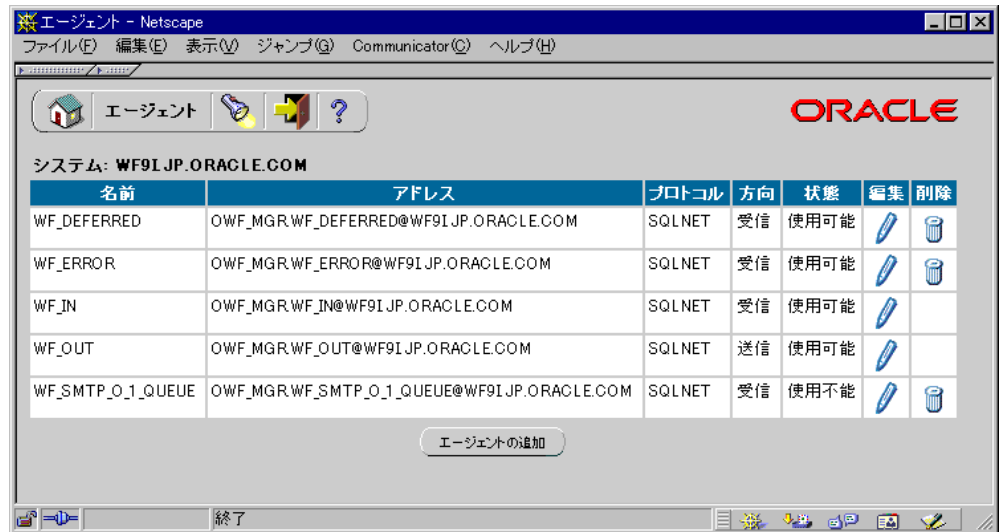
1. Web ブラウザを使用して、次の URL に接続します。

<webagent>/wf\_event\_html.listagents

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・ブリファレンスの設定](#)」を参照してください。

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

**注意：** 「エージェント」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。



2. 「エージェント」画面に既存のエージェントのリストが表示されます。配置されているシステム別にグループ化されています。「エージェント」画面には、各エージェントの内部名、アドレス、プロトコル、方向および状態の一覧が表示されます。

「エージェントの追加」ボタンを選択して「エージェントの編集」画面を開きます。

エージェントの編集 - Netscape

ファイル(F) 編集(E) 表示(V) ジャンプ(G) Communicator(C) ヘルプ(H)

エージェントの編集

名前

表示名

説明

プロトコル

アドレス

システム

キュー・ハンドラ

キュー名

方向

状態

送信 取消

ドキュメント:完了。

3. エージェントの内部名を「名前」フィールドに入力します。エージェントの内部名は、エージェントのシステム内で一意でなければなりません。すべての Oracle Workflow API、SQL スクリプトおよび PL/SQL プロシージャでは、エージェントの識別時に内部名が参照されます。

---

**注意：** 内部名には大文字のみを使用する必要があります。一重引用符 (')、二重引用符 (") またはスペースを含めないでください。

---

4. エージェントの表示名を入力します。
5. オプションで、エージェントの説明を入力します。
6. エージェントが使用するメッセージ通信プロトコルを選択します。
7. エージェントがシステムへのインバウンド通信を利用する場合は、エージェントのアドレスを入力します。アドレスの書式は、選択するプロトコルによって異なります。

SQLNET プロトコルを使用するエージェントの場合、AQ 伝播を有効にするには、次の書式でアドレスを指定する必要があります。

`<schema>.<queue>@<database link>`

<schema> はキューを所有するスキーマ、<queue> はキューの名前、<database link> はキューが配置されているインスタンスへのデータベース・リンクを表します。

---

**注意：** データベース・リンク名には、データベース・リンクの作成時に指定された名前を正確に入力する必要があります。2-90 ページの「[データベース・リンクの作成](#)」を参照してください。

---

8. エージェントが定義されているシステムを入力します。「システム」フィールドの上矢印のアイコンをクリックすると、選択可能なシステムのリストが表示されます。13-23 ページの「[値リストの使用](#)」を参照してください。
9. エージェントのキュー・ハンドラを入力します。キュー・ハンドラは PL/SQL パッケージの 1 つで、Workflow イベント・メッセージ書式 (WF\_EVENT\_T) とエージェントに関連付けられているキューに必要なメッセージ書式との間で変換を行います。7-21 ページの「[キュー・ハンドラの標準 API](#)」を参照してください。

---

**注意：** キュー・ハンドラ名は、すべて大文字で入力する必要があります。

---

10. ローカル・システムがエージェントとの対話に使用するキューの名前を入力します。このキュー名は、ローカル・システムのみが参照するため、このシステムの有効範囲に対応し、データベース・リンクを必要としない名前であればなりません。次の書式を使用してキュー名を指定します。

<schema>.<queue>

<schema> はキューを所有するスキーマ、<queue> はキュー名を表します。

---

**注意：** キュー名は、すべて大文字で入力する必要があります。

---

11. 「方向」フィールドで、システムへのインバウンド通信をサポートするエージェントの場合は「受信」を選択し、システムからのアウトバウンド通信をサポートするエージェントの場合は「送信」を選択します。
12. 「状態」フィールドで、エージェントのステータスとして「使用可能」または「使用不能」を選択します。エージェントを使用不能にしても、そのエージェントは「エージェント」リストに表示され、参照できます。ただし、有効なサブスクリプションでは使用できません。
13. 「送信」ボタンを選択してエージェントを保存し、「エージェント」画面に戻ります。「エージェント」画面に、更新されたエージェントのリストが表示されます。

「取消」ボタンを選択し、エージェントを保存しないで「エージェント」画面に戻ることができます。

### ▶ エージェントの検索

1. Web ブラウザを使用して、次の URL に接続します。

```
<webagent>/wf_event_html.findagent
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

---

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

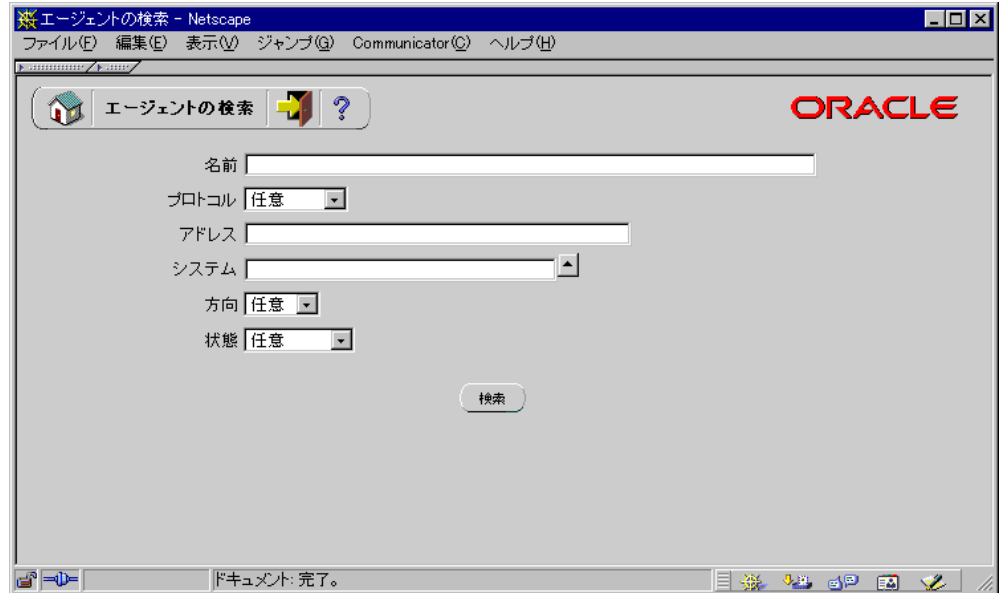
---

---

**注意：** 「エージェントの検索」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---





2. 「エージェントの検索」画面が表示されます。「エージェントの検索」画面では、検索基準を入力して特定のエージェントを検索できます。検索基準として、次の項目を使用できます。
  - 名前： 表示するエージェントの内部名を入力します。
  - プロトコル： 表示するエージェントのプロトコルを選択するか、「任意」を選択してすべてのプロトコルのエージェントを表示します。
  - アドレス： 表示するエージェントのアドレスを入力します。
  - システム： 表示するエージェントのシステムを入力します。このフィールドの上矢印のアイコンをクリックすると、選択可能なシステムのリストが表示されます。13-23 ページの「[値リストの使用](#)」を参照してください。
  - 方向： 表示するエージェントの方向として、「受信」または「送信」を選択します。すべての方向のエージェントを表示するときは、「任意」を選択します。
  - 状態： 表示するエージェントのステータスとして「使用可能」または「使用不能」を選択します。すべてのステータスのエージェントを表示するときは、「任意」を選択します。
3. 「検索」ボタンを選択します。「エージェント」画面に、検索基準と一致したエージェントのリストが表示されます。

➤ エージェントの更新または削除

1. 「エージェント」画面で、更新または削除するエージェントを選択します。「エージェントの検索」画面を使用して目的のエージェントを検索し、「エージェント」画面に表示することができます。13-34 ページの「[エージェントの検索](#)」を参照してください。



2. エージェントを更新するには、そのエージェントの「編集」列にある鉛筆アイコンを選択します。「エージェントの編集」画面が表示されます。エージェント定義に変更を加え、変更内容を保存します。13-30 ページの「[エージェントの定義](#)」を参照してください。
3. エージェントを削除するには、そのエージェントの「削除」列にあるごみ箱アイコンを選択して、表示される確認ウィンドウで「OK」を選択します。確認ウィンドウで「取消」ボタンを選択し、エージェントを削除しないで「エージェント」画面に戻ることもできます。

**注意：** 削除できるエージェントは、サブスクリプションから参照されていないエージェントのみです。

**注意：** エージェントに加えた変更がメッセージ伝播に必要な物理実装に影響する場合は、伝播設定を再度チェックする必要があります。13-56 ページの「[メッセージ伝播の設定](#)」を参照してください。

## イベント・サブスクリプション

イベント・サブスクリプションには、特定のイベントと特定のシステムの関連、およびトリガー・イベントの発生時に実行する処理を登録します。イベント・サブスクリプションは、イベント・マネージャで定義できます。

Oracle Workflow をインストールすると、事前定義済の Workflow イベントに対していくつかのデフォルトのサブスクリプションが自動的に作成されます。これらのサブスクリプションを更新したり、使用可能 / 使用不可を切り替えることにより、必要なイベント処理を実行できます。14-2 ページの「[事前定義済ワークフロー・イベント](#)」を参照してください。

イベントがローカルで呼び出されるか外部ソースから着信すると、イベント・マネージャは、そのイベントまたは Any イベントへの有効なサブスクリプションをローカル・システム単位に検索および実行します。発生したイベントへの有効なサブスクリプションが存在しない場合 (Any イベントへのサブスクリプションを除く)、Oracle Workflow は Unexpected イベントへの有効なサブスクリプションを実行します。14-13 ページの「[Any イベント](#)」および 14-15 ページの「[Unexpected イベント](#)」を参照してください。

### サブスクライバの定義

サブスクリプションの定義を開始するには、サブスクライバとなるシステムを指定します。サブスクライバは、サブスクリプションを実行するシステムです。

各サブスクリプションには、特定のシステムに対するアクションを定義します。つまり、サブスクリプションは、実行する処理に関係するシステム単位に、個別に定義する必要があります。たとえば、システム間でデータを伝播する場合は、送信システムに対して 1 つのサブスクリプションを定義し、受信システムに対して別のサブスクリプションを定義します。

### サブスクリプションのトリガーの定義

サブスクリプションを適用するイベントのソース・タイプを指定する必要があります。イベントには次のソース・タイプがあります。

- ローカル： サブスクリプションは、サブスクライバ・システムで発生したイベントにのみ適用されます。
- 外部： サブスクリプションは、サブスクライバ・システム上のインバウンド・エージェントが受信したイベントにのみ適用されます。

---

**注意：** サブスクライバ・システム上のインバウンド・エージェントが受信したイベント・メッセージは、送信元エージェントがリモート・システム上に存在するかローカル・システム上に存在するかにかかわらず、すべて外部ソースと見なされます。

---

- エラー： サブスクリプションは、WF\_ERROR キューからデキューされたエラー・イベントにのみ適用されます。

次に、サブスクリプションをトリガーするイベントを選択します。個々のイベントまたはイベント・グループを選択できます。イベント・グループを選択した場合、サブスクリプションは、グループのいずれかのメンバー・イベントが発生したときにトリガーされます。

また、特定のソース・エージェントから受信したイベントのみがトリガーするように、サブスクリプションを制限することもできます。ただし、ほとんどの場合、特定のソース・エージェントを指定する必要はありません。

## サブスクリプションの実行の制御

複数のサブスクリプションを同じイベントに対して定義する場合は、各サブスクリプションにフェーズ番号を指定することで、イベント・マネージャがサブスクリプションを実行する順序を制御できます。サブスクリプションは、フェーズの昇順で実行されます。たとえば、イベントの発生時に最初に実行するサブスクリプションに対して 10、次に実行するサブスクリプションに対して 20 を指定します。検証を行うサブスクリプションを実行してから他のタイプの処理を行うサブスクリプションを実行するなど、フェーズを使用すれば、異なるタイプのアクションを適切な順序で実行することができます。

---

---

**注意：** 同じフェーズ番号を複数のサブスクリプションに対して指定した場合は、関連するサブスクリプションが無作為に実行されます。ただし、そのサブスクリプション・グループは、他のフェーズ番号のサブスクリプションに基づいて、フェーズ番号で指定された順序で実行されます。

---

---

サブスクリプションのフェーズ番号を使用して、サブスクリプションをすぐに実行するか遅延させるかを制御することもできます。イベント・マネージャでは、フェーズ番号が 100 以降のサブスクリプションは遅延サブスクリプションとして処理されます。13-43 ページの「[遅延サブスクリプション処理](#)」を参照してください。

サブスクリプションには、実行される処理に応じて、イベント・データに必要なイベント情報が異なります。すべてのイベント情報が必要な場合や、イベントのインスタンスを識別するイベント・キーのみが必要な場合があります。すべてのイベント・データを必要としないサブスクリプションの場合は、イベント・キーをルール・データとして指定すると、パフォーマンスを向上させることができます。ローカルで呼び出されるイベントの場合、イベント・マネージャはサブスクリプションを実行する前に、各サブスクリプションにすべてのイベント・データが必要かどうかをチェックします。必要なイベント・データが渡されていない場合、イベント・マネージャはそのイベントに対してジェネレート関数を実行し、イベント・データを生成します。ただし、イベントへのサブスクリプションがイベント・データを必要としない場合は、サブスクリプションの実行に必要なリソースを最小限に抑えるため、イベント・マネージャはジェネレート関数を実行しません。

---

**注意：** すべてのイベント・データを必要とするサブスクリプションの場合でも、イベントを呼び出した後にできるだけ早く制御をコール側アプリケーションに戻すには、すべてのサブスクリプションを遅延させます。後でそれらのサブスクリプションを実行するまで、イベント・マネージャはジェネレート関数を実行しません。

---

## サブスクリプションに対するアクションの定義

サブスクリプションには、次のタイプの処理を指定できます。

- イベント・メッセージに対する関数の実行
- ワークフロー・プロセスへのイベント・メッセージの送信
- エージェントへのイベント・メッセージの送信

### ルール関数の実行

イベント・メッセージに対して関数を実行するには、実行するルール関数を指定します。また、関数に渡す追加のパラメータを指定することもできます。

Oracle Workflow には、基本的なサブスクリプション処理を実行するための標準のデフォルト・ルール関数が用意されています。この関数は、サブスクリプションに対してルール関数が指定されていない場合に、デフォルトで実行されます。デフォルトのルール関数では、次の処理が行われます。

- ワークフロー・プロセスへのイベント・メッセージの送信（サブスクリプション定義で指定されている場合）
- エージェントへのイベント・メッセージの送信（サブスクリプション定義で指定されている場合）

**参照：** 8-287 ページの「[Default\\_Rule\(\)](#)」を参照してください。

Oracle Workflow には、テストおよびデバッグ、またはその他の目的で使用可能な標準のルール関数もいくつか用意されています。8-286 ページの「[イベント・サブスクリプションのルール関数の API](#)」を参照してください。

サブスクリプション処理は、カスタム・ルール関数を作成することで拡張できます。カスタム・ルール関数は、標準 API に従って定義する必要があります。7-24 ページの「[イベント・サブスクリプションのルール関数の標準 API](#)」を参照してください。

ルール関数は、次のような様々な目的で使用できます。

- 検証を実行する。
- インバウンド・メッセージをアプリケーションの「Receive」メッセージ・ハンドラとして処理する。

- アウトバウンド・メッセージに変更を加える。他のメッセージに関連付ける関連 ID を追加するなど。

---

**注意：** カスタム・ルール関数から

WF\_EVENT\_FUNCTIONS\_PKG.AddCorrelation() をコールすると、カスタム処理中に関連 ID を追加できます。8-300 ページの「[AddCorrelation](#)」を参照してください。

---

ルール関数は、イベント・メッセージに対して読取りまたは書込みを行ったり、任意のデータベース処理を実行したりします。ただし、ルール関数内ではコミットしないでください。コミットは、コール側のアプリケーションが行うため、イベント・マネージャではコミットを発行しません。また、ルール関数を使用して、セキュリティや NLS の設定などの接続コンテキストを変更しないでください。ルール関数がエラーを返した場合、サブスクリプション処理は中止されます。

特定のイベントに対してサブスクリプション処理を実行するときに、連続したステップがいくつか含まれている場合は、再利用できない複雑で特別なルール関数を作成するのではなく、再利用できる単純なルール関数を作成して複数のサブスクリプションをイベントに定義することをお薦めします。複数のサブスクリプションの実行順序を指定するには、サブスクリプションに対してフェーズ値を入力します。

---

**注意：** デフォルト以外のルール関数を入力した場合は、参照する関数に対してワークフロー情報およびエージェント情報を入力できますが、指定されたワークフローおよびエージェントに対してイベント・メッセージは自動的に送信されません。この場合、送信処理をルール関数に明示的に組み込むか、デフォルトのルール関数を使用せずに送信処理を実行する別のサブスクリプションを定義する必要があります。

---

### ワークフロー・プロセスへのイベントの送信

イベントをワークフロー・プロセスに送信するには、送信先のプロセスの項目タイプとプロセス名を指定する必要があります。プロセスの項目キーは、イベント・メッセージに指定されている関連 ID によって決定されます。関連 ID が指定されていない場合は、イベント・キーによって決定されます。

---

**注意：** サブスクリプションの処理中に

WF\_EVENT\_FUNCTIONS\_PKG.AddCorrelation() をコールすると、イベント・メッセージに相関 ID を追加できます。AddCorrelation() を使用するには、ITEMKEY というサブスクリプション・パラメータを入力して、相関 ID を生成する関数を指定する必要があります。関数は次の書式で指定する必要があります。

ITEMKEY=<package\_name.function\_name>

**参照：** 8-300 ページの「[AddCorrelation](#)」を参照してください。

---

イベントをワークフロー・プロセスに送信することにより、事前定義済みの関数を直接実行したり、事前定義済みの受信者にイベントを送信するだけでなく、複雑な処理またはルーティング・ロジックをモデル化できます。たとえば、イベント・メッセージの内容に基づいて、別の関数に分岐したり、サブプロセスを開始したり、通知を送信したり、宛先エージェントを選択したりできます。また、イベント・メッセージ自体を変更することもできます。

ワークフロー・プロセス内では、イベントはイベント・アクティビティ単位に表されます。4-42 ページの「[イベント・アクティビティ](#)」を参照してください。

プロセスがイベントを受信すると、ワークフロー・エンジンは「受信」イベント・アクティビティ・ノードのイベントの詳細の指定に従って、イベント名、イベント・キーおよびイベント・メッセージを項目タイプ属性に格納します。さらに、ワークフロー・エンジンは、イベント・メッセージのパラメータ・リストのパラメータをプロセスの項目タイプとして設定します。パラメータに対応する属性がまだ存在しない場合は、新しい項目タイプ属性を作成します。また、サブスクリプションの GUID（グローバル意識別子）が動的な項目属性として設定されます。このため、ワークフロー・プロセスから、サブスクリプション定義に含まれる他の情報を参照することができます。

イベントが別のワークフロー・プロセスの呼出しイベント・アクティビティによって発生した場合は、そのプロセスの項目タイプと項目キーがイベント・メッセージ内のパラメータ・リストに追加されます。ワークフロー・エンジンは、イベントを受信するプロセスの親として、指定されたプロセスを自動的に設定し、既存の親の設定を上書きします。8-81 ページの「[SetItemParent](#)」を参照してください。

追加パラメータを指定すると、ワークフロー・プロセスの項目属性として設定されます。追加パラメータをサブスクリプションの「パラメータ」フィールドに入力し、サブスクリプションのルール関数で WF\_RULE.SetParametersIntoParameterList() を使用すれば、指定したサブスクリプション・パラメータをイベント・メッセージのパラメータ・リストに設定することができます。ワークフロー・プロセスにイベントが着信すると、それらのイベント・パラメータはワークフロー・プロセスの項目属性として設定されます。8-295 ページの「[SetParametersIntoParameterList](#)」を参照してください。

---

---

**注意：** イベントをワークフロー・プロセスに送信するには、Oracle Workflow から提供されるデフォルトのルール関数を使用するか、送信処理をカスタム・ルール関数に組み込む必要があります。7-24 ページの「[イベント・サブスクリプションのルール関数の標準 API](#)」を参照してください。

---

---

### エージェントへのイベントの送信

イベントをエージェントに送信するには、アウトバウンド・メッセージを送信する送信エージェントまたはインバウンド・メッセージを受信する宛先エージェント、あるいはその両方のエージェントを指定します。

- 宛先エージェントと送信エージェントの両方を指定した場合、イベント・メッセージは送信エージェントのキューに格納され、宛先エージェントに伝播されます。
- 送信エージェントを指定しないで宛先エージェントを指定した場合は、サブスクライバ・システム上のアウトバウンド・エージェントのうち、キュー・タイプが宛先エージェントと一致するものが選択されます。イベント・メッセージは、このアウトバウンド・エージェントのキューに格納され、宛先エージェントに伝播されます。
- 宛先エージェントを指定しないで送信エージェントを指定した場合、イベント・メッセージは送信エージェントのキューに格納されますが、受信者は指定されていません。
  - － 送信エージェントが複数コンシューマ・キューを使用するときに、サブスクライバ・リストが定義されている場合は、宛先エージェントを省略できます（標準の Workflow キュー・ハンドラは複数コンシューマ・キューでのみ機能します）。この場合、キューのサブスクライバ・リストによって、メッセージをデキューできるコンシューマが決定されます。キューに対してサブスクライバ・リストが定義されていない場合、イベント・メッセージはエラー処理のために WF\_ERROR キューに格納されます。

---

---

**注意：** Oracle Advanced Queuing が使用する複数コンシューマ・キューのサブスクライバ・リストは、Oracle Workflow Business Event System のイベント・サブスクリプションとは異なります。詳細は、『Oracle9i アプリケーション開発者ガイド - アドバンスド・キューイング』の「サブスクリプション・リストおよび受信者リスト」を参照してください。

---

---

- － 送信エージェントが単一コンシューマ・キューを使用するときに、カスタム・キュー・ハンドラが定義されている場合は、宛先エージェントを省略できます。単一コンシューマ・キューの場合、コンシューマを指定する必要はありません。

受信者がメッセージをデキューするときの、優先順位を指定することができます。メッセージは、優先順位の昇順でデキューされます。



---

**注意：** イベントをエージェントに送信するには、Oracle Workflow から提供されるデフォルトのルール関数を使用するか、送信処理をカスタム・ルール関数に組み込む必要があります。7-24 ページの「[イベント・サブスクリプションのルール関数の標準 API](#)」を参照してください。

---

伝播したイベント・メッセージを受信者にすぐに送信せずに後で送信する場合は、イベント・メッセージの `SEND_DATE` 属性を未来日付に設定します。設定する送信日は、サブスクリプションの処理期間中で、イベントが送信される前でなければなりません。つまり、送信処理が開始される前に、直前のサブスクリプションまたはルール関数の初期処理が実行されている期間です。イベント・メッセージは宛先エージェントに伝播されますが、指定した日付までデキューされません。

## サブスクリプションの識別情報の定義

サブスクリプション定義を自動的に作成するプログラムを使用する場合は、プログラム独自の名前と簡単な識別子をサブスクリプションの所有者名および所有者タグとして設定できます。この識別情報を使用して、所有するサブスクリプションをプログラムから検索することができます。所有者名および所有者タグは、「サブスクリプションの編集」画面から必要に応じて手動で更新できます。

## 遅延サブスクリプション処理

イベントが発生したときにイベントのサブスクリプションをすぐに実行しない場合は、サブスクリプションを遅延できます。サブスクリプションを遅延すると、コール側アプリケーションに制御を早く戻すことができ、イベント・マネージャは負荷の大きいサブスクリプション処理を後で実行できます。

サブスクリプションは、次の 3 つの方法で遅延できます。

- `SEND_DATE` 属性に未来日付を指定して、イベントを呼び出します。この方法は、ローカルで呼び出されたイベントに対するすべてのサブスクリプション処理を特定の有効日まで遅延するときに使用します。
- イベントに対するサブスクリプションのフェーズ番号を 100 以降に定義します。この方法は、ローカル・イベントまたは外部イベントの特定のサブスクリプション処理を遅延するときに使用します。
- イベントを呼び出す前に、イベント・マネージャのディスパッチ・モードを遅延処理に設定します。この方法は、ローカルで呼び出されるイベントに対するサブスクリプション処理をすべて遅延するときに使用できます。ただし、この方法は特別な状況でのみ使用してください。

これらのいずれかの方法によってイベントのサブスクリプション処理が遅延されると、イベント・メッセージは `WF_DEFERRED` エージェントに関連付けられた標準 `WF_DEFERRED` キューに格納されます。`WF_DEFERRED` エージェントを監視するには、リスナーをスケ

ジュールする必要があります。13-62 ページの「ローカル・インバウンド・エージェントのリスナーのスケジュール」を参照してください。

リスナーは、優先順位に従って WF\_DEFERRED エージェントからイベント・メッセージをデキューします。イベント・メッセージのソース・タイプ（「ローカル」または「外部」）は保持されます。これらのイベントに対するサブスクリプション処理の遅延時間は、リスナーに定義されているスケジュールおよび指定した有効日（未来日付のイベントの場合）によって決まります。

### 未来日付の送信日を使用したサブスクリプション処理の遅延

特定の未来有効日までローカル・イベントのサブスクリプション処理を遅延するには、SEND\_DATE 属性に特定の未来有効日を設定してイベントを呼び出します。たとえば、新しい従業員が採用されたらすぐにその従業員の情報を人事管理アプリケーションに入力し、給与計算の処理は従業員の勤務開始日まで遅延します。

イベントが未来送信日で呼び出されると、イベント・マネージャはそのイベント・メッセージをすぐに WF\_DEFERRED キューに格納し、イベントのサブスクリプションは実行しません。そのイベントに対するすべてのサブスクリプションは、フェーズ番号にかかわらずすべて遅延されます。イベントはその送信日まで WAIT 状態になります。送信日になると、イベント・メッセージはデキュー可能になり、エージェント・リスナーが WF\_DEFERRED キュー上で実行されたときにデキューされます。サブスクリプション処理の遅延時間は、指定した送信日およびリスナーに定義されているスケジュールによって決まります。

リスナーがイベント・メッセージをデキューすると、イベント・マネージャが ERROR\_SUBSCRIPTION 属性のサブスクリプション ID をチェックします。イベント・メッセージにサブスクリプション ID が含まれている場合、つまり、イベントが呼び出された直後にそのイベントに対するすべてのサブスクリプション処理が遅延された場合、イベント・マネージャはイベントに対するすべてのサブスクリプションをフェーズの昇順で実行します。

---

**注意：** イベントが呼び出されたときにイベントが遅延されていなかった場合は、そのイベントが WF\_DEFERRED キューからデキューされるときに、イベント・マネージャはそのイベントに対する適切なサブスクリプションをすべて実行します。サブスクリプションのフェーズ番号は無視されます。100 以降のフェーズ番号が付いている場合でも、サブスクリプションは遅延されません。

---

### 関連項目：

8-268 ページ「Raise」

### サブスクリプションのフェーズ番号を使用したサブスクリプション処理の遅延

サブスクリプションのフェーズ番号を使用して、サブスクリプションがすぐに実行されるか遅延されるかを制御することもできます。イベント・マネージャは、フェーズ番号が 100 以

降のサブスクリプションを遅延サブスクリプションとして処理します。「ローカル」および「外部」のサブスクリプションをこの方法で遅延できます。

---

**注意：** イベントがローカルで呼び出されるときにこの遅延方法を使用するには、そのイベントが未来日付の送信日で呼び出され、イベント・マネージャのサブスクリプション処理が通常の同期モードに設定されている必要があります。それ以外の場合、イベント・メッセージは WF\_DEFERRED キューに格納され、イベントがデキューされるまでイベント・マネージャはサブスクリプションを実行しません。

---

トリガー・イベントが呼び出されるか着信すると、100 以降のフェーズ番号のサブスクリプションを検出するまで、イベント・マネージャはそのイベントに対するサブスクリプションをフェーズ番号順に実行します。イベント・マネージャは、そのサブスクリプションをイベント・メッセージ内の `ERROR_SUBSCRIPTION` 属性に設定し、サブスクリプション・プロパティに指定されている優先度を `PRIORITY` 属性に設定します。イベント・メッセージは、標準 WF\_DEFERRED キューに格納されます。

サブスクリプション処理の遅延時間は、WF\_DEFERRED エージェントを監視するエージェント・リスナーに定義されているスケジュールによって決まります。リスナーがイベント・メッセージをデキューすると、イベント・マネージャは `ERROR_SUBSCRIPTION` 属性のサブスクリプション ID をチェックします。サブスクリプション ID が存在する場合、つまり、そのサブスクリプション以降のサブスクリプション処理が遅延されている場合、イベント・マネージャは最初にそのサブスクリプションを実行してから、そのサブスクリプション以降のフェーズ番号を持つイベントについてすべてのサブスクリプションを実行します。

---

**注意：** イベント・マネージャは、イベント・メッセージに設定されているサブスクリプションのフェーズ番号で、サブスクリプション処理を再開します。そのイベントが最初に処理されたときにフェーズ番号が 100 のサブスクリプションが存在しなかった場合は、フェーズ番号 100 から開始されない場合があります。

---

### イベント・マネージャのディスパッチ・モードを使用したサブスクリプション処理の遅延

イベントをローカル・アプリケーションから呼び出す場合、呼び出すときにそのイベントのサブスクリプション処理をすべて遅延することもできます。この方法を実行するには、`Raise()` API をコールする直前に、遅延（非同期）処理を示す「ASYNC」モードで `SetDispatchMode()` API をコールします。このメソッドは、アプリケーション内で遅延をハードコーディングする必要があるため、特別な状況でのみ使用してください。柔軟性を維持しながら、アプリケーションに影響を与えずにサブスクリプション処理を変更するには、未来日付の送信日でイベントを呼び出すか、サブスクリプションのフェーズ番号を使用して一部またはすべてのサブスクリプションを遅延として指定します。

ディスパッチ・モードが遅延処理に設定された後にイベントが呼び出されると、イベント・マネージャはイベント・メッセージをすぐに WF\_DEFERRED キューに格納し、イベントの

サブスクリプションは実行しません。イベントのすべてのサブスクリプションは、フェーズ番号にかかわらずすべて遅延されます。

サブスクリプション処理の遅延時間は、WF\_DEFERRED エージェントを監視するエージェント・リスナーに定義されているスケジュールによって決まります。リスナーがイベント・メッセージをデキューすると、イベント・マネージャは ERROR\_SUBSCRIPTION 属性のサブスクリプション ID をチェックします。イベント・メッセージにサブスクリプション ID が含まれている場合、つまり、イベントが呼び出された直後にそのイベントに対するすべてのサブスクリプション処理が遅延された場合、イベント・マネージャはイベントのすべてのサブスクリプションをフェーズの昇順で実行します。

---

**注意：** イベントが呼び出されたときにイベントが遅延されていなかった場合は、そのイベントが WF\_DEFERRED キューからデキューされるときに、イベント・マネージャはそのイベントに対する適切なサブスクリプションをすべて実行します。サブスクリプションのフェーズ番号は無視されます。100 以降のフェーズ番号が付いている場合でも、サブスクリプションは遅延されません。

---

### 関連項目：

8-281 ページ [「SetDispatchMode」](#)

## エラー処理

ルール関数から「WARNING」または「ERROR」のステータス・コードが返され、サブスクリプションの処理中に警告状態またはエラーが発生したことを示している場合、イベント・マネージャはイベント・メッセージを WF\_ERROR エージェントに関連付けられている標準 WF\_ERROR キューに格納します。「WARNING」ステータスの場合は、イベントのサブスクリプション処理が続行されます。「ERROR」ステータスの場合は、イベントのサブスクリプション処理が中止され、イベントに対して実行されたすべてのサブスクリプションがロールバックされます。

WF\_ERROR エージェントを監視するには、リスナーをスケジュールする必要があります。このリスナーによって WF\_ERROR キューのイベント・メッセージがデキューされると、そのメッセージに対して「エラー」ソース・タイプが割り当てられます。イベント・マネージャは、「エラー」ソース・タイプを持つそのイベントまたは Any イベントへのサブスクリプションを、ローカル・システム単位に検索および実行します。サブスクリプションが見つからない場合は、「エラー」ソース・タイプの Unexpected イベントへのサブスクリプションがローカル・システム単位に実行されます。

Oracle Workflow には、「エラー」ソース・タイプの Unexpected イベントに対して、1つの事前定義済サブスクリプションが用意されています。カスタム・エラー・サブスクリプションをイベントに定義していない場合は、このサブスクリプションによってデフォルトのエラー処理が実行されます。イベント・メッセージは、「システム:エラー」項目タイプの「デフォルト・イベント・エラー」プロセスに送信されます。

---

**注意：** Unexpected イベントまたはそのイベントへの事前定義済のエラー・サブスクリプションの定義は、変更したり使用不可にしないでください。これらを変更したり使用不可にすると、イベント・マネージャはイベントおよびサブスクリプション処理に対してデフォルトのエラー処理を実行できなくなります。

---

「デフォルト・イベント・エラー」プロセスでは、システム管理者に通知が送信されます。警告状態の場合、応答する必要はありません。エラーの場合、システム管理者はイベント・サブスクリプション処理を強制終了または再試行することができます。14-15 ページの「Unexpected イベント」および 6-32 ページの「デフォルト・イベント・エラー・プロセス」を参照してください。

特定のイベントに対してカスタム・エラー処理を設定するには、「エラー」ソース・タイプのイベントへのサブスクリプションを定義し、サブスクリプション・アクションとして実行するカスタム処理を指定します。この場合、このエラー・イベントは Unexpected イベントでなくなるため、イベント・マネージャはデフォルトのエラー処理を実行しません。デフォルトのエラー処理のかわりに、設定したカスタム・エラー処理が実行されます。

---

**注意：** ルール関数で例外が発生した場合は、イベント・マネージャによってイベントへのすべてのサブスクリプションがロールバックされ、コール側アプリケーションにエラーが呼び出されます。この場合、イベント・メッセージは WF\_ERROR キューに格納されません。

---

#### 関連項目：

7-24 ページ「イベント・サブスクリプションのルール関数の標準 API」

13-62 ページ「ローカル・インバウンド・エージェントのリスナーのスケジュール」

▶ イベント・サブスクリプションの定義

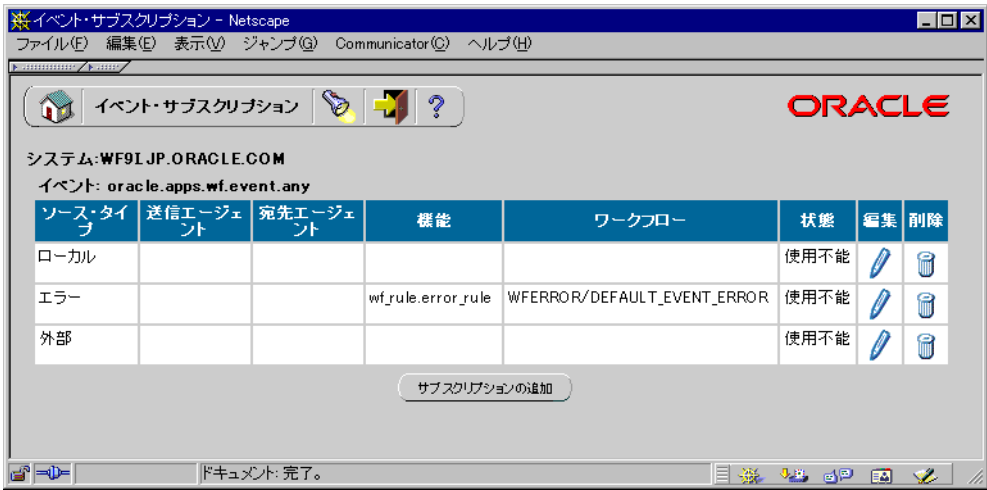
1. Web ブラウザを使用して、次の URL に接続します。

<webagent>/wf\_event\_html.listsubscriptions

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

**注意：** 「イベント・サブスクリプション」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。



2. 「イベント・サブスクリプション」画面に既存のサブスクリプションのリストが表示されます。サブスクライバ・システムおよびトリガー・イベント別にグループ化されています。「イベント・サブスクリプション」画面には、各サブスクリプションのソース・タイプ、送信エージェント、宛先エージェント、関数、ワークフローおよび状態の一覧が表示されます。

「サブスクリプションの追加」ボタンを選択して「サブスクリプションの編集」画面を開きます。

3. 「サブスクライバ」領域に、サブスクリプションを実行するシステムを入力します。「システム」フィールドの上矢印のアイコンをクリックすると、選択可能なシステムのリストが表示されます。13-23 ページの「[値リストの使用](#)」を参照してください。
4. 「トリガー条件」領域の「ソース・タイプ」フィールドに、サブスクリプションを適用するソース・システムのタイプを指定します。
  - ローカル： サブスクリプションは、サブスクライバ・システムで発生したイベントにのみ適用されます。
  - 外部： サブスクリプションは、サブスクライバ・システム上のインバウンド・エージェントが受信したイベントにのみ適用されます。

---

---

**注意：** サブスクライバ・システム上のインバウンド・エージェントが受信したイベント・メッセージは、送信元エージェントがリモート・システム上に存在するかローカル・システム上に存在するかにかかわらず、すべて外部ソースと見なされます。

---

---

- エラー： サブスクリプションは、WF\_ERROR キューからデキューされたエラー・イベントにのみ適用されます。
- 5. サブスクリプションを適用するイベントを「イベント・フィルタ」フィールドに入力します。個々のイベントまたはイベント・グループを指定できます。「イベント・フィルタ」フィールドの上矢印のアイコンをクリックすると、選択可能なイベントのリストが表示されます。13-23 ページの「[値リストの使用](#)」を参照してください。
- 6. オプションで、サブスクリプションを適用するソース・エージェントを入力します。ソース・エージェントを指定すると、サブスクリプションは、トリガー・イベントがそのエージェントから着信した場合にのみ実行されます。「ソース・エージェント」フィールドの上矢印のアイコンをクリックすると、選択可能なエージェントのリストが表示されます。13-23 ページの「[値リストの使用](#)」を参照してください。

---

---

**注意：** ほとんどの場合、「ソース・エージェント」フィールドは空白のままにします。

---

---

- 7. 同じイベントに複数のサブスクリプションを適用する場合、その実行順序を指定するときは、「実行制御」領域にサブスクリプションのフェーズ番号を入力します。フェーズ番号を使用して、サブスクリプションがすぐに実行されるか遅延されるかを制御することもできます。
- 8. サブスクリプションの状態として「使用可能」または「使用不能」を選択します。サブスクリプションを使用不能にしても、そのサブスクリプションは「イベント・サブスクリプション」リストに表示され、参照できます。ただし、このサブスクリプションを使用して、イベントに応答することはできません。
- 9. 「ルール・データ」フィールドで、サブスクリプションに必要なイベント情報を指定します。
  - キー： サブスクリプションにはイベント・キーのみが必要です。
  - メッセージ： サブスクリプションにはすべてのイベント・データが必要です。
- 10. 「処理」領域で、トリガー・イベントの発生時に実行するサブスクリプション処理を定義します。サブスクリプションには、次の処理を指定できます。
  - イベント・メッセージに対する関数の実行
  - ワークフロー・プロセスへのイベント・メッセージの送信



- エージェントへのイベント・メッセージの送信

11. イベント・メッセージに対して関数を実行する場合は、実行するルール関数を入力します。ルール関数は、標準 API に従って定義する必要があります。7-24 ページの「[イベント・サブスクリプションのルール関数の標準 API](#)」を参照してください。

ルール関数を指定しない場合は、Oracle Workflow によってデフォルトのルール関数が実行され、イベント・メッセージは指定したワークフロー・プロセスおよびエージェントに送信されます。

---

**注意：** デフォルト以外のルール関数を入力した場合、指定されたワークフローおよびエージェントにイベント・メッセージは自動的に送信されません。この場合、送信処理をカスタム・ルール関数に明示的に組み込む必要があります。ただし、「処理」領域には、参照する関数に対するワークフロー情報およびエージェント情報を入力できます。

---

12. イベント・メッセージをワークフロー・プロセスに送信する場合は、プロセスの項目タイプを「ワークフロー項目タイプ」フィールドに、プロセスの名前を「ワークフロー・プロセス名」フィールドに入力します。各フィールドの上矢印のアイコンをクリックすると、選択可能な値のリストが表示されます。13-23 ページの「[値リストの使用](#)」を参照してください。

---

**注意：** 「ワークフロー・プロセス」フィールドの値リストには、指定した項目タイプの実行可能プロセスのみが表示されます。

---

13. イベントをエージェントに送信する場合は、アウトバウンド・メッセージを送信する送信エージェントまたはインバウンド・メッセージを受信する宛先エージェント、あるいはその両方のエージェントを入力します。各フィールドの上矢印のアイコンをクリックすると、選択可能な値のリストが表示されます。13-23 ページの「[値リストの使用](#)」を参照してください。

- 宛先エージェントと送信エージェントの両方を指定した場合、イベント・メッセージは送信エージェントのキューに格納され、宛先エージェントに伝播されます。
- 送信エージェントを指定しないで宛先エージェントを指定した場合は、サブスクライバ・システム上のアウトバウンド・エージェントのうち、キュー・タイプが宛先エージェントと一致するものが選択されます。イベント・メッセージは、このアウトバウンド・エージェントのキューに格納され、宛先エージェントに伝播されます。
- 宛先エージェントを指定しないで送信エージェントを指定した場合、イベント・メッセージは送信エージェントのキューに格納されますが、受信者は指定されていません。送信エージェントは、サブスクライバ・リストが定義された複数コンシューマ・キューまたは単一コンシューマ・キューを使用する必要があります。

---

---

**注意：** 送信エージェントは、サブスクライバ・システム上に存在する必要があります。「送信エージェント」フィールドの値リストには、送信方向のエージェントのみが表示されます。

「宛先エージェント」フィールドの値リストには、インバウンド方向のエージェントのみが表示されます。

---

---

14. イベント・メッセージをエージェントに送信する場合は、受信者がメッセージをデキューするときの優先順位として、「標準」、「高」または「低」を選択します。
15. オプションで、「パラメータ」フィールドにルール関数の追加パラメータを入力します。複数のパラメータを区切るには、スペースを使用します。各パラメータの名前および値は、次の書式で指定します。

`<name1>=<value1> <name2>=<value2> ... <nameN>=<valueN>`

---

---

**注意：** イベント・メッセージをワークフロー・プロセスに送信するとき、追加パラメータを指定すると、ワークフロー・プロセスの項目属性として設定されます。追加パラメータをサブスクリプションの「パラメータ」フィールドに入力し、サブスクリプションのルール関数で `WF_RULE.SetParametersIntoParameterList()` を使用すれば、指定したサブスクリプション・パラメータをイベント・メッセージのパラメータ・リストに設定することができます。ワークフロー・プロセスにイベントが着信すると、それらのイベント・パラメータはワークフロー・プロセスの項目属性として設定されます。8-295 ページの「[SetParametersIntoParameterList](#)」を参照してください。

---

---

16. 「文書」領域を利用して、サブスクリプションを所有するプログラムまたはアプリケーションを識別することができます。プログラム名を「所有者名」フィールドに、プログラム ID を「所有者タグ」フィールドに入力してください。「サブスクリプションの編集」画面でサブスクリプションを手動で定義する場合は、「所有者名」および「所有者タグ」を入力する必要はありません。ただし、サブスクリプション定義を自動的に作成するプログラムを使用する場合、これらのフィールドにはそのプログラムで設定された所有者情報が表示されます。この情報は、「サブスクリプションの編集」画面から必要に応じて手動で更新できます。
17. オプションで、サブスクリプションの説明を入力します。
18. 「送信」ボタンを選択してサブスクリプションを保存し、「イベント・サブスクリプション」画面に戻ります。「イベント・サブスクリプション」画面に、更新されたサブスクリプションのリストが表示されます。  
  
「取消」ボタンを選択し、サブスクリプションを保存しないで「イベント・サブスクリプション」画面に戻ることもできます。

## ▶ イベント・サブスクリプションの検索

1. Web ブラウザを使用して、次の URL に接続します。

<webagent>/wf\_event\_html.findsubscription

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

---

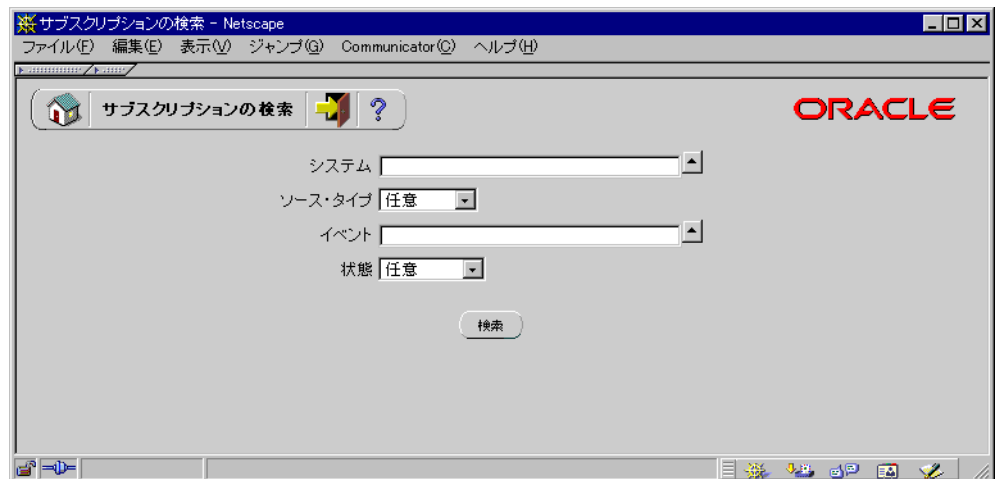
**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

---

---

**注意：** 「サブスクリプションの検索」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---



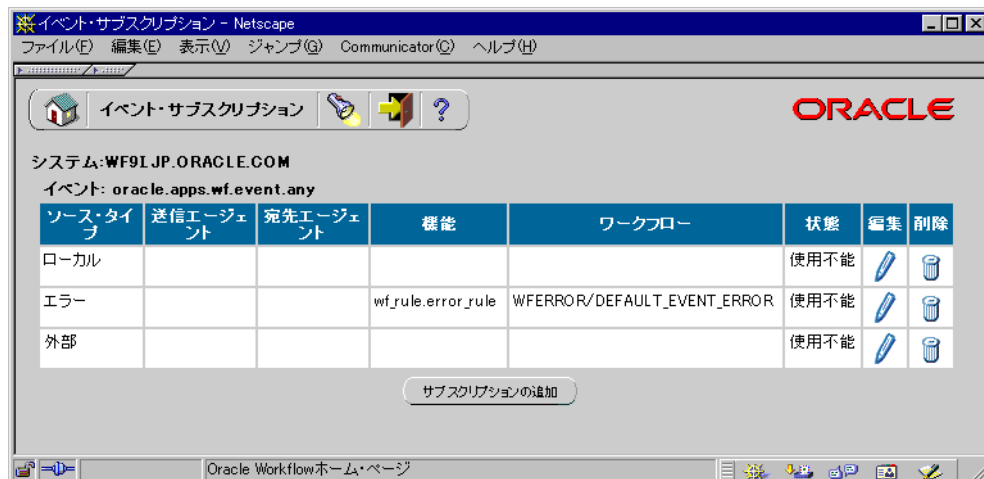
2. 「サブスクリプションの検索」画面が表示されます。「サブスクリプションの検索」画面では、検索基準を入力して特定のイベント・サブスクリプションを検索できます。検索基準として、次の項目を使用できます。

- システム： サブスクリプションを表示するシステムを選択します。このフィールドの上矢印のアイコンをクリックすると、選択可能なシステムのリストが表示されます。13-23 ページの「[値リストの使用](#)」を参照してください。
  - ソース・タイプ： サブスクリプションを表示するソース・システムのタイプとして、「ローカル」、「外部」または「エラー」を選択します。すべてのタイプのソース・システムについてサブスクリプションを表示する場合は、「任意」を選択します。
  - イベント： サブスクリプションを表示するイベントを選択します。このフィールドの上矢印のアイコンをクリックすると、選択可能なイベントのリストが表示されます。13-23 ページの「[値リストの使用](#)」を参照してください。
  - 状態： 表示するサブスクリプションのステータスとして、「使用可能」または「使用不能」を選択します。すべてのステータスのサブスクリプションを表示する場合は、「任意」を選択します。
3. 「検索」ボタンを選択します。「イベント・サブスクリプション」画面に、検索基準と一致したサブスクリプションのリストが表示されます。

検索基準にシステムまたはイベントを含めた場合は、「サブスクリプションの追加」ボタンを選択すると、「サブスクリプションの編集」画面が表示されます。指定したシステム情報およびイベント情報が、「システム」および「イベント・フィルタ」フィールドにそれぞれ自動的に入力されています。検索基準と一致する既存のサブスクリプションが見つからなかった場合は、この方法で新しいサブスクリプションの定義を開始できます。

#### ► イベント・サブスクリプションの更新または削除

1. 「イベント・サブスクリプション」画面で、更新または削除するイベントを選択します。「サブスクリプションの検索」画面を使用して目的のサブスクリプションを検索し、「イベント・サブスクリプション」画面に表示することができます。13-53 ページの「[イベント・サブスクリプションの検索](#)」を参照してください。



- サブスクリプションを更新するには、そのサブスクリプションの「編集」列にある鉛筆アイコンを選択します。「サブスクリプションの編集」画面が表示されます。サブスクリプション定義に変更を加え、変更内容を保存します。13-48 ページの「[イベント・サブスクリプションの定義](#)」を参照してください。
- サブスクリプションを削除するには、そのサブスクリプションの「削除」列にあるごみ箱アイコンを選択して、表示される確認ウィンドウで「OK」を選択します。確認ウィンドウで「取消」ボタンを選択し、サブスクリプションを削除しないで「イベント・サブスクリプション」画面に戻することもできます。

## メッセージ伝播の設定

イベント、システム、エージェントおよびサブスクリプションを定義したら、ビジネス・イベント・システムのメッセージ伝播を設定する必要があります。イベント・マネージャの「セットアップのチェック」Web 画面を使用して、次の手順を実行します。

1. ビジネス・イベント・システムのセットアップをチェックします。
2. ローカル・インバウンド・エージェントのリスナーをスケジュールします。
3. ローカル・アウトバウンド・エージェントの伝播をスケジュールします。

---

**注意：** Oracle Applications embedded Workflow を使用している場合は、ローカル・インバウンド・エージェントのリスナーをスケジュールする必要があります。「セットアップのチェック」Web 画面は、セットアップを確認および検証する場合、およびローカル・アウトバウンド・エージェントの伝播をスケジュールする場合にのみ使用します。

---

エージェントに加えた変更が伝播に必要な物理実装に影響する場合は、伝播設定を再度チェックする必要があります。13-24 ページの「[エージェント](#)」を参照してください。

---

**注意：** ローカル・システムのステータスは、デフォルトでは「使用可能」に設定されます。ビジネス・イベント・システムの設定を終了した後で、イベントの処理に必要なシステム・ステータスを設定する場合は、「グローバル・ワークフロー・プリファレンス」Web 画面を使用します。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

---

## ビジネス・イベント・システムのセットアップのチェック

ビジネス・イベント・システムでメッセージ伝播を有効にするときは、「セットアップのチェック」Web 画面を使用して、必要なパラメータおよびコンポーネントが設定されていることを確認します。

## ▶ ビジネス・イベント・システムのセットアップのチェック

1. Web ブラウザを使用して、次の URL に接続します。

`<webagent>/wf_setup.check_all`

`<webagent>` は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

---

---

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

---

---

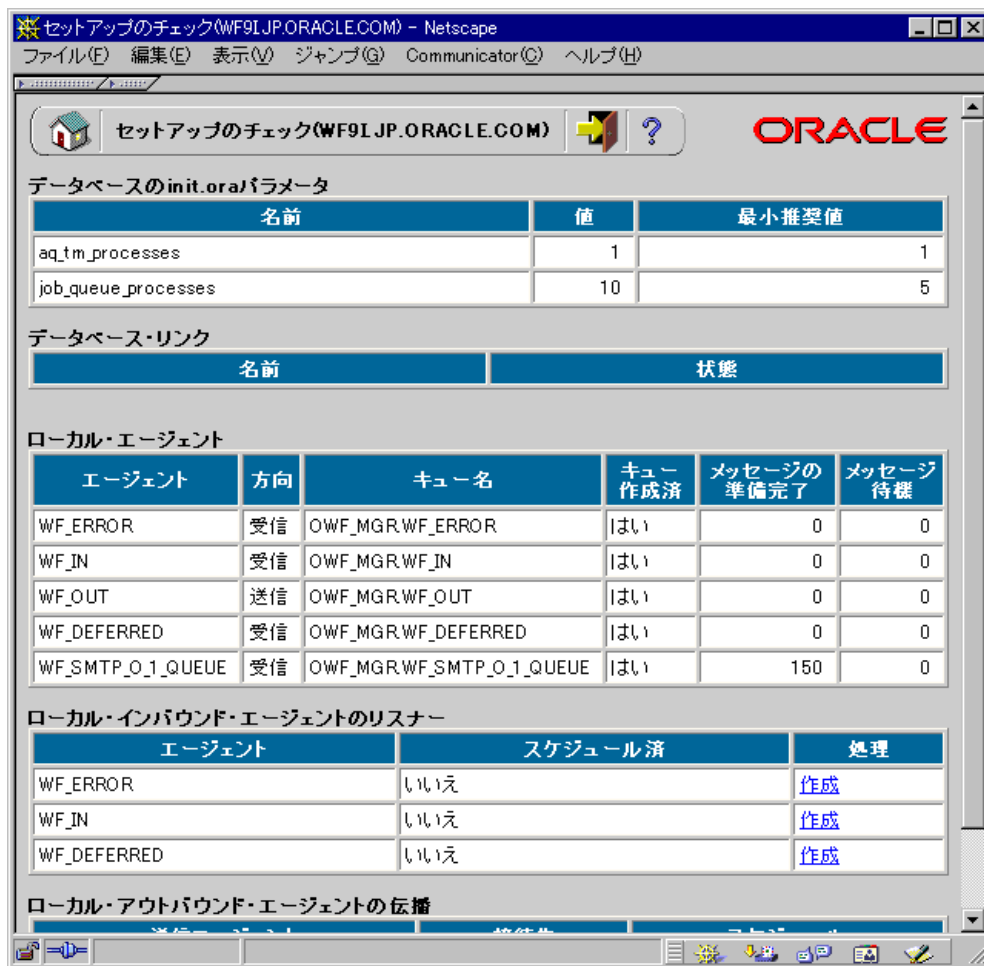
---

---

**注意：** 「セットアップのチェック」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---

---



2. 「セットアップのチェック」画面に、ローカル・システムの伝播に関する設定およびコンポーネントが表示されます。
3. 「データベースの init.ora パラメータ」領域を使用して、AQ に関連するデータベース初期化パラメータの設定を確認します。「セットアップのチェック」画面に、各パラメータに定義されている実際の値および Oracle Workflow の推奨最小値が表示されます。

Oracle8i を使用している場合にこれらのパラメータを変更するときは、データベースの init.ora ファイルの設定を変更します。変更内容を有効にするには、変更の保存後にデータベースを再起動する必要があります。



Oracle9i を使用している場合は、init.ora ファイルのパラメータを変更してデータベースを再起動します。ALTER SYSTEM 文を使用して、インスタンスの継続時間中に AQ\_TM\_PROCESSES および JOB\_QUEUE\_PROCESSES の値を動的に変更することもできます。

- **AQ\_TM\_PROCESSES:** このパラメータは、Oracle Advanced Queuing (AQ) のタイム・マネージャ・プロセスを使用可能にします。タイム・マネージャ・プロセスは、Oracle Workflow 標準の「待機」アクティビティなど、Oracle Workflow がキュー内の遅延イベントを監視するときが必要です。Oracle Workflow のタイム・マネージャ・プロセスの最小推奨数は 1 です。
- **JOB\_QUEUE\_INTERVAL:** Oracle8i を使用している場合は、ジョブ・キュー間隔を指定して、インスタンス内で各 SNP ジョブ・キュー・プロセスの待機が解除される頻度を決定します。Oracle Workflow では、ジョブ・キュー間隔を AQ 伝播スケジュールに定義されている待機時間パラメータの値以下にして、指定した待機時間が経過したときにキューのメッセージが再度チェックされるようにする必要があります。Oracle Workflow の推奨ジョブ・キュー間隔は、5 秒です。

---

**注意：** JOB\_QUEUE\_INTERVAL パラメータは Oracle9i でサポートされていないため、Oracle9i を使用している場合、「セットアップのチェック」画面にこのパラメータは表示されません。このため、このパラメータの値を設定する必要はありません。

---

- **JOB\_QUEUE\_PROCESSES:** このパラメータは、インスタンスの SNP ジョブ・キュー・プロセス数を定義します。Oracle Workflow のジョブ・キュー・プロセスでは、ビジネス・イベント・システムのイベント・メッセージの伝播を AQ キュー単位に処理する必要があります。メッセージ伝播を有効にするには、1 つ以上のジョブ・キュー・プロセスを開始する必要があります。Oracle Workflow のプロセスの最小推奨数は 2 です。

---

**注意：** AQ に関連するイベントをデータベース・レベルで詳細にトレースするには、EVENT という別の初期化パラメータを使用できます。次の行を init.ora ファイルに追加します。

```
event = "24040 trace name context forever, level 10"
```

データベースを再起動してこの変更を有効にします。このパラメータを使用すると、大きなサイズのトレース・ファイルが生成される場合がありますので注意してください。

---

4. 「データベース・リンク」領域を使用して、データベース・リンクを確認します。「セットアップのチェック」画面には、エージェントのアドレスで参照される各データベース・リンクの名前とステータスが表示されます。必要なデータベース・リンクが存在し

ていない場合は、作成してください。2-90 ページの「[データベース・リンクの作成](#)」を参照してください。

---

**注意：** エージェントのアドレスに使用されているデータベース・リンク名が、データベース・リンクの作成時に指定されたデータベース・リンク名と正確に一致していることを確認してください。

---

5. 「ローカル・エージェント」領域を使用して、ローカル・システム上のエージェントに設定されているキューを確認します。「セットアップのチェック」Web 画面に、各エージェントの名前および方向、エージェントに割り当てられているキューの名前、キューが作成されているかどうか、キューのメッセージのうち処理済のメッセージ数と処理可能なメッセージ数、および処理待ちのメッセージ数が表示されます。必要なキューが存在しない場合は、作成してください。13-24 ページの「[エージェント](#)」および 2-91 ページの「[キューの設定](#)」を参照してください。

---

**注意：** 「ローカル・エージェント」領域には、ビジネス・イベント・システムのエージェントの他に、通知メーラー SMTP キュー (WF\_SMTP\_O\_1\_QUEUE) に定義されている標準エージェントも表示されます。このエージェントの情報を表示して、通知メーラー・キューにある通知メッセージの数をチェックできます。ただし、WF\_SMTP\_O\_1\_QUEUE エージェントは、ビジネス・イベント・システムでは使用されません。13-27 ページの「[標準エージェント](#)」を参照してください。

---

6. 「ローカル・インバウンド・エージェントのリスナー」領域を使用して、インバウンド・イベント・メッセージを受信するようにリスナーをスケジュールします。13-62 ページの「[ローカル・インバウンド・エージェントのリスナーのスケジュール](#)」を参照してください。
7. 「ローカル・アウトバウンド・エージェントの伝播」領域を使用して、イベント・メッセージを送信するように伝播をスケジュールします。13-67 ページの「[ローカル・アウトバウンド・エージェントの伝播のスケジュール](#)」を参照してください。

### 関連項目：

『Oracle9i データベース・リファレンス』

『Oracle9i アプリケーション開発者ガイド - アドバンスト・キューイング』

## ローカル・インバウンド・エージェントのリスナーのスケジュール

「セットアップのチェック」Web 画面を使用して、ローカル・システムのインバウンド・エージェントのリスナーをスケジュールします。ビジネス・イベント・システムでは、インバウンド・イベント・メッセージを受信するようにリスナーをスケジュールする必要があります。

エージェントのリスナーをスケジュールすると、指定した実行日にエージェントのキューの監視が開始され、インバウンド・イベント・メッセージがデキューされます。イベント・メッセージが着信すると、「外部」ソース・タイプのそのイベントへの有効なサブスクリプションおよび「外部」ソース・タイプの Any イベントへの有効なサブスクリプションが、イベント・マネージャによってローカル・システム単位に検索および実行されます。

リスナーは、エージェントのキューにあるすべてのイベント・メッセージがデキューされた後で終了します。Oracle Workflow は、リスナーを指定した間隔で無期限に再実行します。

リスナーをスケジュールした後で設定を更新すれば、スケジュールを変更できます。また、リスナーを削除すれば、リスナーを完全に停止できます。

ビジネス・イベント・システムの遅延サブスクリプション処理およびエラー処理を有効にするには、標準 WF\_DEFERRED エージェントと標準 WF\_ERROR エージェントのリスナーをそれぞれスケジュールする必要があります。また、標準の WF\_IN エージェントをイベント・メッセージの伝播に使用する場合は、そのエージェントのリスナーもスケジュールしてください。

---

---

**注意：** Oracle Applications embedded Workflow を使用している場合は、コンカレント・マネージャを使用して「ワークフロー・エージェント・リスナー」コンカレント・プログラムを発行し、ローカル・インバウンド・エージェントのリスナーをスケジュールする必要があります。「セットアップのチェック」Web 画面は、セットアップを確認および検証する場合、およびローカル・アウトバウンド・エージェントの伝播をスケジュールする場合にのみ使用します。

Oracle Applications Manager を実装している環境で Oracle Applications embedded Workflow を使用している場合は、Oracle Workflow Manager を使用して「ワークフロー・エージェント・リスナー」コンカレント・プログラムを発行および管理することができます。詳細は、Oracle Applications Manager のオンライン・ヘルプを参照してください。

また、Oracle9i リリース 2 で利用可能なスタンドアロン版の Oracle Workflow を使用している場合は、Oracle Enterprise Manager から利用可能なスタンドアロン版の Oracle Workflow Manager コンポーネントを使用して、「ワークフロー・エージェント・リスナー」データベース・ジョブを発行および管理できます。詳細は、Oracle Workflow Manager のオンライン・ヘルプを参照してください。

---

---

---

---

**注意：** 標準 WF SMTP\_O\_1\_QUEUE エージェントに対してエージェント・リスナーを実行しないでください。このエージェントは、通知メーラー SMTP キューに対して定義されており、ビジネス・イベント・システムによって使用されません。13-27 ページの「[標準エージェント](#)」を参照してください。

---

---

**関連項目：**

13-24 ページ [「エージェント」](#)

8-277 ページ [「Listen」](#)

8-279 ページ [「ワークフロー・エージェント・リスナー」コンカレント・プログラム](#)

16-6 ページ [「wfagtlst.sql」](#)

➤ **ローカル・インバウンド・エージェントのリスナーのスケジュール**

1. Web ブラウザを使用して、次の URL に接続します。

```
<webagent>/wf_setup.check_all
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

---

---

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

---

---

---

---

**注意：** 「セットアップのチェック」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---

---

The screenshot shows the 'Setup Check' (セットアップのチェック) page in Netscape. The page title is 'セットアップのチェック(WF9LJP.Oracle.COM) - Netscape'. The page content includes:

**データベースのinit.oraパラメータ**

名前	値	最小推奨値
aq_tm_processes	1	1
job_queue_processes	10	5

**データベース・リンク**

名前	状態

**ローカル・エージェント**

エージェント	方向	キュー名	キュー作成済	メッセージの準備完了	メッセージ待機
WF_ERROR	受信	OWF_MGR.WF_ERROR	はい	0	0
WF_IN	受信	OWF_MGR.WF_IN	はい	0	0
WF_OUT	送信	OWF_MGR.WF_OUT	はい	0	0
WF_DEFERRED	受信	OWF_MGR.WF_DEFERRED	はい	0	0
WF_SMTP_O_1_QUEUE	受信	OWF_MGR.WF_SMTP_O_1_QUEUE	はい	150	0

**ローカル・インバウンド・エージェントのリスナー**

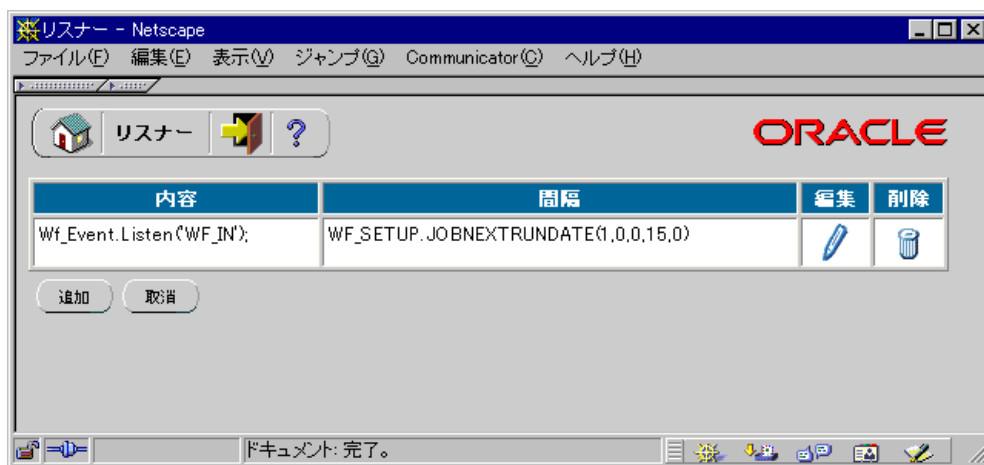
エージェント	スケジュール済	処理
WF_ERROR	いいえ	<a href="#">作成</a>
WF_IN	いいえ	<a href="#">作成</a>
WF_DEFERRED	いいえ	<a href="#">作成</a>

**ローカル・アウトバウンド・エージェントの伝播**

- 「セットアップのチェック」画面に、ローカル・システムの伝播に関する設定およびコンポーネントが表示されます。
- 「ローカル・インバウンド・エージェントのリスナー」領域で、リスナーをスケジュールするエージェントを選択します。「セットアップのチェック」画面には、各ローカル・インバウンド・エージェントのエージェント名および「スケジュール済」ステータスの一覧が表示されます。
- エージェントに対してリスナーがスケジュールされていない場合、エージェントの「スケジュール済」ステータスは「いいえ」になります。エージェントの最初のリスナーを作成するには、そのエージェントの「処理」列にある「作成」リンクを選択します。「リスナーの編集」画面に、選択したエージェントの名前が表示されます。

エージェントに対して1つ以上のリスナーがすでにスケジュールされている場合、エージェントの「スケジュール済」ステータスは「はい」になります。追加のリスナーを作成するには、そのエージェントの「処理」列にある「編集」リンクを選択します。「リスナー」画面に、エージェントに対してすでにスケジュールされているリスナーの一覧が表示されます。「リスナー」画面には、エージェントに対して実行される各リスニング・プロシージャおよびプロシージャの再送信間隔の一覧が表示されます。

「追加」ボタンを選択します。「リスナーの編集」画面に、選択したエージェントの名前が表示されます。「取消」ボタンを選択し、新しいリスナーを作成しないで「セットアップのチェック」画面に戻ることもできます。



5. 「リスナーの編集」画面で、リスナーの実行開始日を「実行日」フィールドに入力します。現在のシステム日付にリスナーを開始する場合は、このフィールドを空白のままにします。

---

**注意：** 設定したユーザー設定項目の日付書式に時間設定が含まれる場合は、日付以外にリスナーの開始時間を指定できます。時間設定が含まれる場合は、日付のみを指定します。9-6 ページの「[ユーザー設定項目の設定](#)」を参照してください。

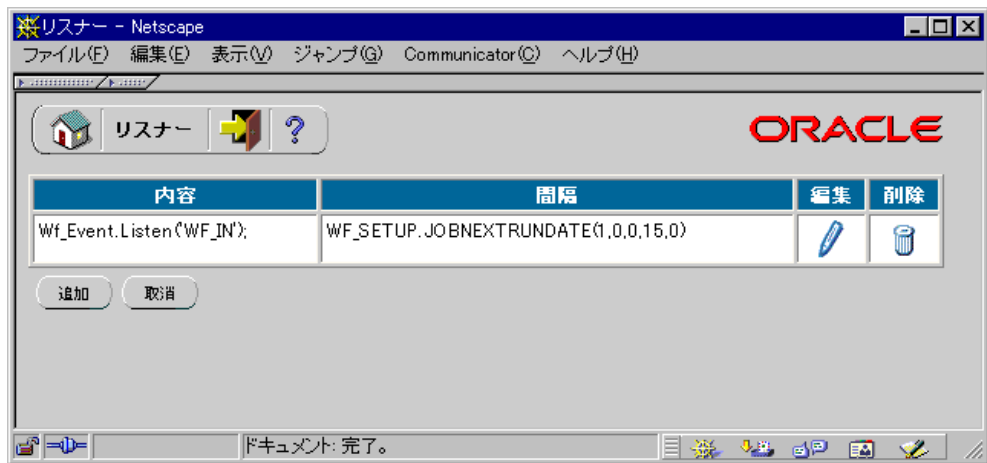
---

6. 「実行頻度」フィールドに、間隔を入力してリスナーの実行頻度を指定します。間隔は、日数、時間、分および秒で指定できます。
7. 「送信」ボタンを選択してリスナーのスケジュールを保存します。エージェントの最初のリスナーをスケジュールする場合は、「セットアップのチェック」画面が表示され、「スケジュール済」ステータスが更新されます。追加のリスナーをスケジュールする場合は、「リスナー」画面が表示され、リスナーのリストが更新されます。

「取消」ボタンを選択し、リスナーを保存しないで「セットアップのチェック」または「リスナー」画面に戻ることもできます。

### ▶ リスナーの更新または削除

1. 「セットアップのチェック」画面の「ローカル・インバウンド・エージェントのリスナー」領域を使用して、更新または削除するインバウンド・エージェントを選択します。13-62 ページの「ローカル・インバウンド・エージェントのリスナーのスケジュール」を参照してください。
2. エージェントの「スケジュール済」ステータスが「はい」の場合は、エージェントに対して 1 つ以上のリスナーがすでにスケジュールされています。「処理」列にある「編集」リンクを選択すると、「リスナー」画面に既存のリスナーの一覧が表示されます。



3. リスナーを更新するには、そのリスナーの「編集」列にある鉛筆アイコンを選択します。「リスナーの編集」画面が表示されます。リスナーのスケジュールに変更を加え、変更内容を保存します。13-62 ページの「ローカル・インバウンド・エージェントのリスナーのスケジュール」を参照してください。
4. リスナーを削除するには、そのリスナーの「削除」列にあるごみ箱アイコンを選択して、表示される確認ウィンドウで「OK」を選択します。確認ウィンドウで「取消」ボタンを選択し、リスナーを削除しないで「リスナー」画面に戻することもできます。

### ローカル・アウトバウンド・エージェントのリスナーのスケジュール

「セットアップのチェック」Web 画面を使用して、ローカル・システムのアウトバウンド・エージェントのリスナーをスケジュールします。ビジネス・イベント・システムからアウトバウンド・イベント・メッセージを送信するには、伝播をスケジュールする必要があります。

イベント・メッセージをエージェントに送信すると、メッセージはアウトバウンド・エージェントに関連付けられたキューに格納されます。メッセージは、伝播によってインバウンド・エージェントに非同期で配信されます。「セットアップのチェック」画面を使用して、



必要な伝播がスケジュールされているかどうかを確認し、SQLNET プロトコルを使用するエージェントに対して AQ 伝播をスケジュールすることができます。

他のプロトコルを使用するエージェントの場合は、外部伝播ロジックを指定する必要があります。13-24 ページの「エージェント」を参照してください。

標準 WF\_OUT エージェントをイベント・メッセージの伝播に使用する場合は、そのエージェントの伝播をスケジュールしてください。

---

**注意：** Oracle Applications Manager を実装している環境で Oracle Applications embedded Workflow を使用している場合は、Oracle Workflow Manager を使用してローカル・アウトバウンド・エージェントの伝播スケジュールを確認することができます。詳細は、Oracle Applications Manager のオンライン・ヘルプを参照してください。

また、Oracle9i リリース 2 で利用可能なスタンドアロン版の Oracle Workflow を使用している場合は、Oracle Enterprise Manager から利用可能なスタンドアロン版の Oracle Workflow Manager コンポーネントを使用して、ローカル・アウトバウンド・エージェントの伝播スケジュールを確認できます。詳細は、Oracle Workflow Manager のオンライン・ヘルプを参照してください。

---

#### 関連項目：

13-24 ページ「エージェント」

### ▶ ローカル・アウトバウンド・エージェントの伝播のスケジュール

1. Web ブラウザを使用して、次の URL に接続します。

```
<webagent>/wf_setup.check_all
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「グローバル・ワークフロー・プリファレンスの設定」を参照してください。

---

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

---

**注意：**「セットアップのチェック」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

セットアップのチェック(WF91JP.Oracle.COM) - Netscape

ファイル(F) 編集(E) 表示(V) ジャンプ(G) Communicator(C) ヘルプ(H)

名前	値	最小推奨値
aq_tm_processes	1	1
job_queue_processes	10	5

データベース・リンク

名前	状態

ローカル・エージェント

エージェント	方向	キュー名	キュー作成済	メッセージの準備完了	メッセージ待機
WF_ERROR	受信	OWF_MGR.WF_ERROR	はい	0	0
WF_IN	受信	OWF_MGR.WF_IN	はい	0	0
WF_OUT	送信	OWF_MGR.WF_OUT	はい	0	0
WF_DEFERRED	受信	OWF_MGR.WF_DEFERRED	はい	0	0
WF_SMTP_O_1_QUEUE	受信	OWF_MGR.WF_SMTP_O_1_QUEUE	はい	150	0

ローカル・インバウンド・エージェントのリスナー

エージェント	スケジュール済	処理
WF_ERROR	いいえ	<a href="#">作成</a>
WF_IN	いいえ	<a href="#">作成</a>
WF_DEFERRED	いいえ	<a href="#">作成</a>

ローカル・アウトバウンド・エージェントの伝播

送信エージェント	接続先	スケジュール
WF_OUT	ローカル	<a href="#">作成</a>

ドキュメント: 完了。

- 「セットアップのチェック」画面に、ローカル・システムの伝播に関する設定およびコンポーネントが表示されます。
- 「ローカル・アウトバウンド・エージェントの伝播」領域には、伝播を必要とするローカル・アウトバウンド・エージェントとデータベース・リンクの組合せの一覧が表示されます。この一覧でローカル・アウトバウンド・エージェントに対応付けられている

データベース・リンクは、定義済のインバウンド・エージェントのアドレスに表示される、リモート・システムへのデータベース・リンクです。各ローカル・アウトバウンド・エージェントは、ローカル・システムのインバウンド・エージェントに伝播するときに、「データベース・リンク」列の「ローカル」接続先としても表示されます。13-24ページの「[エージェント](#)」を参照してください。

伝播をスケジュールするアウトバウンド・エージェントとデータベース・リンクの組合せを選択します。

4. 目的のエージェントおよびデータベース・リンクに対して伝播がスケジュールされていない場合は、「スケジュール」列の「作成」リンクを選択して伝播をスケジュールします。「伝播の編集」画面には、アウトバウンド・エージェントのキューの名前が表示されます。さらに、リモート接続先の場合はデータベース・リンク名、ローカル接続先の場合は「ローカル・システム」が表示されます。



5. 「継続時間」フィールドに、伝播ウィンドウの継続時間を秒単位で入力します。
6. 「実行頻度」フィールドに、間隔を秒単位で入力して伝播ウィンドウの表示頻度を指定します。

---

**注意：** 実行間隔は、伝播ウィンドウの継続時間より長くする必要があります。

---

7. 「待機時間」フィールドに、待機時間を秒単位で入力します。これは、すべてのメッセージが伝播されてから、接続先への新しいメッセージがキュー内で再チェックされるまでの待機時間となります。

この待機時間は、メッセージがエンキューされてから、伝播ウィンドウからメッセージが伝播されるまでの、最大待機時間を表します。エンキュー後にすぐにメッセージを伝播するには、待機時間として 0 を入力します。デフォルトの待機時間は 60 秒です。

---

**注意：** AQ に待機時間を適用するには、データベース初期化パラメータのジョブ・キュー間隔の設定を待機時間の値以下にする必要があります。13-56 ページの「[ビジネス・イベント・システムのセットアップのチェック](#)」を参照してください。

---

8. 「送信」ボタンを選択して伝播スケジュールを保存し、「セットアップのチェック」画面に戻ります。「取消」ボタンを選択し、伝播を保存しないで「セットアップのチェック」画面に戻ることもできます。

### ▶ 伝播の更新または削除

1. 「セットアップのチェック」画面の「ローカル・アウトバウンド・エージェントの伝播」領域を使用して、必要なアウトバウンド・エージェントとデータベース・リンクの組合せを選択します。13-67 ページの「[ローカル・アウトバウンド・エージェントの伝播のスケジュール](#)」を参照してください。
2. 伝播がすでにスケジュールされている場合は、「スケジュール」列にある「編集」リンクを選択すると、「伝播の編集」画面に伝播設定が表示されます。



3. 伝播を更新するには、設定に変更を加え、変更内容を保存します。13-67 ページの「[ローカル・アウトバウンド・エージェントの伝播のスケジュール](#)」を参照してください。

4. 伝播を削除するには、「取消」ボタンを選択します。「取消」ボタンを選択し、伝播を変更または削除しないで「セットアップのチェック」画面に戻ることもできます。

---

**注意：** 伝播を削除するときは、伝播ウィンドウを閉じる必要があります。

---

## イベントの呼出し

イベントの呼出しは、アプリケーションまたはワークフローを介して行う以外に、「イベントの呼出し」Web 画面を使用してテストのために手動で呼び出すことができます。後者の場合は、パラメータを追加することはできません。イベントを呼び出すと、イベント・マネージャは、ソース・タイプを「ローカル」にして、そのイベントの有効なサブスクリプションをローカル・システム単位に検索および実行します。また、ソース・タイプを「ローカル」にして、Any イベントの有効なサブスクリプションをローカル・システム単位に検索および実行します。

### ▶ イベントの呼出し

1. Web ブラウザを使用して、次の URL に接続します。

```
<webagent>/wf_event_html.entereventdetails
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

---

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

---

---

**注意：** 「イベントの呼出し」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---

The screenshot shows a Netscape browser window titled 'イベントの呼出し - Netscape'. The address bar shows a URL starting with 'http://'. The browser's menu bar includes 'ファイル(F)', '編集(E)', '表示(V)', 'ジャンプ(G)', 'Communicator(C)', and 'ヘルプ(H)'. The main content area has a header with a home icon, the title 'イベントの呼出し', a right arrow icon, and a question mark icon. The Oracle logo is in the top right. Below the header, there are three input fields: 'イベント名' (a dropdown menu), 'イベント・キー' (a text field), and 'イベント・データ' (a large text area). At the bottom of the form are two buttons: '送信' (Send) and '取消' (Cancel). The status bar at the bottom of the browser window shows 'ドキュメント: 完了。' (Document: Completed).

2. 「イベントの呼出し」画面が表示されます。
3. 「イベント名」フィールドで、呼び出すイベントを選択します。
4. イベントのインスタンスを一意に識別するイベント・キーを入力します。
5. オプションで、イベントの内容を説明するイベント・データを入力します。

---

**注意：**「イベント・データ」フィールドに入力可能な最大データ長は、32KB です。イベント・データが 32KB を超える場合は、「イベント・データ」フィールドにデータを直接入力せずに、イベント定義にジェネレート関数を割り当ててイベント・データを生成する必要があります。13-6 ページの「[イベントの定義](#)」を参照してください。

---

WF\_EVENT.Raise API を使用してイベントを呼び出すこともできます。この方法を使用すると、最大 4GB のデータを格納する CLOB としてイベント・データを渡すことができます。8-268 ページの「[Raise](#)」を参照してください。

---

6. 「送信」ボタンを選択して、イベントをイベント・マネージャに呼び出します。「取消」ボタンを選択し、イベントを呼び出さないで Oracle Workflow ホーム・ページに戻ることもできます。

「送信」ボタンを選択すると、イベントが呼び出され、確認メッセージにイベント名およびイベント・キーが表示されます。「OK」ボタンを選択して「イベントの呼出し」画面に戻ります。

## システムのサインアップ

システム間でビジネス・イベントを送信するには、イベント・メッセージの受信者として宛先システムをソース・システムにサインアップする必要があります。システムのサインアップとは、宛先システムおよびそのインバウンド・エージェントをソース・システムのイベント・マネージャに定義して、ソース・システムのイベント・メッセージを宛先エージェントに送信できるようにすることです。

通常、宛先システムおよびソース・システムを相互にサインアップして、システム間でメッセージを送受信できるようにする必要があります。

宛先システムをサインアップして、ソース・システムのイベント・メッセージを受信するには、次の手順を実行します。

1. ローカル・システムおよびインバウンド・エージェントの定義を宛先システムから取得します。これらの情報は、システム識別子情報となります。宛先システムの「システム識別子」Web 画面を使用して、システム識別子情報で構成される XML 文書を生成できます。13-74 ページの「[システム識別子情報の取得](#)」を参照してください。

---

**注意：** 宛先システムにインストールされている Oracle Workflow へのアクセス権がない場合は、宛先システムのワークフロー管理者にこの手順の実行を依頼してください。

---

2. 宛先システムの識別子情報をソース・システムのイベント・マネージャに追加します。ソース・システムの「システムのサインアップ」Web 画面を使用して情報を追加するこ

とができます。システムのサインアップ・イベントを XML 文書とともに宛先システムからイベント・データとして呼び出します。ソース・システムでシステムのサインアップ・イベントを呼び出すと、事前定義済のサブスクリプションが Oracle Workflow によって実行され、システム識別子情報がソース・システムのイベント・マネージャに追加されます。13-75 ページの「[システムのサインアップ](#)」を参照してください。

---

**注意：** ソース・システムにインストールされている Oracle Workflow へのアクセス権がない場合は、ソース・システムのワークフロー管理者にこの手順を実行するよう依頼してください。

---

### ▶ システム識別子情報の取得

1. Web ブラウザを使用して、宛先システムとしてサインアップするシステムの次の URL に接続します。

```
<webagent>/wf_event_html.getsystemidentifier
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

---

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

宛先システムにインストールされている Oracle Workflow へのアクセス権がない場合は、宛先システムのワークフロー管理者にこの手順の実行を依頼してください。

---

---

**注意：** 「システム識別子」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---

2. Oracle Workflow によって、ローカル・システムおよびそのインバウンド・エージェントの定義を含む、システム識別子 XML 文書が作成されます。この文書をテキスト・ファイルとして保存します。このシステムをソース・システムにサインアップするときに、保存した文書をコピーすれば、システムのサインアップ・イベントのイベント・データとして入力できます。13-75 ページの「[システムのサインアップ](#)」を参照してください。



## ▶ システムのサインアップ

1. Web ブラウザを使用して、宛先システムをサインアップするソース・システムの次の URL に接続します。

```
<webagent>/wf_event_html.entereventdetails?p_event_name=  
oracle.apps.wf.event.system.signup
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

---

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

ソース・システムにインストールされている Oracle Workflow へのアクセス権がない場合は、ソース・システムのワークフロー管理者にこの手順を実行するよう依頼してください。

---

---

**注意：** 「システムのサインアップ」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---



2. 「システムのサインアップ」画面に、システムのサインアップ・イベントの内部名が表示されます。
3. イベントのインスタンスを一意に識別するイベント・キーを入力します。
4. 宛先システムの識別子情報を含む XML 文書を、「イベント・データ」フィールドにコピーします。13-74 ページの「[システム識別子情報の取得](#)」を参照してください。
5. 「送信」ボタンを選択して、システムのサインアップ・イベントをイベント・マネージャに呼び出します。確認メッセージが表示されます。システムのサインアップ・イベントが呼び出されると、事前定義済みのサブスクリプションが Oracle Workflow によって実行され、システム識別子情報がイベント・データからイベント・マネージャに追加されます。14-12 ページの「[システムのサインアップ・イベント](#)」を参照してください。

「取消」ボタンを選択し、システムのサインアップ・イベントを呼び出さないで Oracle Workflow ホーム・ページに戻ることもできます。

## システムの同期

システムの同期とは、ソース・システムに定義されているすべてのイベント・マネージャ・オブジェクトを、ターゲット・システムにレプリケートすることです。イベント・システムの同期イベントを使用して、ソース・システムおよびターゲット・システムを同期させることができます。

### ► システムの同期

1. ソース・システムおよびターゲット・システムを相互にサインアップします。13-73 ページの「[システムのサインアップ](#)」を参照してください。
2. ソース・システム上で、「ローカル」ソース・タイプを持つ「シード・イベント・グループ」への事前定義済サブスクリプションを、次のように変更します。
  - イベント・メッセージを受信するターゲット・システムに対してインバウンド・エージェントを指定するか、イベント・メッセージをターゲット・システムに送信するワークフロー・プロセスを指定します。

---

**注意：** イベント・メッセージを複数のターゲット・システムに送信する場合は、追加のサブスクリプションを定義するか、イベント・メッセージを複数のシステムに送信するワークフロー・プロセスを指定します。

---

- サブスクリプションを使用可能にします。

---

**注意：** ソース・システムにインストールされている Oracle Workflow へのアクセス権がない場合は、ソース・システムのワークフロー管理者にこの手順を実行するよう依頼してください。

---

3. ターゲット・システム上で、「外部」ソース・タイプを持つ「シード・イベント・グループ」への事前定義済サブスクリプションを、使用可能にします。

---

**注意：** ターゲット・システムにインストールされている Oracle Workflow へのアクセス権がない場合は、ターゲット・システムのワークフロー管理者にこの手順の実行を依頼してください。

---

4. ソース・システム上で、「イベントの呼出し」画面を使用してイベント・システムの同期イベント (oracle.apps.wf.event.all.sync) を呼び出します。一意のイベント・キーを入力し、「イベント・データ」フィールドは空白のままにします。13-71 ページの「[イベントの呼出し](#)」を参照してください。

---

---

**注意：** ソース・システムにインストールされている Oracle Workflow へのアクセス権がない場合は、ソース・システムのワークフロー管理者にこの手順を実行するよう依頼してください。

---

---

ソース・システム上でイベント・システムの同期イベントが呼び出されると、「ローカル」ソース・タイプを持つ「シード・イベント・グループ」へのサブスクリプションがトリガーされます。イベント・マネージャがイベント・メッセージを生成します。このメッセージには、ローカル・システム上のイベント、イベント・グループ、システムおよびサブスクリプションを含む、すべてのイベント・マネージャ・オブジェクトの定義が含まれます。イベント・メッセージは、ターゲット・システム上の特定のインバウンド・エージェント、またはイベント・メッセージをターゲット・システムに送信する特定のワークフロー・プロセスに送信されます。

ターゲット・システムにイベント・システムの同期イベントが着信すると、「外部」ソース・タイプを持つ「シード・イベント・グループ」へのサブスクリプションがトリガーされます。イベント・メッセージのオブジェクト定義がターゲット・システムのイベント・マネージャにロードされます。必要に応じて、新しい定義が作成されたり、既存の定義が更新されます。

## 自動レプリケーション

手順 2 および 3 で事前定義済のサブスクリプションを使用可能にすると、これらのサブスクリプションは、ソース・システムのイベント・マネージャのオブジェクト定義に加えられた変更をレプリケートします。イベント、イベント・グループ・メンバー、システム、エージェントまたはサブスクリプションの作成、更新または削除を行うたびに、対応する事前定義済のイベントが Oracle Workflow によって呼び出されます。これらのイベントにより、ソース・システム上の「シード・イベント・グループ」への「ローカル」サブスクリプションがトリガーされ、オブジェクト定義データがターゲット・システムに送信されます。ターゲット・システム上の「シード・イベント・グループ」への「外部」サブスクリプションは、そのデータを受信して、イベント・マネージャのオブジェクト定義を追加、更新または削除します。

ソース・システムの変更をターゲット・システムに自動レプリケートする処理を停止する場合は、システムの同期の完了後にサブスクリプションを使用不可にするか、変更に対応する事前定義済のイベントを使用不可にします。

## マスター/コピー・システム

必要に応じて、特定のシステムをマスター・システムとして設定できます。このようにすると、マスター・システムのイベント・マネージャのオブジェクト定義を関連付けられたコピー・システムにレプリケートすることができます。ただし、コピー・システムのオブジェクト定義への変更は、マスター・システムにレプリケートされません。マスター/コピーのレプリケーションをセットアップするには、ターゲット・コピー・システムをソース・マスター・システムに同期させる手順を通常どおりに行います。次に、オブジェクト定義がコピー・システムから送信されないようにするために、コピー・システム上の「シード・イベ

ント・グループ」への「ローカル」サブスクリプションを使用不可にします。オブジェクト定義がマスター・システムに着信しないようにするために、マスター・システム上の「シード・イベント・グループ」への「外部」サブスクリプションも使用不可にします。

**関連項目：**

14-2 ページ [「事前定義済ワークフロー・イベント」](#)

14-8 ページ [「イベント・システムの同期イベント」](#)

14-8 ページ [「シード・イベント・グループ」](#)

13-48 ページ [「イベント・サブスクリプションの定義」](#)

## ローカル・キューの確認

「イベント・システム・ローカル・キュー」画面を使用して、ビジネス・イベント・システムで使用されているローカル・キューおよびそれらのキューに現在保持されているメッセージを確認できます。

---

---

**注意：**「セットアップのチェック」画面を使用して、ローカル・キューの設定を確認することもできます。13-56 ページの [「ビジネス・イベント・システムのセットアップのチェック」](#) を参照してください。

---

---

---

---

**注意：** Oracle Applications Manager を実装している環境で Oracle Applications embedded Workflow を使用している場合は、Oracle Workflow Manager を使用してローカル・キューに保持されているメッセージのステータスを確認することができます。詳細は、Oracle Applications Manager のオンライン・ヘルプを参照してください。

---

---

Oracle9i リリース 2 で利用可能なスタンドアロン版の Oracle Workflow を使用している場合は、Oracle Enterprise Manager から利用可能なスタンドアロン版の Oracle Workflow Manager コンポーネントを使用して、ローカル・キューに保持されているメッセージのステータスを確認できます。詳細は、Oracle Workflow Manager のオンライン・ヘルプを参照してください。

---

---

---

---

**注意：**「イベント・システム・ローカル・キュー」画面には、ビジネス・イベント・システムのエージェントの他に、通知メーラー SMTP キュー (WF SMTP\_O\_1\_QUEUE) に定義されている標準エージェントも表示されます。このエージェントの情報を表示して、通知メーラー・キューにある通知メッセージの数をチェックできます。ただし、WF SMTP\_O\_1\_QUEUE エージェントは、ビジネス・イベント・システムでは使用されません。このキューは WF\_EVENT\_T をペイロード・タイプとして使用しないため、キューに関するメッセージの詳細を確認することはできません。13-27 ページの「[標準エージェント](#)」を参照してください。

---

---

➤ **ローカル・キューの確認**

1. Web ブラウザを使用して、次の URL に接続します。

```
<webagent>/wf_event_html.eventqueuedisplay
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL に置き換えてください。2-13 ページの「[グローバル・ワークフロー・ブリファレンスの設定](#)」を参照してください。

---

---

**注意：** この URL はセキュリティ画面にアクセスするため、現行 Web セッションで有効なユーザーとしてログオンしていない場合は、ページが表示される前に有効なユーザーとしてのログオンを求めるプロンプトが表示されます。「イベント・マネージャ」Web 画面にアクセスするには、ワークフロー管理者権限が必要です。

---

---

---

---

**注意：**「イベント・システム・ローカル・キュー」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---

---



- 「イベント・システム・ローカル・キュー」画面に、ビジネス・イベント・システムで使用されているローカル・キューのリストが表示されます。「イベント・システム・ローカル・キュー」画面には、キューに関連付けられているエージェントのプロトコルおよび名前、エージェントが使用されている通信（インバウンドまたはアウトバウンド）、キューに現在保持されているメッセージの数の一覧がキューごとに表示されます。
- キューに格納されているメッセージの詳細のうち、標準 WF\_EVENT\_T データ型をペイロード・タイプとして使用するものを確認できます。キューに格納されているメッセージの詳細を確認するには、そのキューの「詳細の表示」列にある懐中電灯アイコンを選択します。

---

**注意：** 標準 WF\_EVENT\_T データ型をペイロード・タイプとして使用しないキューに関するメッセージの詳細は確認できません。

---



4. 「標準イベント・キュー・メッセージの検索」画面に、選択したキューの名前が表示されます。「標準イベント・キュー・メッセージの検索」画面では、検索基準を入力して特定のイベントを検索できます。検索基準として、次の項目を使用できます。
  - イベント名： 表示するイベント・メッセージの内部イベント名を入力します。
  - イベント・キー： 表示するイベント・メッセージのイベント・キーを入力します。
  - 状態： 表示するイベント・メッセージのステータスとして、「準備完了」、「待機」、「処理済」または「期限満了」を選択します。すべてのステータスのメッセージを表示するときは、「任意」を選択します。
5. 「実行」を選択します。「ローカル・キュー・メッセージ」画面に、選択したキュー内で検索基準と一致したイベント・メッセージの一覧が表示されます。「ローカル・キュー・メッセージ」画面には、イベント名、イベント・キー、相関 ID、イベント・パラメータ、メッセージを送信した「送信元システム」、メッセージを受信した「宛先システム」、送信日、エラー・メッセージ、エラー・スタックおよびメッセージのステータスの一覧が、メッセージごとに表示されます。



ローカル・キュー・メッセージ - Netscape

ファイル(F) 編集(E) 表示(V) ジャンプ(G) Communicator(C) ヘルプ(H)

ローカル・キュー・メッセージ

イベント名	イベント・キー	相関ID	パラメータ	送信元システム	宛先システム	送信日付	エラー・メッセージ	イベント・データ (XML 形式)	イベント・データ (Text 形式)
oracle.apps.ecx.bes.qa.t5	5734			WF_OUT@WF9I.JP.ORACLE.COM	WF_IN@WF9I.JP.ORACLE.COM	26-JUN-2002 11:58:11		準備完了	
oracle.apps.ecx.bes.qa.t5	5734			WF_OUT@WF9I.JP.ORACLE.COM	WF_IN@WF9I.JP.ORACLE.COM	26-JUN-2002 12:11:01		準備完了	
oracle.apps.ecx.bes.qa.t5	5734			WF_OUT@WF9I.JP.ORACLE.COM	WF_IN@WF9I.JP.ORACLE.COM	26-JUN-2002 12:11:18		準備完了	

ドキュメント: 完了。

6. メッセージのイベント・データを XML 文書で確認するには、そのメッセージの「イベント・データ (XML 形式)」列にある懐中電灯アイコンを選択します。
7. メッセージのイベント・データをテキスト文書で確認するには、そのメッセージの「イベント・データ (Text 形式)」列にある懐中電灯アイコンを選択します。

## ワークフロー・エージェントの Ping/ 確認

「ワークフロー・エージェントの Ping/ 確認」を使用して、ビジネス・イベント・システムの設定をテストできます。このワークフローは、Ping イベント・メッセージをローカル・システムまたは外部システム上の各インバウンド・エージェントに送信して、各エージェントからの受信確認イベント・メッセージを待機します。このワークフローが正常に完了すると、これらのエージェントとの通信に関するビジネス・イベント・システムの基本的な設定が完了します。

### 「ワークフロー・エージェントの Ping/ 確認」の機能

「ワークフロー・エージェントの Ping/ 確認」ワークフローを開始するには、「プロセスの開始」Web 画面を使用します。このワークフローは、「マスター Ping」プロセスと「詳細 Ping」プロセスの2つのプロセスで構成されます。すべてのインバウンド・エージェントを ping するには、「マスター Ping」プロセスを選択して、一意の項目キーを入力します。12-2 ページの「[ワークフロー定義のテスト](#)」を参照してください。

「マスター Ping」プロセスを開始すると、ワークフロー・エンジンによって、ローカル・システムまたは外部システム上で定義されているすべてのインバウンド・エージェントが識別され、各エージェントの「詳細 Ping」プロセスが開始されます。マスター・プロセスは、各詳細プロセスの完了を待機します。

「詳細 Ping」プロセスは、マスター・プロセスによって識別されたインバウンド・エージェントにエージェントの Ping イベントが送信されたときに開始されます。詳細プロセスは、エージェントの Ping イベント・メッセージをローカル・システム上のアウトバウンド・エージェントに関連付けられたキューに格納されます。イベント・メッセージには、インバウンド・エージェントが宛先として指定され、所属する詳細プロセスを識別する相関 ID が含まれます。AQ 伝播により、アウトバウンド・キュー内のイベント・メッセージは、指定されたインバウンド・エージェントに関連付けられたキューに送信されます。

受信側システムのインバウンド・エージェントのリスナーは、次回動作したときに、「エージェントの Ping」メッセージをデキューします。イベント・メッセージがデキューされると、「外部」ソース・タイプを持つエージェントの Ping イベントまたは Any イベントへの有効なサブスクリプションが、イベント・マネージャによって検索および実行されます。

エージェントの Ping イベントへの事前定義済みの「外部」サブスクリプションが実行されると、そのルール関数により、受信側システムのアウトバウンド・エージェントに関連付けられているキューに Ping の確認イベントが格納されます。イベント・メッセージには、元のシステムのインバウンド・エージェントが宛先として指定され、エージェントの Ping イベント・メッセージからの相関 ID が含まれます。AQ 伝播により、アウトバウンド・キュー内のイベント・メッセージは、指定されたインバウンド・エージェントに関連付けられているキューに送信されます。

送信元システムのインバウンド・エージェントのリスナーは、次回動作したときに、「Ping の確認」メッセージをデキューします。イベント・メッセージがデキューされると、「外部」ソース・タイプを持つ Ping の確認イベントまたは Any イベントへのアクティブなサブスクリプションが、イベント・マネージャによって検索および実行されます。

Ping の確認イベントへの事前定義済の「外部」サブスクリプションが実行されると、そのルール関数（デフォルトのルール関数）により、イベント・メッセージが「詳細 Ping」プロセスに送信されます。ワークフロー・エンジンは、イベント・メッセージおよび関連 ID に関連付けられた実行中の詳細プロセスを照合します。イベント・メッセージの受信が終了すると、「詳細 Ping」プロセスが完了します。

すべての詳細プロセスが完了すると、マスター・プロセスも完了します。

Workflow Monitor を使用して、「ワークフロー・エージェントの Ping/ 確認」ワークフローの進行状況を確認できます。「イベント・システム・ローカル・キュー」画面を使用して、エージェントの Ping および Ping の確認イベント・メッセージの処理を確認することもできます。11-2 ページの「ワークフロー・モニター」および 13-79 ページの「ローカル・キューの確認」を参照してください。

「ワークフロー・エージェントの Ping/ 確認」ワークフローの完了に必要な時間は、リスナーが動作してインバウンド・エージェントのメッセージをデキューする頻度によって異なります。13-62 ページの「ローカル・インバウンド・エージェントのリスナーのスケジュール」を参照してください。

#### 関連項目：

14-10 ページ「エージェントの Ping イベント」

## 「ワークフロー・エージェントの Ping/ 確認」項目タイプ

「ワークフロー・エージェントの Ping/ 確認」プロセスは、「ワークフロー・エージェントの Ping/ 確認」という項目タイプに関連付けられています。「ワークフロー・エージェントの Ping/ 確認」には現在、「マスター Ping プロセス」と「詳細 Ping プロセス」という 2 つのワークフロー・プロセスが関連付けられています。

「ワークフロー・エージェントの Ping/ 確認」項目タイプの詳細を Workflow Builder に表示するには、「ファイル」メニューから「オープン」を選択します。次に、データベースに接続して「ワークフロー・エージェントの Ping/ 確認」項目タイプを選択するか、`<ORACLE_HOME>/wf/Data/<language>` サブディレクトリにある `wfping.wft` というファイルに接続します。

「ワークフロー・エージェントの Ping/ 確認」のプロパティ画面を見ると、維持タイプが「一時」に、維持日数が 0 になっています。これは、この項目タイプの作業項目に関連付けられているランタイム・データは、終了直後に削除の対象となることを意味します。

「ワークフロー・エージェントの Ping/ 確認」項目タイプには、複数の属性が関連付けられています。これらの属性は、ワークフローのアプリケーション表にある情報を参照します。属性は、イベント・アクティビティと同様に、プロセス全体を通して関数アクティビティによって使用され保存されます。次の表は、「ワークフロー・エージェントの Ping/ 確認」項目タイプ属性を示しています。

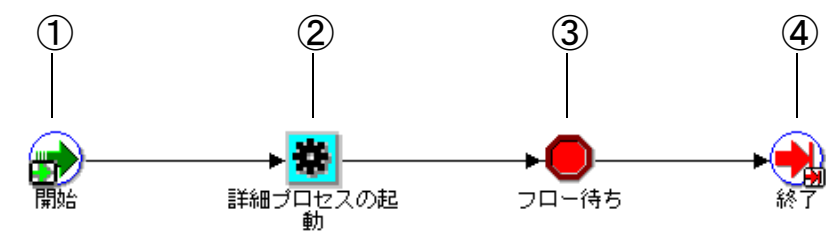
表 13-8

表示名	説明	タイプ	長さ / 書式 / 選択肢タイプ
宛先エージェント	イベント・メッセージを受信するインバウンド・エージェント (<agent>@<system> の書式)	テキスト	
イベント名	イベントの内部名	テキスト	
送信エージェント	イベント・メッセージを送信するアウトバウンド・エージェント (<agent>@<system> の書式)	テキスト	
イベント・キー	イベントの特定のインスタンスを一意に識別するイベント・キー	テキスト	
イベント・メッセージ	イベント・メッセージ	イベント	

「マスター Ping」 プロセスの概要

「マスター Ping」 プロセスのプロパティを表示するには、ナビゲータ・ツリーでプロセスを選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは実行可能です。つまり、最上位レベルのプロセスとして実行できます。

「マスター Ping」 プロセスの「プロセス」ウィンドウを表示すると、プロセスが 4 つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



「プロセスの開始」 Web 画面を使用して「マスター Ping プロセス」を開始すると、「ワークフロー・エージェントの Ping/ 確認」ワークフローが開始されます。オプションで、宛先エージェント、イベント名、送信エージェント、イベント・キーおよびイベント・メッセージを渡すことができます。12-2 ページの「ワークフロー定義のテスト」を参照してください。

ワークフローは、ノード1の「開始」アクティビティから開始します。ノード2で、マスター・プロセスはローカル・システムまたは外部システムに定義されている各インバウンド・エージェントの詳細プロセスを起動します。詳細プロセスは、エージェントの Ping イベントを送信してインバウンド・エージェントを ping し、Ping の確認イベントの形式で受信確認を待機します。

ノード3は「フロー待ち」アクティビティで、すべての詳細プロセスの終了を待機します。すべての詳細プロセスが完了すると、マスター・プロセスが終了します。

## 「マスター Ping」プロセス・アクティビティ

ここでは、プロセスの各アクティビティについて、アクティビティの表示名別に説明します。

### 開始（ノード1）

この「標準」関数アクティビティは、プロセスの開始をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	なし

### 詳細プロセスの起動（ノード2）

この関数アクティビティは、ローカル・システムまたは外部システム上で定義されているすべてのインバウンド・エージェントを識別し、各エージェントの「詳細 Ping」プロセスを起動します。また、エージェントの Ping イベント（oracle.apps.wf.event.test.ping）を「詳細 Ping」プロセスに送信されるように設定します。

関数	WF_EVENT_PING_PKG.LAUNCH_PROCESSES
結果タイプ	なし
前提条件アクティビティ	なし
関数によって設定される項目属性	イベント名、宛先エージェント

### フロー待ち（ノード3）

この「標準」関数アクティビティは、指定されたアクティビティを対応する詳細プロセスが完了するまで、フローを一時停止します。

関数	WF_STANDARD.WAITFORFLOW
結果タイプ	なし
前提条件アクティビティ	詳細プロセスの起動

## 終了（ノード 4）

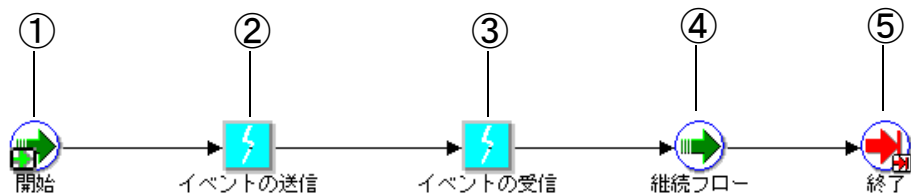
この「標準」関数アクティビティは、プロセスの終了をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	フロー待ち

## 「詳細 Ping」プロセスの概要

「詳細 Ping」プロセスのプロパティを表示するには、ナビゲータ・ツリーでプロセス・アクティビティを選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは実行可能です。つまり、最上位レベルのプロセスとして実行できます。

「詳細 Ping」プロセスの「プロセス」ウィンドウを表示すると、プロセスが 5 つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



「詳細 Ping」プロセスは、「マスター Ping」プロセスによって開始されます。13-86 ページの「[「マスター Ping」プロセスの概要](#)」を参照してください。

ワークフローは、ノード 1 の「開始」アクティビティから開始します。ノード 2 で、「詳細 Ping」プロセスはエージェントの Ping イベントを選択されたインバウンド・エージェントに送信します。ノード 3 で、「詳細 Ping」プロセスはエージェント Ping の確認イベントを待機します。受信確認を受信したマスター・プロセスは、フローを続行できます。「詳細 Ping」プロセスはこの時点で終了します。

## 「詳細 Ping」 プロセス・アクティビティ

ここでは、プロセスの各アクティビティについて、アクティビティの表示名別に説明します。

### 開始（ノード 1）

この「標準」関数アクティビティは、プロセスの開始をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	なし

### イベントの送信（ノード 2）

このイベント・アクティビティは、ローカル・システム上のアウトバウンド・エージェントのエージェントの Ping イベント（oracle.apps.wf.event.test.ping）を、マスター・プロセスによって識別されたインバウンド・エージェントに送信します。イベント・メッセージには、所属する詳細プロセスを識別する関連 ID が含まれます。

イベント・アクション	送信
前提条件アクティビティ	なし
アクティビティによって取得される項目属性	イベント・メッセージ、イベント名、イベント・キー、宛先エージェント

### イベントの受信（ノード 3）

このイベント・アクティビティは、エージェントの Ping イベントを受信したシステムから発信元システムに返された、Ping の確認イベント（oracle.apps.wf.event.test.ack）を受信します。Ping の確認イベント・メッセージには、ワークフロー・エンジンがイベント・メッセージの所属する詳細プロセスを識別するために使用する、関連 ID が含まれます。

イベント・アクション	受信
イベント・フィルタ	oracle.apps.wf.event.test.ack
前提条件アクティビティ	イベントの送信
アクティビティによって設定される項目属性	イベント名、イベント・キー、イベント・メッセージ

**継続フロー（ノード 4）**

この「標準」関数アクティビティは、詳細プロセスの位置をマークします。詳細プロセスが完了すると、対応するマスター・プロセスの停止が解除され、処理を続行します。

<b>関数</b>	WF_STANDARD.CONTINUEFLOW
<b>結果タイプ</b>	なし
<b>前提条件アクティビティ</b>	イベントの受信

**終了（ノード 5）**

この「標準」関数アクティビティは、プロセスの終了をマークします。

<b>関数</b>	WF_STANDARD.NOOP
<b>結果タイプ</b>	なし
<b>前提条件アクティビティ</b>	継続フロー



---

## 事前定義済ワークフロー・イベント

この章では、Oracle Workflow の事前定義済イベントの使用方法について説明します。

## 事前定義済ワークフロー・イベント

Oracle Workflow には、ビジネス・イベント・システム内で関連付けられた状態変化に対して、いくつかの事前定義済イベントが用意されています。これらのイベントへのサブスクリプションを定義して、レプリケーションや検証などを行うことができます。

デフォルトでは、すべての事前定義済イベントが使用可能になっています。これらのイベントの多くは、必要に応じて使用不可にすることができます。

いくつかの事前定義済イベントは、Oracle Workflow のインストール時に自動的に作成されるデフォルトのサブスクリプションによって参照されます。デフォルト・サブスクリプションのサブスクライバは、すべてローカル・システムです。これらのサブスクリプションの多くは、更新したり、使用可能 / 使用不可を切り替えることにより、必要なイベント処理を実行できます。

---

---

**注意：** Unexpected イベントまたはそのイベントへの事前定義済のエラー・サブスクリプションの定義は、変更したり使用不可にしないでください。これらを変更したり使用不可にすると、イベント・マネージャはイベントおよびサブスクリプション処理に対してデフォルトのエラー処理を実行できなくなります。

---

---

---

---

**注意：** 事前定義済イベントおよびデフォルト・サブスクリプションは、Oracle Applications および Oracle Self-Service Web Applications にも用意されています。Oracle Applications に固有のワークフロー・イベントの詳細は、該当する Oracle Applications 製品のドキュメントまたはヘルプを参照してください。

---

---

## イベント定義イベント

### イベント作成

このイベントは、新しい単一イベントまたはイベント・グループ定義が作成されたときに、Oracle Workflow によって呼び出されます。

**内部名** oracle.apps.wf.event.event.create

**ジェネレート関数** wf\_event\_functions\_pkg.generate

### イベント更新

このイベントは、単一イベントまたはイベント・グループ定義が更新されたときに、Oracle Workflow によって呼び出されます。

**内部名** oracle.apps.wf.event.event.update

**ジェネレート関数** wf\_event\_functions\_pkg.generate

### イベント削除

このイベントは、単一イベントまたはイベント・グループ定義が削除されたときに、Oracle Workflow によって呼び出されます。

**内部名** oracle.apps.wf.event.event.delete

**ジェネレート関数** wf\_event\_functions\_pkg.generate

## イベント・グループ定義イベント

### イベント・グループ作成

このイベントは、新しいイベント・グループ・メンバー定義が作成されたときに、Oracle Workflow によって呼び出されます。

**内部名** oracle.apps.wf.event.group.create

**ジェネレート関数** wf\_event\_functions\_pkg.generate

### イベント・グループ更新

このイベントは、イベント・グループ・メンバー定義が更新されたときに、Oracle Workflow によって呼び出されます。

**内部名** oracle.apps.wf.event.group.update

**ジェネレート関数** wf\_event\_functions\_pkg.generate

### イベント・グループ削除

このイベントは、イベント・グループ・メンバー定義が削除されたときに、Oracle Workflow によって呼び出されます。

**内部名** oracle.apps.wf.event.group.delete

**ジェネレート関数** wf\_event\_functions\_pkg.generate

## システム定義イベント

### システム作成

このイベントは、新しいシステム定義が作成されたときに、Oracle Workflow によって呼び出されます。

内部名	oracle.apps.wf.event.system.create
ジェネレート関数	wf_event_functions_pkg.generate

### システム更新

このイベントは、システム定義が更新されたときに、Oracle Workflow によって呼び出されます。

内部名	oracle.apps.wf.event.system.update
ジェネレート関数	wf_event_functions_pkg.generate

### システム削除

このイベントは、システム定義が削除されたときに、Oracle Workflow によって呼び出されます。

内部名	oracle.apps.wf.event.system.delete
ジェネレート関数	wf_event_functions_pkg.generate

## エージェント定義イベント

### エージェント作成

このイベントは、新しいエージェント定義が作成されたときに、Oracle Workflow によって呼び出されます。

内部名	oracle.apps.wf.event.agent.create
ジェネレート関数	wf_event_functions_pkg.generate

### エージェント更新

このイベントは、エージェント定義が更新されたときに、Oracle Workflow によって呼び出されます。

内部名	oracle.apps.wf.event.agent.update
ジェネレート関数	wf_event_functions_pkg.generate

### エージェント削除

このイベントは、エージェント定義が削除されたときに、Oracle Workflow によって呼び出されます。

内部名	oracle.apps.wf.event.agent.delete
ジェネレート関数	wf_event_functions_pkg.generate

## イベント・サブスクリプション定義イベント

### サブスクリプション作成

このイベントは、新しいサブスクリプション定義が作成されたときに、Oracle Workflow によって呼び出されます。

**内部名** oracle.apps.wf.event.subscription.create

**ジェネレート関数** wf\_event\_functions\_pkg.generate

### サブスクリプション更新

このイベントは、サブスクリプション定義が更新されたときに、Oracle Workflow によって呼び出されます。

**内部名** oracle.apps.wf.event.subscription.update

**ジェネレート関数** wf\_event\_functions\_pkg.generate

### サブスクリプション削除

このイベントは、サブスクリプション定義が削除されたときに、Oracle Workflow によって呼び出されます。

**内部名** oracle.apps.wf.event.subscription.delete

**ジェネレート関数** wf\_event\_functions\_pkg.generate

## イベント・システムの同期イベント

このイベントは、ローカル・システム上のイベント・マネージャを別のシステム上のイベント・マネージャに同期させるときに呼び出すことができます。システムの同期イベントのイベント・メッセージには、ローカル・システム上のすべてのイベント・マネージャ・オブジェクトの定義が含まれます。13-77 ページの「[システムの同期](#)」を参照してください。

<b>内部名</b>	oracle.apps.wf.event.all.sync
<b>ジェネレート関数</b>	wf_event_functions_pkg.generate

## シード・イベント・グループ

このイベント・グループは、システム間でビジネス・イベント・システムオブジェクトを自動レプリケートするときに使用されるイベントで構成されます。シード・イベント・グループには、すべてのイベント、イベント・グループ、システム、エージェントおよびサブスクリプション定義イベント以外に、イベント・システムの同期イベントも含まれます。

<b>内部名</b>	oracle.apps.wf.event.group.all
<b>メンバー</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 には、シード・イベント・グループに対して、2 つのデフォルト・サブスクリプションが用意されています。一方のサブスクリプションでは、いずれかのグループ・メンバー・イベントがローカルで呼び出されたときに、イベント・マネージャ・データを



エージェントまたはワークフロー・プロセスに送信することができます。このサブスクリプションを使用するには、データの送信先のエージェントまたはワークフローを追加して、サブスクリプションを使用可能にする必要があります。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-1

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	ローカル
イベント・フィルタ	oracle.apps.wf.event.group.all
ステータス	使用不能
ルール・データ	メッセージ
ルール関数	wf_rule.default_rule
優先度	標準

もう一方のサブスクリプションでは、いずれかのグループ・メンバー・イベントが外部ソースから着信したときに、イベント・マネージャ・データをローカル・システムにロードすることができます。このサブスクリプションを使用するには、サブスクリプションを使用可能にする必要があります。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-2

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	外部
イベント・フィルタ	oracle.apps.wf.event.group.all
ステータス	使用不能
ルール・データ	キー
ルール関数	wf_event_functions_pkg.receive

#### 関連項目：

13-48 ページ [「イベント・サブスクリプションの定義」](#)

13-77 ページ [「システムの同期」](#)

## エージェントの Ping イベント

### エージェントの Ping イベント

このイベントは、インバウンド・エージェントを ping するために、「ワークフロー・エージェントの Ping/ 確認」項目タイプの詳細 Ping プロセスから送信されます。「プロセスの開始」Web 画面を使用してマスター Ping プロセスを開始すると、マスター Ping プロセスによって詳細 Ping プロセスが開始されます。13-84 ページの「ワークフロー・エージェントの Ping/ 確認」を参照してください。

内部名	oracle.apps.wf.event.test.ping
ジェネレート関数	なし

Oracle Workflow には、エージェント・イベントの Ping イベントに対して、1 つのデフォルト・サブスクリプションが用意されています。このサブスクリプションでは、外部ソースからエージェント・イベントの Ping が着信したときに、Ping の確認イベントを発信元のシステムに返送することができます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-3

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	外部
イベント・フィルタ	oracle.apps.wf.event.test.ping
ステータス	使用可能
ルール・データ	キー
ルール関数	wf_event_ping_pkg.acknowledge

### Ping の確認

このサブスクリプションは、Ping の確認イベントが着信したときに、Oracle Workflow から発信元のシステムに返送されます。13-84 ページの「ワークフロー・エージェントの Ping/ 確認」を参照してください。

内部名	oracle.apps.wf.event.test.ack
ジェネレート関数	なし

Oracle Workflow には、Ping の確認イベントに対して、1 つのデフォルト・サブスクリプションが用意されています。このサブスクリプションでは、外部ソースからイベントが着信したときに、Ping の確認イベントを「ワークフロー・エージェントの Ping/ 確認」項目タ

イブの詳細 Ping プロセスに送信することができます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-4

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	外部
イベント・フィルタ	oracle.apps.wf.event.test.ack
ステータス	使用可能
ルール・データ	キー
ルール関数	wf_rule.default_rule
ワークフロー項目タイプ	WFPING
ワークフロー・プロセス名	WFDTLPNG

**関連項目：**

13-48 ページ [「イベント・サブスクリプションの定義」](#)

## システムのサインアップ・イベント

ソース・システムの「システムのサインアップ」Web 画面からこのイベントを呼び出して、ソース・システムからイベント・メッセージを受信する宛先システムをサインアップすることができます。13-73 ページの「[システムのサインアップ](#)」を参照してください。

内部名	oracle.apps.wf.event.system.signup
ジェネレート関数	なし

Oracle Workflow には、システムのサインアップ・イベントに対して、1 つのデフォルト・サブスクリプションが用意されています。このサブスクリプションでは、システムのサインアップ・イベントがローカルで呼び出されたときに、イベント・マネージャ・データがローカル・システムにロードされます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-5

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	ローカル
イベント・フィルタ	oracle.apps.wf.event.system.signup
ステータス	使用可能
ルール・データ	キー
ルール関数	wf_event_functions_pkg.receive

関連項目：  
13-48 ページ「[イベント・サブスクリプションの定義](#)」

# Any イベント

このイベントは、他のイベントがローカルで呼び出されたときまたは外部ソースから着信したときに、暗黙的に呼び出されます。Any イベントへのサブスクリプションを定義すれば、イベントが発生するたびに実行される処理を実装できます。

**注意：** Any イベントの定義を変更したり、使用不可にしないでください。これらを変更したり使用不可にすると、イベント・マネージャはイベントおよびサブスクリプション処理に対してエラー処理を実行できなくなります。

内部名	oracle.apps.wf.event.any
ジェネレート関数	なし

Oracle Workflow には、Any イベントに対して、3つのデフォルト・サブスクリプションが用意されています。最初のサブスクリプションは、イベントがローカルで呼び出されたときにトリガーできます。このサブスクリプションを使用するには、サブスクリプションにアクションを定義して、サブスクリプションを使用可能にする必要があります。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-6

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	ローカル
イベント・フィルタ	oracle.apps.wf.event.any
フェーズ	100
ステータス	使用不能
ルール・データ	キー
優先度	標準

2番目のサブスクリプションは、イベントが外部ソースから着信したときにトリガーできます。このサブスクリプションを使用するには、サブスクリプションにアクションを定義して、サブスクリプションを使用可能にする必要があります。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-7

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	外部
イベント・フィルタ	oracle.apps.wf.event.any
フェーズ	100
ステータス	使用不能
ルール・データ	キー
優先度	標準

3 番目のサブスクリプションでは、イベントが「エラー」ソースから着信したとき（つまり、WF\_ERROR キューからデキューされたとき）に、イベント・メッセージが「システム: エラー」項目タイプのデフォルト・イベント・エラー・プロセスに送信され、例外が呼び出されます。このサブスクリプションを使用するには、サブスクリプションを使用可能にする必要があります。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-8

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	エラー
イベント・フィルタ	oracle.apps.wf.event.any
フェーズ	100
ステータス	使用不能
ルール・データ	キー
ルール関数	wf_rule.error_rule
ワークフロー項目タイプ	WF_ERROR
ワークフロー・プロセス名	DEFAULT_EVENT_ERROR
優先度	標準

関連項目：

13-48 ページ [「イベント・サブスクリプションの定義」](#)

# Unexpected イベント

このイベントへのサブスクリプションは、イベントがローカルで呼び出されたときまたは外部ソースから着信したときに、そのイベントにサブスクリプションが存在しない場合に、Oracle Workflow によって実行されます。

内部名	oracle.apps.wf.event.unexpected
ジェネレート関数	なし

Oracle Workflow には、Unexpected イベントに対して、3 つのデフォルト・サブスクリプションが用意されています。最初のサブスクリプションでは、Unexpected イベントがローカルで呼び出されたときに、「システム:エラー」項目タイプのデフォルト・イベント・エラー・プロセスにイベント・メッセージが送信されます。このサブスクリプションを使用するには、サブスクリプションを使用可能にする必要があります。

**注意：** このサブスクリプションを使用可能にする場合は、ローカル・システム上で呼び出されるすべてのイベントを考慮してからトリガーしてください。呼び出されるローカル・イベントの多くは、このサブスクリプションを必要としません。また、ローカル・システム上で多数のイベントが呼び出される場合は、このサブスクリプションを使用可能にすると、ビジネス・イベント・システムの容量を超えることがあります。

次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-9

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	ローカル
イベント・フィルタ	oracle.apps.wf.event.unexpected
ステータス	使用不能
ルール・データ	キー
ワークフロー項目タイプ	WF_ERROR
ワークフロー・プロセス名	DEFAULT_EVENT_ERROR

2 番目のサブスクリプションでは、Unexpected イベントが外部ソースから着信したときに、イベント・メッセージが「システム:エラー」項目タイプのデフォルト・イベント・エラー・プロセスに送信されます。このサブスクリプションにより、外部システムから受信した予期しないイベント・メッセージをローカル・システムで処理できます。

デフォルト・イベント・エラー・プロセスからシステム管理者にエラーが通知されます。通知を受けたシステム管理者は、イベントへのサブスクリプション処理を再試行または中止することができます。たとえば、システム管理者はイベントを処理するサブスクリプションを定義してから、イベントを再試行することもできます。

Unexpected イベントへの「外部」サブスクリプションは、デフォルトで使用可能になっています。このサブスクリプションは、必要に応じて使用不可にすることができます。

**注意：** このサブスクリプションを使用不可にする場合は、外部ソースから受信した Unexpected イベントの処理に関する結果をすべて考慮してください。このサブスクリプションを使用不可にした場合、これらのイベント・メッセージは、着信したインバウンド・キューに残り、一定期間検出されないことがあります。

次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-10

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	外部
イベント・フィルタ	oracle.apps.wf.event.unexpected
ステータス	使用可能
ルール・データ	キー
ワークフロー項目タイプ	WF_ERROR
ワークフロー・プロセス名	DEFAULT_EVENT_ERROR

3 番目のサブスクリプションでは、Unexpected イベントが「エラー」ソースから着信したとき（つまり、WF\_ERROR キューからデキューされたとき）に、イベント・メッセージが「システム:エラー」項目タイプのデフォルト・イベント・エラー・プロセスに送信されます。このサブスクリプションは、デフォルトで使用可能になっています。

**注意：** Unexpected イベントへの事前定義済の「エラー」サブスクリプションの定義を変更したり、使用不可にしないでください。このサブスクリプションを使用不可にすると、カスタム「エラー」サブスクリプションが定義されていないイベントに対して、イベント・マネージャがエラー処理を実行できなくなります。

次の表は、このサブスクリプションに対して定義されているプロパティを示しています。



表 14-11

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	エラー
イベント・フィルタ	oracle.apps.wf.event.unexpected
ステータス	使用可能
ルール・データ	キー
ルール関数	wf_rule.error_rule
ワークフロー項目タイプ	WF_ERROR
ワークフロー・プロセス名	DEFAULT_EVENT_ERROR
優先度	標準

**関連項目：**13-48 ページ [「イベント・サブスクリプションの定義」](#)

## User Entry Has Changed イベント

このイベントは、変更されたユーザー情報が LDAP から取得されたときに、Workflow LDAP API によって呼び出されます。OID 統合を実装する場合は、Workflow LDAP API を実行して、Workflow ディレクトリ・サービスを Oracle Internet Directory に同期させることができます。変更されたユーザーごとに 1 つのイベントが呼び出されます。2-29 ページの「手順 WF-4 Workflow ディレクトリ・サービスと Oracle Internet Directory の同期」および 8-148 ページの「Workflow LDAP API」を参照してください。

内部名	oracle.apps.wf.public.user.change
ジェネレート関数	wf_entity_mgr.gen_xml_payload

Oracle Workflow には、User Entry Has Changed イベントに対して、2つのデフォルト・サブスクリプションが用意されています。一方のサブスクリプションでは、User Entry Has Changed イベントがローカルで呼び出されたときに、変更されたユーザー・データが Oracle Workflow のスタンドアロン版の WF\_LOCAL\_USERS 表にロードされます。Oracle Workflow のスタンドアロン版を使用しているときに、Oracle Internet Directory との統合を実装する場合は、このサブスクリプションを使用可能にする必要があります。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-12

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	ローカル
イベント・フィルタ	oracle.apps.wf.public.user.change
ステータス	使用不能
ルール・データ	キー
ルール関数	wf_sso.user_change

もう一方のサブスクリプションでは、**User Entry Has Changed** イベントがローカルで呼び出されたときに、変更されたユーザー・データが **Oracle Applications embedded Workflow** の **WF\_LOCAL\_USERS** 表にロードされます。**Oracle Applications embedded Workflow** を使用しているときに、**Oracle Internet Directory** との統合を実装する場合は、このサブスクリプションを使用可能にする必要があります。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-13

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	ローカル
イベント・フィルタ	oracle.apps.wf.public.user.change
ステータス	使用不能
ルール・データ	キー
ルール関数	fnd_user_pkg.user_change

**関連項目：**13-48 ページ [「イベント・サブスクリプションの定義」](#)

## ワークフロー送信プロトコル

ワークフロー送信プロトコル・プロセスは、イベント・メッセージの受信、送信および受信確認の例を示す、サンプル・ワークフロー・プロセスです。必要に応じて、このプロセスをコピーまたはカスタマイズし、組織固有の要件を取り込むことができます。

ワークフロー送信プロトコル・プロセスでは、サブスクリプションからイベント・メッセージを受信して、サブスクリプションに指定されているインバウンド・メッセージにイベント・メッセージを送信し、必要に応じて受信確認を待機します。また、必要に応じて受信確認を送信します。たとえば、あるシステムでこのプロセスを使用して、イベント・メッセージを別のシステムに送信できます。また、受信したシステムで同じプロセスを使用して、受信確認メッセージを送信元のシステムに返送することもできます。

「ワークフロー送信プロトコル」ワークフローは、ワークフロー・イベント・プロトコル・プロセスのみで構成されます。このプロセスは、イベント・サブスクリプションからイベント・メッセージを受信したときに開始されます。

「ワークフロー送信プロトコル」ワークフローは、次の方法で開始できます。

- 「イベントの呼出し」画面から、ワークフロー送信プロトコル・イベントを呼び出します。一意のイベント・キーを入力し、有効な XML 文書をイベント・データとして入力します。事前定義済のサブスクリプションにより、イベント・メッセージはワークフロー・イベント・プロトコル・プロセスに送信されます。13-71 ページの「[イベントの呼出し](#)」および 14-27 ページの「[ワークフロー送信プロトコル・イベント](#)」を参照してください。
- エージェントがワークフロー送信プロトコル・イベントを外部ソースから受信したときに、プロセスを開始することもできます。事前定義済のサブスクリプションにより、イベント・メッセージはワークフロー・イベント・プロトコル・プロセスに送信されます。14-27 ページの「[ワークフロー送信プロトコル・イベント](#)」を参照してください。
- 独自のサブスクリプションを定義して、選択したイベントをワークフロー・イベント・プロトコル・プロセスに送信します。サブスクリプションのワークフローに対して、項目タイプを WFSNDPRT に、プロセス名を WFEVPRTC に指定します。イベントをワークフローに送信するために、デフォルトのルール関数を使用するか、送信処理をカスタム・ルール関数に組み込んでください。ワークフロー・イベント・プロトコル・プロセスは、サブスクリプションが実行され、プロセスがイベント・メッセージを受信したときに開始します。13-48 ページの「[イベント・サブスクリプションの定義](#)」を参照してください。

## 「ワークフロー送信プロトコル」項目タイプ

ワークフロー送信プロトコル・プロセスは、「ワークフロー送信プロトコル」という項目タイプに関連付けられています。現在、「ワークフロー送信プロトコル」には、ワークフロー・イベント・プロトコル・プロセスという 1 つのワークフローが関連付けられています。

「ワークフロー送信プロトコル」項目タイプの詳細を **Workflow Builder** に表示するには、「ファイル」メニューから「オープン」を選択します。データベースに接続して、「ワークフロー送信プロトコル」項目タイプを選択するか、ファイル・システムの `<ORACLE_HOME>/wf/Data/<language>` サブディレクトリにある `wfsndprt.wft` というファイルに接続します。

「ワークフロー送信プロトコル」のプロパティ画面を見ると、維持タイプが「一時」に、維持日数が 0 になっています。これは、この項目タイプの作業項目に関連付けられているランタイム・データは、終了直後に削除の対象となることを意味します。

「ワークフロー送信プロトコル」項目タイプには、複数の属性が関連付けられています。これらの属性は、ワークフローのアプリケーション表にある情報を参照します。属性は、イベント・アクティビティと同様に、プロセス全体を通して関数アクティビティによって使用され保存されます。次の表は、「ワークフロー送信プロトコル」項目タイプの属性を示しています。

表 14-14

表示名	説明	タイプ	長さ / 書式 / 選択肢タイプ
イベント名	イベントの内部名	テキスト	
イベント・キー	イベントの特定のインスタンスを一意に識別するイベント・キー	テキスト	
イベント・メッセージ	イベント・メッセージ	イベント	
宛先エージェント	イベント・メッセージを受信するインバウンド・エージェント ( <code>&lt;agent&gt;@&lt;system&gt;</code> の書式)	テキスト	
送信元エージェント	イベント・メッセージを送信するアウトバウンド・エージェント ( <code>&lt;agent&gt;@&lt;system&gt;</code> の書式)	テキスト	
受信確認が必要ですか？	送信するイベント・メッセージに、受信者からの受信確認が必要かどうかを指定するオプション	テキスト	

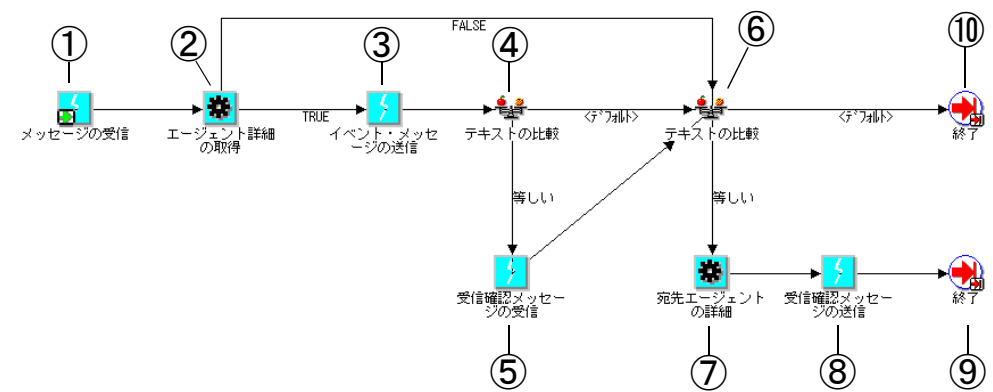
表 14-14（続き）

表示名	説明	タイプ	長さ / 書式 / 選択肢タイプ
受信確認を送信しますか？	受信したメッセージの受信確認を送信するかどうかを指定するオプション	テキスト	
受信確認メッセージ	送信される受信確認メッセージ	イベント	
宛先エージェントの確認	受信確認メッセージを受信するインバウンド・エージェント（書式は <agent>@<system>）	テキスト	
サブスクリプション GUID	サブスクリプションのグローバル一意識別子	テキスト	

ワークフロー・イベント・プロトコル・プロセスの概要

ワークフロー・イベント・プロトコル・プロセスのプロパティを表示するには、ナビゲータ・ツリーでプロセスを選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは実行可能です。つまり、最上位レベルのプロセスとして実行できます。

ワークフロー・イベント・プロトコル・プロセスの「プロセス」ウィンドウを表示すると、このプロセスが 8 個の一意のアクティビティで構成されていることがわかります。そのうちのいくつかが再利用され、ワークフロー・ダイアグラムに表示される 10 個のアクティビティ・ノードを構成しています。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



「ワークフロー送信プロトコル」ワークフローは、イベント・マネージャからワークフロー・イベント・プロトコル・プロセスにイベント・メッセージが送信されたときに開始します。たとえば、ワークフロー送信プロトコル・イベントをローカルで呼び出すか、外部ソースからこのイベントを受信すると、事前定義済のサブスクリプションによって、ワークフロー・イベント・プロトコル・プロセスにイベント・メッセージが送信されます。14-27 ページの「ワークフロー送信プロトコル・イベント」を参照してください。

ワークフローは、ノード 1 の「受信」メッセージ・アクティビティから開始します。ノード 2 で、プロセスはサブスクリプションから目的のアウトバウンド・エージェントおよびインバウンド・エージェントのエージェント詳細を取得します。

インバウンド・エージェントが指定されていない場合、プロセスはノード 6 まで直接進み、受信確認メッセージを送信するかどうかを決定します。

サブスクリプションに「宛先エージェント」が指定されている場合、プロセスはそのエージェントにイベント・メッセージを送信します。プロセスは、サブスクリプション・パラメータに基づいて、イベント・メッセージに受信者からの受信確認が必要かどうかを決定します。受信確認が必要な場合、ワークフロー・エンジンは受信確認メッセージの受信を待機します。受信確認が必要な場合、プロセスはノード 6 まで直接進み、受信確認メッセージを送信するかどうかを決定します。

ノード 6 で、プロセスは受信した元のメッセージの受信確認を送信するかどうかを決定します。受信確認を送信する必要がない場合、この時点でプロセスは終了します。受信確認を送信する必要がある場合は、受信確認の送信先インバウンド・エージェントのエージェント詳細を取得して、受信確認をそのエージェントに送信した後で、プロセスは終了します。

## ワークフロー・イベント・プロトコル・プロセスのアクティビティ

ここでは、プロセスの各アクティビティについて、アクティビティの表示名別に説明します。

### メッセージの受信（ノード 1）

このイベント・アクティビティは、イベント・マネージャからワークフロー・イベント・プロトコル・プロセスに送信されたイベント・メッセージを受信して、新しい項目を開始します。

イベント・アクション	受信
イベント・フィルタ	なし
前提条件アクティビティ	なし
アクティビティによって設定される項目属性	イベント名、イベント・キー、イベント・メッセージ

### エージェント詳細の取得（ノード 2）

この関数アクティビティは、メッセージを送信するアウトバウンド・エージェントおよびメッセージを受信するインバウンド・エージェントのサブスクリプションからエージェント詳細の取得を試みます。インバウンド・エージェントが指定されていない場合、プロセスはノード 6 まで直接進みます。インバウンド・エージェントは指定されているが、アウトバウンド・エージェントが指定されていない場合は、ローカル・システムのデフォルトのアウトバウンド・エージェントが関数によって選択されます。

関数	WF_STANDARD.GETAGENTS
結果タイプ	ブール
前提条件アクティビティ	メッセージの受信
関数によって取得される項目属性	サブスクリプション GUID
関数によって設定される項目属性	宛先エージェント、送信元エージェント

### イベント・メッセージの送信（ノード 3）

このイベント・アクティビティは、イベント・メッセージを、ローカル・システムのアウトバウンド・エージェントから、指定されたインバウンド・エージェントに送信します。

イベント・アクション	送信
前提条件アクティビティ	エージェント詳細の取得
アクティビティによって取得される項目属性	イベント・メッセージ、送信元エージェント、宛先エージェント



## テキストの比較（ノード 4）

この「標準」関数アクティビティは、2つのテキストの値を比較します。このノードでは、プロセスは「受信確認が必要ですか？」項目属性をチェックして、ノード3で送信されるイベント・メッセージに受信者からの受信確認が必要かどうかを判断します。

プロセスを開始したサブスクリプションにパラメータ名と値のペア「ACKREQ=Y」が含まれていた場合、ワークフロー・エンジンはプロセスの開始時に「受信確認が必要ですか？」項目属性をYに設定します。この場合、プロセスはノード4からノード5に進みます。それ以外の場合は、ノード4からノード6に直接進みます。

<b>関数</b>	WF_STANDARD.COMPARE
<b>結果タイプ</b>	比較
<b>前提条件アクティビティ</b>	なし
<b>アクティビティによって取得される項目属性</b>	受信確認が必要ですか？

## 受信確認メッセージの受信（ノード 5）

このイベント・アクティビティは、ワークフロー送信プロトコル受信確認イベントの受信を待機します。つまり、ワークフロー・イベント・プロトコル・プロセスのノード3から送信されたイベント・メッセージに対して、それを受信したシステムが返す受信確認イベントを待機します。

<b>イベント・アクション</b>	受信
<b>イベント・フィルタ</b>	oracle.apps.wf.event.wf.ack
<b>前提条件アクティビティ</b>	テキストの比較

## テキストの比較（ノード 6）

この「標準」関数アクティビティは、2つのテキストの値を比較します。このノードで、プロセスは「受信確認を送信しますか？」項目属性をチェックして、着信した元のメッセージの受信確認を送信するかどうかを判断します。

元のメッセージに受信確認が必要な場合、ワークフロー・エンジンはプロセスの開始時に、「受信確認を送信しますか？」項目属性をYに設定します。この場合、プロセスはノード6からノード7に進みます。それ以外の場合、プロセスはノード10で終了します。

<b>関数</b>	WF_STANDARD.COMPARE
<b>結果タイプ</b>	比較
<b>前提条件アクティビティ</b>	なし

**アクティビティによって取得される項目属性** 受信確認を送信しますか？

**宛先エージェントの詳細（ノード7）**

この関数アクティビティは、受信確認を必要とする送信元システムのインバウンド・エージェントを選択して、そのエージェントのエージェント詳細を取得します。

<b>関数</b>	WF_STANDARD.GETACKAGENT
<b>結果タイプ</b>	なし
<b>前提条件アクティビティ</b>	テキストの比較
<b>関数によって設定される項目属性</b>	宛先エージェントの確認

**受信確認メッセージの送信（ノード8）**

このイベント・アクティビティは、「ワークフロー送信プロトコル受信確認」メッセージを、ローカル・システムのアウトバウンド・エージェントからノード7で識別されたインバウンド・エージェントに送信します。

<b>イベント・アクション</b>	送信
<b>前提条件アクティビティ</b>	宛先エージェント詳細
<b>アクティビティによって取得される項目属性</b>	受信確認メッセージ、イベント・キー、宛先エージェントの確認

**終了（ノード9 および 10）**

この「標準」関数アクティビティは、プロセスの終了をマークします。

<b>関数</b>	WF_STANDARD.NOOP
<b>結果タイプ</b>	なし
<b>前提条件アクティビティ</b>	なし

# ワークフロー送信プロトコル・イベント

## ワークフロー送信プロトコル・イベント

このイベントを「イベントの呼出し」 Web 画面から呼び出すと、「ワークフロー送信プロトコル」項目タイプのワークフロー・イベント・プロトコル・プロセスを使用して、イベントをエージェントに送信できます。このワークフロー・プロセスには、メッセージに受信者からの受信確認が必要かどうか、受信したメッセージの受信確認を送信するかどうかを指定できます。

内部名	oracle.apps.wf.event.wf.send
ジェネレート関数	なし

Oracle Workflow には、ワークフロー送信プロトコル・イベントに対して、2 つのデフォルト・サブスクリプションが用意されています。一方のサブスクリプションでは、ワークフロー送信プロトコル・イベントがローカルで呼び出されたときに、「ワークフロー送信プロトコル」項目タイプのワークフロー・イベント・プロトコル・プロセスにイベント・メッセージが送信されます。サブスクリプション・パラメータに対して、メッセージに受信確認が必要であることを指定します。このサブスクリプションは、デフォルトで使用可能になっています。アウトバウンド・エージェントおよびインバウンド・エージェントをサブスクリプションに追加すると、ワークフロー・イベント・プロトコル・プロセスから送信するイベント・メッセージの送信先を指定できます。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-15

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	ローカル
イベント・フィルタ	oracle.apps.wf.event.wf.send
ステータス	使用可能
ルール・データ	キー
ルール関数	wf_rule.workflow_protocol
ワークフロー項目タイプ	WFSNDPRT
ワークフロー・プロセス名	WFEVPRTC
パラメータ	ACKREQ=Y

もう一方のサブスクリプションでは、「ワークフロー送信プロトコル・イベントが外部ソースから着信したときに、ワークフロー送信プロトコル」項目タイプのワークフロー・イベン

ト・プロトコル・プロセスにイベント・メッセージが送信されます。このサブスクリプションは、デフォルトで使用可能になっています。アウトバウンド・エージェントおよびインバウンド・エージェントをサブスクリプションに追加して、ワークフロー・イベント・プロトコル・プロセスから別のエージェントにイベント・メッセージが送信されるように指定することもできます。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-16

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	外部
イベント・フィルタ	oracle.apps.wf.event.wf.send
ステータス	使用可能
ルール・データ	キー
ルール関数	wf_rule.workflow_protocol
ワークフロー項目タイプ	WFSNDPRT
ワークフロー・プロセス名	WFEVPRTC

ワークフロー送信プロトコル受信確認イベント

このイベントは、ワークフロー・イベント・プロトコル・プロセスに送信されたイベント・メッセージまたはワークフロー・イベント・プロトコル・プロセスから着信したイベント・メッセージに対して受信確認が必要なときに、発信元のシステムに返送されます。

内部名 oracle.apps.wf.event.wf.ack

ジェネレート関数 なし

Oracle Workflow には、ワークフロー送信プロトコル受信確認イベントに対して、1つのデフォルト・サブスクリプションが用意されています。このサブスクリプションでは、ワークフロー送信プロトコル受信確認イベントが外部ソースから着信したときに、「ワークフロー送信プロトコル」項目タイプのワークフロー・イベント・プロトコル・プロセスにイベント・メッセージが送信されます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 14-17

サブスクリプションのプロパティ	値
システム	<local system>

表 14-17 (続き)

サブスクリプションのプロパティ	値
ソース・タイプ	外部
イベント・フィルタ	oracle.apps.wf.event.wf.ack
ステータス	使用可能
ルール・データ	キー
ルール関数	wf_rule.workflow_protocol
ワークフロー項目タイプ	WFSNDPRT
ワークフロー・プロセス名	WFEVPRTC

**関連項目：**

13-48 ページ [「イベント・サブスクリプションの定義」](#)



---

## デモンストレーション用ワークフロー・プロセス

この章では、Oracle Workflow に用意されているデモンストレーション用のワークフロー・プロセスについて説明します。各デモンストレーション用プロセスは、Oracle Workflow の多数の機能を具体的に示すものです。

## サンプル・ワークフロー・プロセス

Oracle Workflow には、次のサンプル・ワークフロー・プロセスが含まれています。各プロセスは、様々な Oracle Workflow 機能の統合例です。これらのプロセスを使用して、Oracle Workflow のインストールをチェックできます。

- 購買申請プロセス： 結果ベースの分岐、並列分岐、サブプロセス、タイムアウト、ループおよび通知での PL/SQL 文書の統合を示します。15-5 ページの「[購買申請プロセス](#)」を参照してください。
- 製品アンケートプロセス： 投票アクティビティの実装、通知での PL/SQL 文書の統合およびマスター / デティール・プロセスの連携を示します。15-33 ページの「[製品アンケート・プロセス](#)」を参照してください。
- ドキュメント・レビュー・プロセス： 文書管理の統合（この機能は今後使用する目的で確保されています）とループを示します。15-48 ページの「[ドキュメント・レビュー・プロセス](#)」を参照してください。
- エラー・チェック・プロセス： Oracle Alert の定期警告機能をワークフロー・プロセスでシミュレーションする方法および標準「待機」アクティビティの使用法を示します。15-53 ページの「[エラー・チェック・プロセス](#)」を参照してください。
- イベント・システム・デモンストレーション・プロセス： 2つのシステム間のビジネス・イベントの送受信と外部 Java 関数アクティビティの使用法の例を示します。15-62 ページの「[イベント・システム・デモンストレーション](#)」を参照してください。

---

**注意：** Oracle Workflow のスタンドアロン版を使用している場合は、Workflow Configuration Assistant により、前述のサンプル・ワークフロー・プロセスがすべてインストールされます。

Oracle Applications embedded Workflow を使用している場合は、AutoUpgrade により、ドキュメント・レビューおよびエラー・チェック・プロセスのみがインストールされます。この 2つのサンプル・ワークフロー・プロセスは、サポートするデータ・モデルの作成を必要としません。

---

これらのサンプル・ワークフローは、「ワークフロー・デモンストレーション」ホーム・ページまたは「プロセスの開始」Web 画面から開始できます。次の URL を使用して、「ワークフロー・デモンストレーション」ホーム・ページにアクセスできます。

```
<webagent>/wf_demo.home
```

---

**注意：**「ワークフロー・デモンストレーション」Web 画面には、Oracle Workflow ホーム・ページからもアクセスできます。9-2 ページの「[Oracle Workflow ホーム・ページへのアクセス](#)」を参照してください。

---



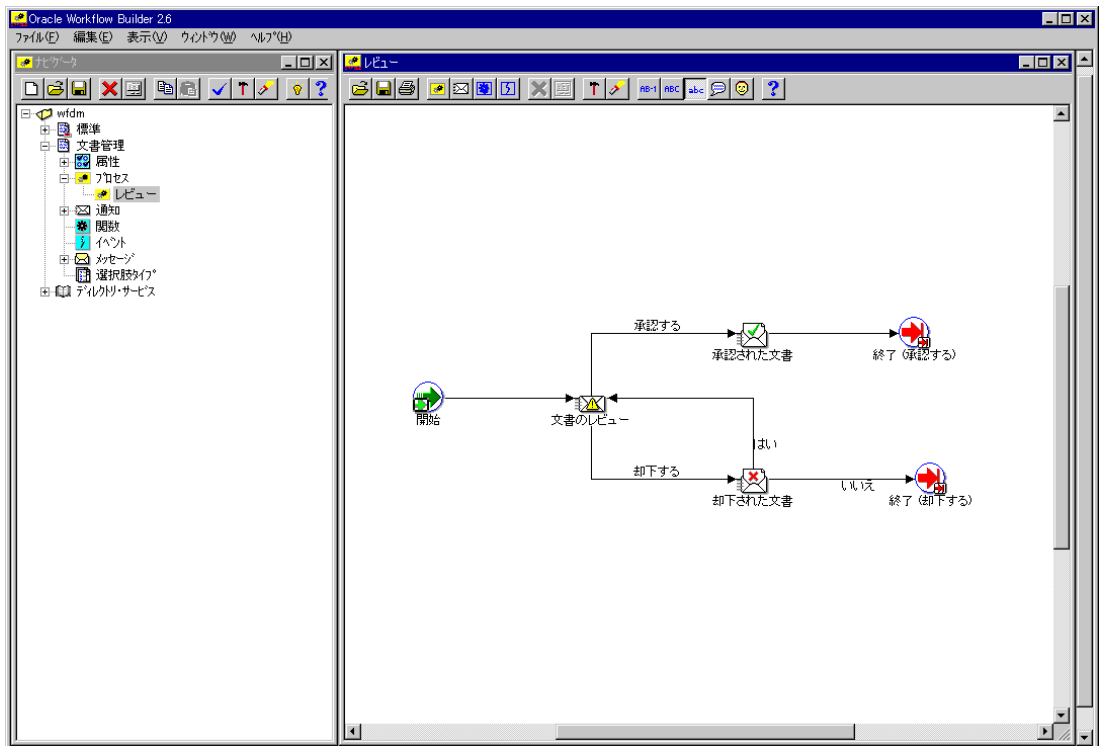
「ワークフロー・デモンストレーション」ホーム・ページでは、右側のフレームに通知のワークリストが表示され、左側のフレームには、他の Web 画面から各サンプル・ワークフローを開始するためのリンクがリストされます。

#### 関連項目：

12-2 ページ「ワークフロー定義のテスト」

## サンプル・ワークフローのプロセス・ダイアグラムの表示

Oracle Workflow Builder の「プロセス」ウィンドウに、サンプル・プロセスのプロセス・ダイアグラムを表示できます。



### ➤ Oracle Workflow Builder でのサンプル・プロセスの表示

1. 「ファイル」メニューから「オープン」を選択します。使用している Oracle Workflow データベースに接続して、必要な項目タイプを選択します。

または、PC 上の Oracle Workflow の wf/data/<language> サブディレクトリにある、どのサンプル・ワークフロー定義ファイルに接続することもできます。

2. データ・ソースを展開し、そのデータ・ソース内の項目タイプのブランチを展開します。
3. 「プロセス」のブランチを展開してプロセス・アクティビティをダブルクリックし、そのプロセスのダイアグラムを「プロセス」ウィンドウに表示します。

## 購買申請プロセス

購買申請プロセスは、品目を購入するための新規購買申請を作成するときに開始されるワークフロー・プロセスの一例です。購買申請プロセスは、承認階層と承認限度情報が格納される2つの表に基づいています。

このデモンストレーションでは、購買申請を発行すると、プロセスによって通知が承認階層の次のマネージャに送信され、購買申請が承認されます。承認担当マネージャの承認限度額が購買申請金額より低ければ、その購買申請を承認できる限度額を持つマネージャが見つかるまで、購買申請は承認階層内で次に上位のマネージャへと順に転送されます。各中間マネージャは、その購買申請を承認して次に上位のマネージャにまわす必要があります。適切な承認限度を持つマネージャが購買申請を承認すると、「承認する」という結果でプロセスが終了します。

次の場合には、プロセスが「却下」という結果で終了することがあります。

- いずれかのマネージャが購買申請を却下した場合
- 購買申請金額が最大の承認限度を超えている場合
- 購買申請者を管理するマネージャがいない場合

Oracle Workflow のスタンドアロン版を使用している場合は、このサンプル・プロセスを設定し、実行できます。Oracle Applications embedded Workflow を使用している場合、このプロセスは、主にデモンストレーションではなく説明のための一例と考えてください。

Oracle Applications embedded Workflow には、このデモンストレーションの設定と実行に必要なファイルが用意されていません。

---

**注意：** Oracle Applications または Oracle Self-Service Web Applications と統合された実行可能ワークフロー・プロセスの詳細は、該当する Oracle Applications のユーザーズ・ガイドまたはオンライン・マニュアルを参照してください。B-2 ページの「[Oracle E-Business Suite 埋込みの事前定義済のワークフロー](#)」を参照してください。

---

---

**注意：** Oracle Self-Service Web Applications では、ここで説明するサンプル・プロセスとは異なるバージョンの購買申請プロセスが事前定義されています。この項で説明するサンプル・プロセスは、あくまでもデモンストレーション用であり、実作業用ではありません。

---

このサンプル・ワークフローは、デモンストレーション・データ・モデルに基づいています。このデータ・モデルには、データの入った2つの表が組み込まれています。一方の表では従業員の承認階層を管理し、他方では各従業員の費用限度額を管理します。この2つの表が、購買申請の承認に使用するデータベース・アプリケーションを構成しています。また、

このデータ・モデルには、このサンプル実装に使用されている Oracle Workflow ユーザーとロールを識別するディレクトリ・サービスも組み込まれています。

架空の購買申請に基づいて購買申請プロセスを実行するには、2つの方法があります。つまり、スクリプトを実行する方法と、Web ベースのインタフェースを使用して購買申請を発行する方法です。どちらの方法でも、購買申請を作成した従業員の氏名、購買申請金額、購買申請番号、購買申請摘要、購買申請プロセス所有者および開始するワークフロー・プロセスの名称を指定する必要があります。

この項では、購買申請プロセスの詳細を説明し、このワークフローの各アクティビティが何を達成するのか理解できるようにします。

## 購買申請データ・モデルのインストール

購買申請データ・モデルは、Oracle Workflow のスタンドアロン版でのみインストールされます。データ・モデルは、Workflow Configuration Assistant によって自動的にインストールされます。インストールで使用されるファイルは、Oracle Workflow Server のディレクトリ構造の demo および demo//<language> サブディレクトリにコピーされます。

---

---

**注意：** 購買申請プロセスのデモンストレーションを正常に機能させるには、インストール後に、Oracle Workflow の必須設定手順を実行する必要があります。2-6 ページの「[設定の概要](#)」を参照してください。

---

---

インストール時には、次の処理が行われます。

- スクリプト `wfdemou.sql` がコールされ、後述のシード・データ表にリストされたユーザーごとに、データベース・アカウントが作成されます。このスクリプトは、パブリック権限とシノニムを作成し、これらのアカウントが Oracle Workflow の Web ベースのユーザー・インタフェースにフルアクセスできるようにします。

---

---

**注意：** セキュリティ上の理由から、インストール・プロセスでは、これらのユーザー・アカウントは作成後に自動的にロックされます。アカウントを使用する前に、`wfdemoul.sql` というスクリプトを使用して、これらのアカウントをロック解除する必要があります。このスクリプトは、Oracle ホームの `wf/demo` サブディレクトリにあります。SQL\*Plus を使用して SYSTEM データベース・アカウントに接続し、次のコマンドを使用してスクリプトを実行します。

---

---

```
sqlplus SYSTEM/<SYSTEM pwd> @wfdemoul
```

SYSTEM アカウントおよびパスワードの詳細は、Oracle データベース管理者に確認してください。

- スクリプト `wfdemoc.sql` がコールされ、シード・データで 2 つの表が作成されます。これらの表は、ワークフローを使用可能なデモンストレーション・データベース・アプリケーションを構成します。
  - `WF_REQDEMO_EMP_HIERARCHY`: 従業員の承認階層が保存されます。承認チェーンは、これらの従業員のユーザー ID で構成されます。BLEWIS、KWALKER、CDOUGLAS および SPIERSON というように、各従業員は昇順で表示され、最大の権限を持つ従業員が最後に表示されます。
  - `WF_REQDEMO_EMP_AUTHORITY`: 各従業員の支払限度額が保存されます。各従業員の限度額は、BLEWIS:500、KWALKER:1000、CDOUGLAS:2000、SPIERSON:3000 というように、従業員のユーザー ID の後に表示されます。
- また、スクリプト `wfdemoc.sql` では、シード・データも `WF_LOCAL_USERS`、`WF_LOCAL_ROLES` および `WF_LOCAL_USER_ROLES` の各表に挿入されます。次の表は、スクリプトによってシードされるユーザーおよびロールを示しています。

表 15-1

ユーザー	ADMIN ロール	MANAGERS ロール	WORKERS ロール	OTHERS ロール
SYSADMIN	yes			
WFADMIN	yes			
BLEWIS			yes	
KWALKER			yes	
CDOUGLAS			yes	yes
SPIERSON		yes		yes

**注意：** 各ユーザーは、電子メール・アドレス `WFINVALID` を持ち、各ロールはロール名と同じ電子メール・アドレスを持ちます。Directory Service API の `SetAdHocUserAttr` または `SetAdHocRoleAttr` をコールすると、ユーザーおよびロールの電子メール・アドレスを他の値に変更できます。また、すべてのユーザーおよびロールへの電子メール通知が 1 つの電子メール受信ボックスに入るようにする場合は、通知メーラーの構成ファイルに、テスト用電子メール・アドレスを指定します。2-55 ページの「[通知メーラーの構成ファイルの作成](#)」を参照してください。

---

---

**注意：** BLEWIS 以外のユーザーは、すべて通知環境設定 MAILHTML を持ち、これによって、「通知」Web 画面から通知を参照する他に、電子メールを介して通知を個々に受け取ることができます。BLEWIS は通知環境設定 SUMMARY を持ち、「通知」Web 画面から通知を参照する以外に、現在オープンしている全通知の要約を、定期的に電子メールで受け取ることができます。通知メーラーを、電子メールの通知を配信するように設定する必要があるため注意してください。

---

---

---

---

**注意：** 購買申請データ・モデルのユーザーとロールを含めるには、Oracle Workflow のディレクトリ・サービス・ビューを WF\_LOCAL\_USERS、WF\_LOCAL\_ROLES および WF\_LOCAL\_USER\_ROLES の各表にマップする必要があります。2-20 ページの「[手順 WF-3 Oracle Workflow のディレクトリ・サービスの設定](#)」を参照してください。

---

---

- スクリプト wfdemos.sql および wfdemob.sql がコールされ、WF\_REQDEMO および WF\_DEMO と呼ばれるパッケージの PL/SQL の仕様と本文が作成されます。これらのパッケージには、次の内容が含まれます。
  - デモンストレーション・ホーム・ページに関連した PL/SQL ストアド・プロシージャ
  - 購買申請プロセスのワークフローで使用される関数アクティビティでコールされる PL/SQL ストアド・プロシージャ
  - 購買申請プロセスのデモンストレーション用に Web ベースのインタフェース・ページを生成するために、Oracle Workflow の Web Agent によってコールされる PL/SQL プロシージャの WF\_REQDEMO.Create\_Req
- ワークフロー・リソース・ジェネレータ・プログラムが実行され、wfdemo.msg からデータベースにメッセージがロードされます。メッセージは、購買申請プロセスのデモンストレーション用に、Web ベースのインタフェース・ページによって使用されます。
- 購買申請プロセスのワークフロー定義が、wfdemo.wft からデータベースにロードされます。このプロセスは、Oracle Workflow Builder で表示できます。

## 購買申請ワークフローの開始

購買申請ワークフローを開始するには、次のいずれかの方法を使用できます。

- スクリプト gwfrund.sql を実行します。
- 「ワークフロー・デモンストレーション」ホーム・ページから、「購買申請デモンストレーション」Web 画面にアクセスします。

- 「プロセスの開始」 Web 画面を使用します。12-2 ページの「[ワークフロー定義のテスト](#)」を参照してください。

また、ユーザーが、購買申請プロセスのワークフローを自動的に開始する購買申請を作成できるように、カスタムのエンド・ユーザー・アプリケーション・インタフェースを作成することもできます。ただし、ユーザーが購買申請をアプリケーション・データベースに保存するときに、購買申請プロセスを開始する WF\_REQDEMO.StartProcess と同じような PL/SQL ストアド・プロシージャをアプリケーションがコールするように、アプリケーション・インタフェースをカスタマイズする必要があります。15-23 ページの「[StartProcess 関数の例](#)」を参照してください。

## ➤ wfrund.sql の実行

1. 次のコマンドを入力し、スクリプト wfrund.sql を SQL\*Plus で実行します。

```
sqlplus <username>/<password>@<alias> @wfrund.sql
<req_num> <req_desc> <req_amount> <requestor>
<req_process_owner> <process_int_name> <item_type>
```

<username>/<password>@<alias> を、ユーザー名、パスワードおよびデモンストレーション・データ・モデルをインストールしたデータベース・アカウントの別名に置き換えます。

<req\_num> を、特定の購買申請を識別する購買申請番号に置き換えます。

<req\_desc> を、特定の購買申請を識別するエンド・ユーザー定義の摘要に置き換えます。

<req\_amount> を購買申請金額に、<requestor> を購買申請作成者（従業員の承認階層に表示される従業員）の氏名に、<req\_process\_owner> を購買申請プロセスの所有者（従業員の承認階層に表示される従業員）の氏名に、<process\_int\_name> をプロセス・アクティビティの内部名（この場合は REQUISITION\_APPROVAL）に、<item\_type> をワークフロー・プロセスが関連付けられている項目タイプの内部名に置き換えます。

2. このスクリプトの完了後に、SQL> プロンプトから Commit と入力し、トランザクションを保存して、SQL\*Plus を終了します。
3. 承認階層に基づいて、購買申請作成者または作成者のマネージャとしてログオンし、プロセスを完了まで進める一連の通知メッセージに従い、応答できます。10-2 ページの「[電子メールによる通知の閲覧](#)」および 10-13 ページの「[Web ブラウザによる通知の表示](#)」を参照してください。

また、ワークフロー・モニターを使用して、ワークフロー・プロセスのステータスを表示することもできます。11-10 ページの「[プロセス検索 Web 画面の使用](#)」を参照してください。

➤ 「購買承認申請」 Web 画面の使用

1. Web ブラウザに次の URL を入力して「ワークフロー・デモンストレーション」 Web 画面にアクセスします。次に、「購買承認申請」リンクをクリックして、「購買承認申請」 Web 画面を表示します。

`<webagent>/wf_demo.home`

`<webagent>` は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

あるいは、次の URL を入力して「購買承認申請」 Web 画面を直接表示できます。

`<webagent>/wf_reqdemo.create_req`

---

---

**注意：** どちらのページにもセキュリティが適用されるため、現行 Web セッションで有効なワークフロー・ユーザーとしてログオンしていない場合は、ページが表示される前に有効なワークフロー・ユーザーとしてのログオンを求めるプロンプトが表示されます。

---

---



Workflow購買申請承認デモンストレーション - Netscape

ファイル(F) 編集(E) 表示(V) ジャンプ(G) Communicator(C) ヘルプ(H)

購買承認申請

注意: このデモは、Oracle Web Employees Enter Requisitionsとは関係ありません。Oracle Workflow単体のデモです。

購買申請

番号

説明

金額

申請者 BLEWIS

プロセス所有者 BLEWIS

項目タイプ 購買申請

送信 取消

**承認階層と費用の権限**

従業員	費用限度額	マネージャ
BLEWIS	500	KWALKER
KWALKER	1000	CDOUGLAS
CDOUGLAS	2000	SPIERSON
SPIERSON	3000	

**説明**

このデモンストレーションでは、購買申請を発行すると、プロセスによって通知が承認階層の次のマネージャに送信され、購買申請が承認されます。承認担当マネージャの費用限度額が購買申請金額より低ければ、その申請を承認できる限度額を持つマネージャが見つかるまで、購買申請は承認階層内で次に上位のマネージャへと順に転送されます。各中間マネージャは、その購買申請を承認して次に上位のマネージャにまわす必要があります。適切な費用限度額を持つマネージャが購買申請を承認すると、プロセスは終了して「承認」という結果が戻されます。いずれかのマネージャが購買申請を却下した場合、購買申請金額が費用限度の最高額を上回っている場合、または購買申請者にマネージャがない場合には、このプロセスが「却下」で終わることもあります。

ドキュメント: 完了。

2. 一意の購買申請番号を入力します。
3. 特定の購買申請摘要を半角英数字 80 文字以内で指定します。
4. 購買申請金額を入力します。この金額は書式なしの数値として入力する必要があります。
5. ドロップ・ダウン・フィールドを使用して、購買申請者とプロセス所有者を指定します。これらのドロップ・ダウン・リストに表示される名前は、デモンストレーション・データ・モデル内のロール名に限定されています。
6. 「購買申請」の入力フィールドの次に、「承認階層と費用の権限」の表、および購買申請デモンストレーション・プロセスがどのように動作するか説明があります。「承認階層と費用の権限」の表は、デモンストレーション・データ・モデルの中身の要約です。
7. 「送信」を選択して購買申請プロセスを開始し、「購買申請が作成されました」確認ページにナビゲートします。



8. 確認ページには、どのロールでログインすると、プロセスの通知を参照できるかという情報の他に、ワークフロー・モニターへの HTML リンクがあり、ここで、「ダイアグラム表示」を選択すると、ADMIN モードで送信した購買申請のプロセス・ダイアグラムが表示されます。11-2 ページの「ワークフロー・モニター」を参照してください。
9. 「プロセスのタイムアウト回数」HTML リンクを選択して、バックグラウンド・エンジンでタイムアウト通知を検索し、タイムアウト時に実行される予定の次のアクティビティを実行します。

このリンクの下には、プロセスにタイムアウトが発生する時期を示す 2 つのメッセージが表示されます。
10. 別の購買申請を「購買申請デモンストレーション」Web 画面に入力して送信する場合は、「購買申請の作成」HTML リンクを選択します。

## 「購買申請」項目タイプ

購買申請プロセスには、「購買申請」と呼ばれる項目タイプが関連付けられています。購買申請に関連付けられているワークフロー・プロセスは、現在 2 つあります。「購買承認申請」と「承認者に通知」です。

購買申請のプロパティ画面を見ると、維持タイプが「一時」に、維持日数が 0 になっています。これは、この項目タイプの作業項目に関連付けられているランタイム・データは、終了直後に削除の対象となることを意味します。また、WF\_REQDEMO.SELECTOR というセレクト関数がコールされることもわかります。このセレクト関数は、指定した項目タイプに複数のプロセスが存在する場合に、実行するプロセス名を戻す PL/SQL ストアド・プロシージャのサンプルです。このセレクト関数の例では、実行するプロセスとして、REQUISITION\_APPROVAL または「Requisition Approval」を返します。

「購買申請」項目タイプには、複数の属性も関連付けられています。これらの属性は、デモンストレーション・アプリケーションの表にある情報を参照します。属性は、通知アクティビティと同様に、プロセス全体を通して関数アクティビティによって使用され保存されます。次の表は、「購買申請」項目タイプの属性を示しています。

表 15-2

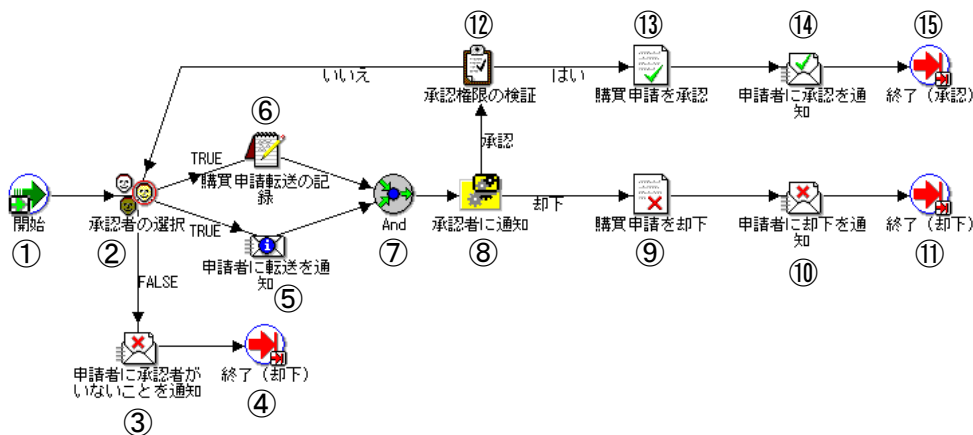
表示名	説明	タイプ	長さ / 書式 / 選択肢タイプ
転送元ユーザー名	購買申請の転送元のユーザー名	ロール	
転送先ユーザー名	購買申請の転送先のユーザー名	ロール	
申請者のユーザー名	購買申請作成者のユーザー名	ロール	
購買申請金額	購買申請金額	数値	9,999,999,999.99
購買申請番号	購買申請の一意の識別子	テキスト	
購買申請摘要	購買申請の一意のユーザー識別子	テキスト	30
購買申請プロセス所有者	購買申請所有者のユーザー名	ロール	
ノート	ノート	テキスト	
モニター URL	モニター URL	URL	
購買申請文書	PL/SQL により作成される購買申請文書	ドキュメント	
購買申請の督促文書	PL/SQL により作成される購買申請の督促文書	ドキュメント	

## 購買承認申請プロセスの概要

購買承認申請プロセスのプロパティを表示するには、ナビゲータ・ツリーでプロセスを選択し、「編集」メニューから「プロパティ」を選択します。購買申請プロセスには、プロセスの完了時に結果が「承認」になるか「却下」になるか（標準項目タイプに関連付けられた「承認」選択肢タイプの選択肢コード）を示す「承認」という結果タイプがあります。このプロセス・アクティビティも実行可能です。つまり、最上位レベルのプロセスとして開始できます。

プロセス・アクティビティの「詳細」プロパティ画面では、購買申請にエラー・プロセスが割り当てられており、プロセスでエラーが起きた場合にのみ起動されることが示されます。このエラー・プロセスには **WFERROR** という項目タイプが関連付けられており、**DEFAULT\_ERROR** と呼ばれます。たとえば、承認階層にリストされていない従業員によって作成された購買申請で、購買承認申請プロセスを起動すると、「承認者の選択」アクティビティの実行を試みたときに、ワークフロー・エンジンはエラーを起こします。このエラーにより、デフォルト・エラー・プロセスである **WFERROR/DEFAULT\_ERROR** が実行されます。6-23 ページの「デフォルト・エラー・プロセス」を参照してください。

購買承認申請プロセスの「プロセス」ウィンドウを表示すると、このプロセスが 12 個の一意のアクティビティで構成されていることがわかります。そのうちのいくつかが再利用され、ワークフロー・ダイアグラムに表示される 15 個のアクティビティ・ノードを構成しています。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



スクリプト **wfrund.sql** を実行するか、「購買申請デモンストレーション」Web 画面を使用して購買申請を発行すると、購買申請ワークフローが開始されます。どちらの場合も、購買申請者、購買申請番号、購買申請金額、購買申請摘要およびプロセス所有者を指定する必要があります。15-8 ページの「購買申請ワークフローの開始」を参照してください。

ワークフローは、ノード1の「開始」アクティビティから開始します。

ノード2では、プロセスは購買申請の承認者を選択しようとします。購買申請の承認者が見つからなければ、申請者に通知され、プロセスは「却下」の最終プロセス結果で終了します。承認者が見つかり、その承認者が誰なのか申請者に通知され、購買却下が承認者に転送されていることがアプリケーション内で関数により記録されます。この2つのアクティビティは、どちらもノード8で実際に承認者に通知される前に完了する必要があります。

ノード8は、指定された期間中に購買申請を承認するように、承認者に要求するサブプロセスです。その時刻までに承認者が応答しなければ、サブプロセスによって「タイムアウト」アクティビティが実行され、承認者が応答するまで督促通知が送信され続けます。承認者が購買申請を却下すると、購買申請はノード9で却下されたものとして更新され、ノード10で申請者に通知されます。プロセスはこの時点で、「却下」という結果で終了します。

承認者が購買申請を承認すると、プロセスはノード12に移動し、購買申請金額が承認者の支払限度額の範囲内かどうか検証されます。範囲内であれば、購買申請はノード13で承認され、ノード14で申請者に通知されます。この場合、プロセスは「承認」という結果で終了します。

## 購買申請プロセス・アクティビティ

ここでは、各アクティビティの説明をアクティビティの表示名ごとに示します。アクティビティのコンポーネントは、関数アクティビティによってコールされる PL/SQL ストアド・プロシージャ以外すべて、グラフィカルな Oracle Workflow Builder で作成できます。関数アクティビティは、Oracle Advanced Queuing を統合することで、データベース外の関数を実行できます。また、Oracle RDBMS で作成し、格納した PL/SQL ストアド・プロシージャも実行できます。購買申請プロセスの関数アクティビティはすべて、PL/SQL ストアド・プロシージャを実行します。購買申請プロセスで使用される PL/SQL ストアド・プロシージャのネーミング規則は、次のとおりです。

WF\_REQDEMO.<PROCEDURE>

WF\_REQDEMO は、購買申請プロセスに使用されるすべてのプロシージャがグループ化されたパッケージの名前です。<PROCEDURE> は、プロシージャ名です。

いくつかのアクティビティについてはより詳細に説明し、その設計根拠についても説明します。15-26 ページの「[関数アクティビティの例](#)」および 15-30 ページの「[通知アクティビティの例](#)」を参照してください。

### 開始（ノード1）

これは、単にプロセスの開始をマークする「標準」関数アクティビティです。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	なし

## 承認者の選択（ノード 2）

この関数アクティビティでは、架空の従業員の承認階層表がチェックされ、購買申請の次の承認者が決定されます。また、このアクティビティでは、前の承認者の名前か、その購買申請がまだ一度も承認されていない場合は作成者の名前が保存されます。承認者が見つかり、このプロシージャでは TRUE を表す値「T」が戻され、それ以外の場合は FALSE を表す値「F」が戻されます。

関数 WF\_REQDEMO.SelectApprover

結果タイプ ブール

前提条件アクティビティ なし

## 申請者に承認者がいないことを通知（ノード 3）

このアクティビティでは、購買申請に該当する承認者が見つからなかったことが、購買申請作成者に通知されます。メッセージには、購買申請番号、購買申請摘要、購買申請金額および最終承認者がいればその氏名を表示する「送信」属性が含まれます。

このアクティビティは、プロセス・ノード 3 で発生します。ノードのプロパティ画面を表示すると、「申請者のユーザー名」という項目タイプ属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。

メッセージ 購買承認申請者が見つからない

結果タイプ なし

前提条件アクティビティ 承認者の選択

## 申請者に転送を通知（ノード 5）

このアクティビティでは、購買申請が承認のために転送されたことが、購買申請作成者に通知されます。メッセージには、購買申請番号、購買申請摘要、購買申請金額、購買申請の転送先となる承認者の氏名、直前の承認者がいればその氏名、および購買申請に追加された最新の注釈などを表示する「送信」属性が含まれます。

ノードのプロパティ画面を表示すると、「申請者のユーザー名」という項目タイプ属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。

メッセージ 購買申請の転送

結果タイプ なし

前提条件アクティビティ 承認者の選択

## 購買申請転送の記録（ノード 6）

現在、このアクティビティでは何も処理されませんが、このワークフローを統合する購買 / 購買申請アプリケーションがある場合は、このアクティビティをカスタマイズし、PL/SQL ストアド・プロシージャを実行して購買 / 購買申請アプリケーション表を更新し、購買申請が次の承認者に転送されたことを示すようにできます。

関数 WF\_REQDEMO.Forward\_Req

結果タイプ なし

前提条件アクティビティ 承認者の選択

## And（ノード 7）

この「標準」関数アクティビティでは、フロー内の並行する 2 つ以上の分岐のすべてのアクティビティが完了した場合にのみ、分岐がマージされます。

関数 WF\_STANDARD.ANDJOIN

結果タイプ なし

前提条件アクティビティ このアクティビティに進むアクティビティが、少なくとも 2 つは必要です。

## 承認者に通知（ノード 8）

このアクティビティは、購買申請を承認または却下する処理を実行する必要があることを、承認者に通知するサブプロセスです。このサブプロセスを表示するには、ナビゲータ・ツリーの「プロセス」のブランチの下にある「承認者に通知」をダブルクリックします。このサブプロセスはその承認者に通知を送信し、承認者が指定された時間内に応答しない場合は、その承認者に処理を実行させるための別の督促通知を送信します。15-20 ページの「[承認者に通知](#)」サブプロセスの概要を参照してください。

結果タイプ 承認

前提条件アクティビティ 承認者の選択

## 購買申請を却下（ノード 9）

現在、このアクティビティでは何も処理されませんが、このワークフローを統合する購買 / 購買申請アプリケーションがある場合は、このアクティビティをカスタマイズし、PL/SQL ストアド・プロシージャを実行して購買 / 購買申請アプリケーション表を更新し、購買申請が却下されたことを示すようにできます。

関数 WF\_REQDEMO.Reject\_Req

結果タイプ なし

前提条件アクティビティ 承認者の選択、承認者に通知

### 申請者に却下を通知（ノード 10）

このアクティビティでは、購買申請が却下されたことが、購買申請作成者に通知されます。メッセージには、購買申請番号、購買申請摘要、購買申請金額、購買申請を却下したマネージャの氏名、およびそのマネージャからの注釈などを表示する「送信」属性が含まれます。

このアクティビティ・ノードのプロパティ画面を表示すると、「申請者のユーザー名」という項目タイプ属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。

メッセージ 購買申請が否認された

結果タイプ なし

前提条件アクティビティ 承認者に通知

### 承認権限の検証（ノード 12）

この関数アクティビティでは、現行の承認者にその購買申請の承認権限があるかどうかを検証されます。このプロセスでは、購買申請金額が承認者の承認限度額と比較され、Yes の場合は値 Y が、No の場合は値 N が戻されます。ビジネス・ルールで承認者による承認額が制限されない場合は、このアクティビティを削除してプロセスをカスタマイズできます。

関数 WF\_REQDEMO.VerifyAuthority

結果タイプ はい / いいえ

前提条件アクティビティ 承認者の選択、承認者に通知

### 購買申請を承認（ノード 13）

現在、このアクティビティでは何も処理されませんが、このワークフローを統合する購買 / 購買申請アプリケーションがある場合は、このアクティビティをカスタマイズし、PL/SQL ストアド・プロシージャを実行して購買 / 購買申請アプリケーション表を更新し、購買申請が承認されたことを示すようにできます。

関数 WF\_REQDEMO.Approve\_Req

結果タイプ なし

前提条件アクティビティ 承認者の選択、承認者に通知、承認権限の検証

### 申請者に承認を通知（ノード 14）

このアクティビティでは、購買申請が承認されたことが、購買申請作成者に通知されます。メッセージには、購買申請番号、購買申請摘要、購買申請金額、承認者名およびその承認者からの注釈を表示する「送信」属性が含まれます。



このアクティビティ・ノードのプロパティ画面を表示すると、「申請者のユーザー名」という項目タイプ属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。

メッセージ	購買承認申請済
結果タイプ	なし
前提条件アクティビティ	承認者の選択、承認者に通知、承認権限の検証

## 終了（ノード 4、11 および 15）

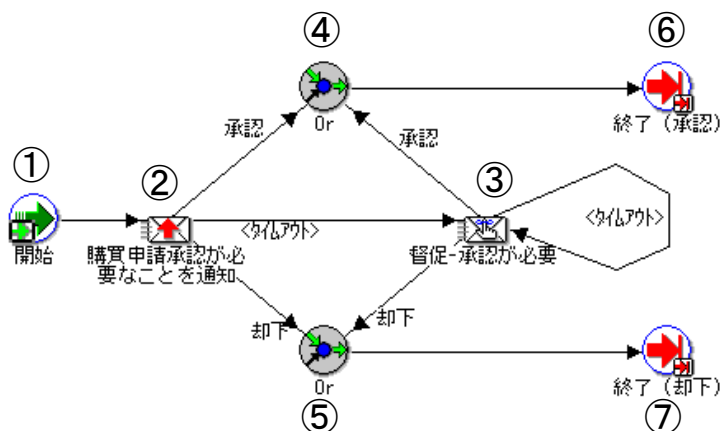
この関数アクティビティでは、プロセスの終了がマークされます。このアクティビティ自体には結果タイプはありませんが、プロセス内のこのアクティビティの各ノードには、プロセスの結果を割り当てる必要があります。プロセスの結果は、アクティビティ・ノードのプロパティ画面で割り当てます。購買申請プロセス・アクティビティには「承認」結果タイプがあるため、各「終了」アクティビティ・ノードには、「承認」選択肢タイプの選択肢コードと一致するプロセスの結果が必要です。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	開始

## 「承認者に通知」 サブプロセスの概要

「承認者に通知」サブプロセスのプロパティを表示するには、そのプロセス・アクティビティをナビゲータ・ツリーで選択し、「編集」メニューから「プロパティ」を選択します。「承認者に通知」サブプロセスには「承認」結果タイプがあります。この結果タイプは、サブプロセスの完了時に、結果が（「承認」選択肢タイプの選択肢コードに基づいて）「承認」または「却下」になることを示します。これは実行可能なプロセスではありません。つまり、最上位プロセスとしては実行できませんが、上位の別プロセスからコールされたときにのみサブプロセスとして実行されます。

「承認者に通知」サブプロセスの「プロセス」ウィンドウを表示すると、このサブプロセスが5個の一意のアクティビティで構成されていることがわかります。そのうちのいくつかは再利用され、ワークフロー・ダイアグラムに表示される7個のアクティビティ・ノードを構成しています。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



このサブプロセスは、ノード1の「開始」アクティビティから開始します。ノード2で、プロセスは指定された期間内に購買申請を承認するように、承認者に通知します。承認者が購買申請を承認すると、サブプロセスはノード6で終了し、最上位レベルの購買申請プロセスに結果「承認」を戻します。承認者が購買申請を却下すると、サブプロセスはノード7で終了し、購買申請プロセスに結果「却下」を戻します。

承認者が期日までに応答しなければ、サブプロセスはノード3へのタイムアウト・トランジションに進み、承認者に購買申請の承認を求める督促通知を送信します。また、ノード3にもタイムアウト値が割り当てられており、承認者がその時刻までに督促通知に応答しなければ、サブプロセスは次のタイムアウト・トランジションに進み、ループをたどってノード3に戻り、承認者に別の督促を送信します。このループは、承認者が購買申請を承認または却下するまで繰り返され、その時点でサブプロセスはノード6または7で終了します。

## 「承認者に通知」サブプロセスのアクティビティ

ここでは、「承認者に通知」サブプロセスの各アクティビティについて、アクティビティの表示名別に説明します。

### 開始（ノード1）

これは、単にサブプロセスの開始をマークする「標準」関数アクティビティです。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	なし

### 購買申請承認が必要なことを通知（ノード2）

このアクティビティでは、購買申請を承認または却下する必要があることが承認者に通知されます。このアクティビティは5分以内に完了しないとタイムアウトになります。

メッセージには、通知送信時に購買申請番号、購買申請摘要、購買申請金額、前の承認者名および購買申請作成者名を表示する「送信」属性が含まれます。

メッセージには、特殊な「RESULT」属性と「応答」属性が含まれます。「RESULT」属性には「処理」という表示名があり、「承認」と呼ばれる選択肢タイプから、「APPROVE」または「REJECT」の値で応答を戻すように、承認者に要求します。承認者の選択した値が、ワークフロー・エンジンが次に進むアクティビティを決める結果となります。

「応答」属性は「ノート」と呼ばれ、通知の応答にオプションのコメントを入れるよう承認者に求めます。

このアクティビティ・ノードのプロパティ画面を表示すると、「転送先ユーザー名」という項目タイプ属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。

メッセージ	購買申請に承認が必要
結果タイプ	承認
前提条件アクティビティ	承認者の選択

### 督促 - 承認が必要（ノード3）

このアクティビティは、「購買申請承認が必要なことを通知」アクティビティが完了前にタイムアウトになった場合にのみ発生します。このアクティビティでは、購買申請を承認または却下する必要があることを示す督促通知が承認者に送られます。

メッセージには、通知送信時に購買申請番号、購買申請摘要、購買申請金額、前の承認者名および購買申請作成者名を表示する「送信」属性が含まれます。

メッセージには、特殊な「RESULT」属性と「応答」属性が含まれます。「RESULT」属性には「処理」という表示名があり、「承認」と呼ばれる選択肢タイプから、「APPROVE」または「REJECT」の値で応答を戻すように、承認者に要求します。承認者の選択した値が、ワークフロー・エンジンが次に進むアクティビティを決める結果となります。

「応答」属性は「ノート」と呼ばれ、通知の応答にオプションのコメントを入れるよう承認者に求めます。

このアクティビティ・ノードのプロパティ画面を表示すると、「転送先ユーザー名」という項目タイプ属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。

メッセージ	購買承認申請の督促
結果タイプ	承認
前提条件アクティビティ	承認者の選択、購買承認申請が必要なことを通知

**Or（ノード 4 および 5）**

この「標準」関数アクティビティでは、フロー内の複数の並列分岐の 1 つでアクティビティが完了すると、それらの分岐がマージされます。

関数	WF_STANDARD.ORJOIN
結果タイプ	なし
前提条件アクティビティ	なし

**終了（ノード 6 および 7）**

この関数アクティビティでは、サブプロセスの終了がマークされます。このアクティビティ自体には結果タイプはありませんが、サブプロセス内のこのアクティビティの各ノードには、プロセスの結果を割り当てする必要があります。プロセスの結果は、アクティビティ・ノードのプロパティ画面で割り当てます。「承認者に通知」プロセス・アクティビティには「承認」結果タイプがあるため、各「終了」アクティビティ・ノードには、「承認」選択肢タイプの選択肢コードと一致するプロセスの結果が必要です。

関数	WF_STANDARD.NOP
結果タイプ	なし
前提条件アクティビティ	開始

## StartProcess 関数の例

wfrund.sql と「購買承認申請」Web 画面では、どちらも PL/SQL ストアド・プロシージャ WF\_REQDEMO.StartProcess がコールされ、購買申請プロセスが実行されます。

StartProcess を詳しく説明するために、このプロシージャをいくつかのセクションに分け、参照しやすいように各セクションに「1⇒」というような番号を付けてあります。この番号と矢印自体はプロシージャの一部ではありません。

```

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⇒ このセクションはプロシージャの仕様部で、**StartProcess** に渡す必要のあるパラメータのリストが入っています。wfrund.sql スクリプトに渡した値または「購買申請デモンストラーション」Web 画面に入力したフィールド値 (WF\_REQDEMO.Create\_Req) と同じパラメータ値が使用されます。

2⇒ プロシージャ本体の宣言部は、このセクションから始まります。**StartProcess** は、ワークフロー・エンジンの複数の PL/SQL API コールで構成されています。8-20 ページの「[Workflow Engine API](#)」を参照してください。

これらの API にはいずれも項目タイプと項目キーの入力が必要であるため、Itemtype と Itemkey はローカル引数として定義されています。引数 Itemtype は「WFDEMO」として定義されています。「WFDEMO」は「購買申請」項目タイプの内部名です。引数 Itemkey は、StartProcess プロシージャに渡される RequisitionNumber パラメータの値です。

3⇒ プロシージャ本体の実行可能部分は、このセクションから始まります。このセクションは、**CreateProcess Workflow Engine API** をコールします。この API では、購買申請プロセスの新しいランタイム・インスタンスが作成されます。このインスタンスの内部名は「WFDEMO」で、指定された項目タイプと項目キーで識別されます。8-22 ページの「[CreateProcess](#)」を参照してください。

---

---

**注意：** wfrund.sql スクリプトに <process\_int\_name> の値を渡さなければ、「購買申請」項目タイプのセレクト関数によって、どのプロセスを実行するかが判別されます。

---

---

4⇒ このセクションでは、SetItemUserKey Workflow Engine API がコールされ、購買申請プロセスの新規のランタイム・インスタンスがエンドユーザー・キーでマークされます。エンドユーザー・キーを使用すると、ユーザーは表示されたプロセス・インスタンスを問い合せて識別できます。8-25 ページの「[SetItemUserKey](#)」を参照してください。

5、6、7、8、9、10、11、12、13⇒ これらのセクションでは、SetItemAttributeText または SetItemAttributeNumber Workflow Engine API がコールされ、このプロセスに対して定義された項目タイプ属性の値が設定されます。属性は、それぞれ REQUISITION\_NUMBER、REQUISITION\_DESCRIPTION、REQUISITION\_AMOUNT、REQUESTOR\_NAME、FORWARD\_TO\_USERNAME、REQUISITION\_PROCESS\_OWNER、MONITOR\_URL、REQ\_DOCUMENT および REM\_DOCUMENT です。8-51 ページの「[SetItemAttribute](#)」を参照してください。

14⇒ このセクションでは、SetItemOwner Workflow Engine API がコールされ、購買申請プロセスの新規のランタイム・インスタンスがプロセス所有者のユーザー名でマークされます。ユーザーは、プロセス所有者を指定してプロセス・インスタンスを問い合わせできます。8-28 ページの「[SetItemOwner](#)」を参照してください。

15⇒ このセクションでは、Oracle Workflow Engine の StartProcess API がコールされ、指定した項目タイプと項目キーの購買申請プロセスが開始されます。8-30 ページの「[StartProcess](#)」を参照してください。

## 関数アクティビティの例

通常、関数アクティビティには、「アクティビティ」プロパティ画面で次の情報を指定する必要があります。

- アクティビティの内部名
- アクティビティの表示名
- アクティビティの結果タイプで、「なし」または事前定義済の選択肢タイプ名
- アクティビティがコールする PL/SQL ストアド・プロシージャの名前

また、関数アクティビティによってコールされる PL/SQL ストアド・プロシージャは、特定の API に準拠している必要があります。7-3 ページの「[関数アクティビティがコールする PL/SQL プロシージャの標準 API](#)」を参照してください。

サーバーの Oracle Workflow ディレクトリ構造の demo サブディレクトリ内で、購買申請プロセスに使用される WF\_REQDEMO ストアド・プロシージャ・パッケージを作成するスクリプトを表示できます。

## 例：承認者の選択

「承認者の選択」関数アクティビティでは、PL/SQL ストアド・プロシージャ WF\_REQDEMO.SelectApprover がコールされ、デモンストレーション・データ・モデルの従業員承認者階層に基づいて、次の承認者が判別されます。

### 結果タイプ

このアクティビティでは、承認者が見つかった場合は応答「T」、見つからなかった場合は応答「F」が戻されます。応答の候補は、標準項目タイプに関連付けられている「ブール」という名前の選択肢タイプで定義されます。

### PL/SQL ストアド・プロシージャ

ここでは、この関数アクティビティでコールされる PL/SQL ストアド・プロシージャについて詳しく説明します。このプロシージャの各セクションには、参照しやすいように「1⇒」のような番号が付いています。

```

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;
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⇒ このセクションで、ローカル引数 l\_forward\_from\_username および l\_forward\_to\_username が宣言されます。

2⇒ funcmode の値が RUN であれば、Workflow Engine API GetItemAttrText をコールして決定された FORWARD\_TO\_USERNAME 項目タイプ属性の値に l\_forward\_to\_username が割り当てられ、この購買申請が承認のために最後に転送された承認者の名前が取り出されます。8-59 ページの「[GetItemAttribute](#)」を参照してください。

3⇒ `l_forward_to_username` の値が `NULL` であれば、購買申請は承認のために転送されることがないことを意味します。この場合は、`Workflow Engine API GetItemAttrText` をコールして判別された `REQUESTOR_USERNAME` 項目タイプ属性の値が割り当てられます。

4⇒ `l_forward_from_username` に `l_forward_to_username` の値が割り当てられます。

5⇒ このセクションでは、`Workflow Engine SetItemAttrText API` がコールされ、`l_forward_from_username` の値が `FORWARD_FROM_USERNAME` 項目タイプ属性に設定されます。

6⇒ このセクションでは、関数 `GetManager` がコールされ、`WF_REQDEMO_EMP_HIERARCHY` 表から `l_forward_from_username` に格納されていた前の承認者のマネージャが戻され、そのマネージャの名前が `l_forward_to_username` に割り当てられます。

7⇒ このセクションでは、`Workflow Engine SetItemAttrText API` がコールされ、`l_forward_to_username` の値が `FORWARD_TO_USERNAME` 項目タイプ属性に設定されます。

8⇒ `l_forward_to_username` が `NULL` であれば、階層には前の承認者より上位のマネージャがないことを意味し、`resultout` が `COMPLETE:F` に設定されます。それ以外の場合は、`resultout` が `COMPLETE:T` に設定されます。

9⇒ このセクションでは、`funcmode = 'RUN'` かどうかのチェックが終了します。

10⇒ `funcmode` の値が `CANCEL` であれば、`resultout` が `COMPLETE` に設定されます。

11⇒ `funcmode` の値が `TIMEOUT` であれば、`resultout` が `COMPLETE` に設定されます。

12⇒ このセクションでは、例外が発生した場合に `WF_CORE.CONTEXT` がコールされます。

13⇒ `SelectApprover` プロシージャが終了します。

## 例：承認権限の検証

「承認権限の検証」関数アクティビティでは、`PL/SQL` ストアド・プロシージャ `WF_REQDEMO.VerifyAuthority` がコールされ、購買申請金額が承認者の支払限度額内かどうかを検証されます。これは、ストアド・プロシージャで実現したビジネス・ルールに基づいて結果を戻す、自動化された関数アクティビティの例です。

### 結果タイプ

このアクティビティでは、プロシージャが完了して承認者に購買申請の承認権限があるかどうかを示すときに、結果として「Yes」または「No」が戻されます。これらの結果値は、標準項目タイプに関連付けられている「Yes/No」という名前の選択肢タイプで定義されます。

## PL/SQL ストアド・プロシージャ

ここでは、この関数アクティビティでコールされる PL/SQL ストアド・プロシージャについて詳しく説明します。このプロシージャの各セクションには、参照しやすいように「1⇒」のような番号が付いています。また、プロシージャ内で使用されるローカル引数を示す場合は「l\_」という表記を使用しています。

```

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:Y';
        return;
    end if;
6⇒  if (funcmode = 'TIMEOUT') then
        resultout := 'COMPLETE:Y';
        return;
    end if;
7⇒  exception
    when others then
        wf_core.context('WF_REQDEMO', 'VerifyAuthority', itemtype,
                        itemkey, actid, funcmode);
        raise;
8⇒  end VerifyAuthority;

```

1⇒ このセクションで、ローカル引数 `l_forward_to_username`、`l_requisition_amount` および `l_spending_limit` が宣言されます。

2⇒ `funcmode` の値が `RUN` であれば、Workflow Engine API `GetItemAttrNumber` をコールして判別された `REQUISITION_AMOUNT` 項目タイプ属性の値が `l_requisition_amount` に設定されます。8-59 ページの「[GetItemAttribute](#)」を参照してください。

3⇒ このセクションでは、Workflow Engine API `GetItemAttrText` をコールして判別された `FORWARD_TO_USERNAME` 項目タイプ属性の値が、`l_forward_to_username` に設定されます。

4⇒ このセクションでは、現行の承認者の関数 `CheckSpendingLimit` がコールされ、購買申請金額が承認者の支払限度額以内かどうか判別されます。購買申請金額が `l_spending_limit` の値以下であれば、その承認者に承認権限があることを意味し、`resultout` に `COMPLETE:Y` が設定されます。それ以外の場合は、`resultout` に `COMPLETE:N` が設定されます。

5⇒ `funcmode` の値が `CANCEL` であれば、`resultout` が `COMPLETE:` に設定されます。

6⇒ `funcmode` の値が `TIMEOUT` であれば、`resultout` が `COMPLETE:` に設定されます。

7⇒ このセクションでは、例外が発生した場合に `WF_CORE.CONTEXT` がコールされます。

8⇒ `VerifyAuthority` プロシージャが終了します。

## 通知アクティビティの例

購買申請プロセスには、ユーザーに情報メッセージを送信する複数の通知アクティビティが含まれています。ただし、「承認者に通知」サブプロセスには、ユーザーからの応答を要求する通知アクティビティも含まれています。

通知アクティビティの場合は、「アクティビティ」プロパティ画面で次の情報を定義する必要があります。

- アクティビティの内部名
- アクティビティの表示名
- アクティビティの結果タイプで、「なし」または事前定義済の選択肢タイプ名
- 通知が送信される事前定義済のメッセージの名前

## 例：購買申請承認が必要なことを通知

「購買申請承認が必要なことを通知」アクティビティでは、「購買申請に承認が必要」というメッセージが承認マネージャに送信されます。このメッセージは、マネージャに対して購買申請を承認または却下するように要求するもので、メッセージ本文には購買申請の詳細が表示されます。

### 結果タイプ

マネージャの応答によって、プロセスが次に進むアクティビティが決まります。考えられる応答として「承認」または「却下」が、「承認」選択肢タイプで定義されています。この2つの値は、メッセージの特殊な「結果」属性（表示名、「処理」）で定義されます。「アクティビティ」プロパティ画面の「結果タイプ」フィールドで定義されているように、これらの値は通知アクティビティの結果の候補でもあります。

### メッセージ

通知の内容は、「購買申請に承認が必要」というメッセージ内で定義されます。

件名                      Requisition &REQUISITION\_NUMBER,  
                              &REQUISITION\_DESCRIPTION for &REQUISITION\_AMOUNT  
                              requires your approval

本文                      &REQ\_DOCUMENT

メッセージの件名と本文のうち、アンパサンド「&」で始まるメッセージ属性は、通知の送信時にランタイム値に置き換えられるトークンです。ただし、トークンの置換が正しく行われるためには、件名とメッセージ本文で参照されるすべてのメッセージ属性が、「送信」ソースでメッセージ属性として定義されている必要があります。

この例では、メッセージの本文には、REQ\_DOCUMENT と呼ばれる1つのメッセージ属性が含まれています。REQ\_DOCUMENT は、PL/SQL 文書タイプの属性で、同じ名前の項目タイプ属性を参照します。購買申請プロセスを開始し、ワークフロー・エンジンが *StartProcess* プロシージャを実行すると、SetItemAttrText() がコールされ、REQ\_DOCUMENT 項目属性が次の値に設定されます。

```
'PLSQL:wf_reqdemo.create_req_document/'||Itemtype||':'||ItemKey
```

この値によって、itemtype と itemkey が引数として連結され、PL/SQL 関数 wf\_reqdemo.create\_req\_document がコールされます。この関数では、この文字列が解析され、指定した itemtype:itemkey の PL/SQL 文書が作成されます。

このメッセージには、「処理」という特殊な結果メッセージ属性と「ノート」という「応答」メッセージ属性も含まれます。

結果のメッセージ属性は、メッセージのプロパティ画面の「結果」タブで定義されています。結果の属性によって、承認者は、指定した選択肢タイプによって与えられる値リストから値を選択して、応答するように要求されます。続いて、この応答が「購買申請承認が必要

なことを通知」アクティビティの結果となります。この場合、使用できる応答値は、承認選択肢タイプで定義されているように、「承認」または「却下」です。この結果によって、プロセスが次にどのアクティビティに進むかが決まります。

「応答」メッセージ属性「ノート」のタイプは、「テキスト」です。この属性は、通知への応答時にオプションのコメントを入力するように、承認者に求めます。

---

---

**注意：** メッセージの内容を表示するには、ナビゲータ・ツリーでメッセージをダブルクリックするか、メッセージを選択して「編集」メニューから「プロパティ」を選択します。

---

---

### プロセス・ノードのプロパティ

「承認者に通知」サブプロセス・ダイアグラムに「購買申請承認が必要なことを通知」アクティビティ・ノードのプロパティを表示すると、「プロセス開始」アクティビティでも「終了」アクティビティでもないため、このノードが「標準」に設定されていることがわかります。

また、「実行者」が「転送先ユーザー名」項目タイプ属性に設定されていることもわかります。これは、名前が項目タイプ属性「転送先ユーザー名」に格納されているユーザーに通知が送信されることを示します。「転送先ユーザー名」の値は、購買申請プロセスの「承認者の選択」アクティビティによって事前に決定されています。

## 製品アンケート・プロセス

グループにアンケートを送り、各メンバーの返答を得るサンプル・ワークフロー・プロセスを開始できます。開始できるアンケート・プロセスは、2種類あります。各アンケート・プロセスは、アンケートを異なる方法で実装します。ただし、どちらのアンケート・プロセスも、アンケートの返答を格納した表と一意のアンケート ID を作成する順序に基づきます。

Oracle Workflow のスタンドアロン版を使用している場合は、このサンプル・プロセスを開始できます。Oracle Applications embedded Workflow を使用している場合、このプロセスは、主にデモンストレーションではなく説明のための一例と考えてください。Oracle Applications embedded Workflow には、このデモンストレーションの設定と実行に必要なファイルが用意されていません。

---

**注意：** Oracle Applications または Oracle Self-Service Web Applications と統合された実行可能ワークフロー・プロセスの詳細は、該当する Oracle Applications のユーザーズ・ガイドまたはオンライン・マニュアルを参照してください。B-2 ページの「[Oracle E-Business Suite 埋込みの事前定義済みのワークフロー](#)」を参照してください。

---

製品アンケート・プロセスは、Oracle Workflow の「プロセスの開始」Web 画面または「ワークフロー・デモンストレーション」Web 画面から開始できます。製品アンケート・プロセスを開始するときに、アンケート申請者ロール、アンケート参加者ロール、アンケート名、タイムアウト分数および実行するプロセス名を指定する必要があります。次の 2 つのプロセス名のうち、1 つを選択できます。

- アンケート - シングル・プロセス： プロセスを 1 つ起動し、ロール内のすべての参加者にアンケートを送ります。
- アンケート - マスター / ディテール・プロセス： マスター・プロセスを起動してロール内のすべての参加者を特定した後で、参加者ごとに 1 つずつ詳細アンケート・プロセスを起動します。各詳細アンケート・プロセスは、各参加者用のアンケートを 1 参加者に送ります。

「アンケート - シングル・プロセス」を選択した場合、アンケート参加者ロールには「ロールの拡張」がオンになっている通知が送られます。これによって、通知システムからそのグループ・ロール内の各ユーザーに個々のアンケートが送られます。通知がタイムアウトになるか、すべての返答を受け取ると、アンケート通知アクティビティに関連付けられている通知後関数によって、返答がチェックされ、表に書き込まれます。その後、アンケート参加者にアンケートの結果とともに FYI 通知が送られます。プロセスが結果なしで終了します。

「マスター / ディテール・プロセス」を選択した場合、マスター・プロセスは、アンケート・ロール内のすべての参加ユーザーを特定し、各ユーザーの詳細作業項目を作成します。その後、マスター・プロセスはすべての詳細作業項目が終了するまで待機してから再開します。詳細作業項目とは、1 ユーザーにアンケート通知を送る詳細プロセスのことです。詳細プロセスのアンケート通知アクティビティに関連付けられている通知後関数によって、受け取った返答が検証され、表に書き込まれます。すべての詳細作業項目がタイムアウトになる

か、すべての詳細返答を受け取ると、ワークフロー・エンジンによって制御がマスター・プロセスに戻されます。その後、マスター・プロセスの FYI 通知によって、アンケート結果がすべてのアンケート参加者に送られます。プロセスが結果なしで終了します。

## 製品アンケート・データ・モデルのインストール

製品アンケート・データ・モデルは、Oracle Workflow のスタンドアロン版でのみインストールされます。データ・モデルは、Workflow Configuration Assistant によって自動的にインストールされます。インストールで使用するファイルは、Oracle Workflow Server のディレクトリ構造の demo および demo//<language> サブディレクトリにコピーされます。

---

**注意：** 製品アンケート・プロセスのデモンストレーションを正常に機能させるには、インストール後に、Oracle Workflow の必須設定手順を実行する必要があります。2-6 ページの「[設定の概要](#)」を参照してください。

---

インストール時には、次の処理が行われます。

- wfsrvc.sql というスクリプトがコールされ、WF\_SURVEY\_DEMO という表と WF\_SURVEYDEMO\_S という順序が作成されます。製品アンケート・プロセスは、各アンケート参加者の応答からの情報で、WF\_SURVEY\_DEMO 表を更新します。
- スクリプト wfsrvs.sql および wfsrvb.sql をコールして、WF\_SURVEYDEMO というパッケージの PL/SQL 仕様と本体を作成します。このパッケージの内容は次のとおりです。
  - 製品アンケート・ワークフローで使用する関数アクティビティでコールされる PL/SQL ストアド・プロシージャ。
  - PL/SQL プロシージャ WF\_SURVEYDEMO。Oracle Workflow Web Agent によって Create\_Survey がコールされ、製品アンケート・デモンストレーション用に Web ベースのインタフェース・ページが作成されます。
- 製品アンケート・ワークフロー定義を wfsrv.wft からデータベースにロードします。このプロセスは、Oracle Workflow Builder で表示できます。
- 製品アンケート・プロセスのデータ・モデルは、サンプルの購買申請プロセスで 사용되는デモンストレーション・ディレクトリ・サービスにも依存します。15-6 ページの「[購買申請データ・モデルのインストール](#)」を参照してください。



---

**注意：** セキュリティ上の理由から、インストール・プロセスでは、デモンストレーション・ディレクトリ・サービスのユーザーのデータベース・アカウントは作成後に自動的にロックされます。アカウントを使用する前に、wfdemoul.sql というスクリプトを使用して、それらのアカウントをロック解除する必要があります。15-6 ページの「[購買申請データ・モデルのインストール](#)」を参照してください。

---

## 製品アンケート・ワークフローの開始

製品アンケート・ワークフローを開始するには、次のどちらかの方法を使用できます。

- 「ワークフロー・デモンストレーション」ホーム・ページから、「製品アンケート」Web 画面にアクセスします。15-35 ページの「[製品アンケート Web 画面の使用](#)」を参照してください。
- 「プロセスの開始」Web 画面を使用します。12-2 ページの「[ワークフロー定義のテスト](#)」を参照してください。

### ▶ 「製品アンケート」Web 画面の使用

1. Web ブラウザに次の URL を入力して「ワークフロー・デモンストレーション」Web 画面にアクセスします。次に、「製品アンケート」リンクをクリックし、「製品アンケート」Web 画面を表示します。

```
<webagent>/wf_demo.home
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

あるいは、次の URL を入力して「製品アンケート」Web 画面を直接表示できます。

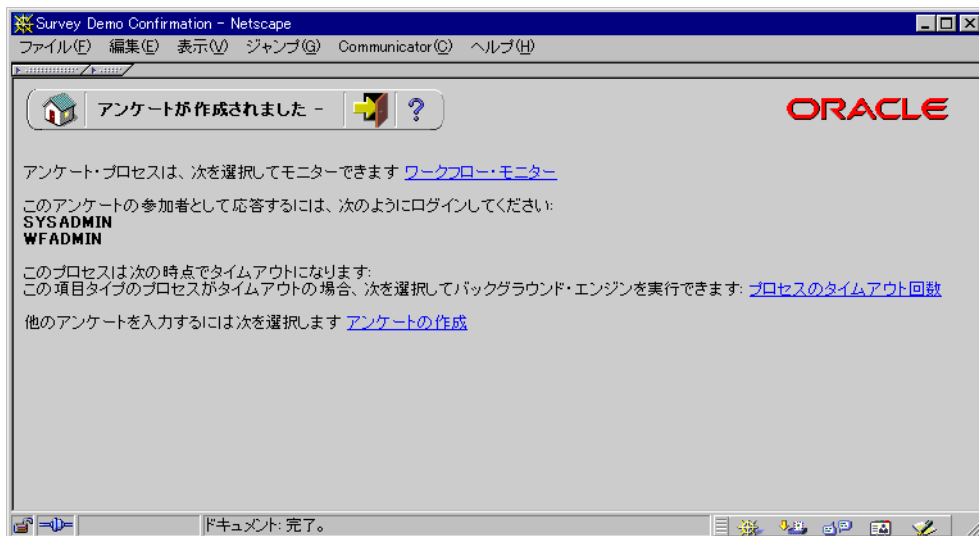
```
<webagent>/wf_surveydemo.create_survey
```

---

**注意：** どちらのページにもセキュリティが適用されるため、現行 Web セッションで有効なワークフロー・ユーザーとしてログオンしていない場合は、ページが表示される前に有効なワークフロー・ユーザーとしてのログオンを求めるプロンプトが表示されます。

---

2. 「アンケート」の「申請者」のロール名を選択します。
3. 「アンケート」の「参加者」のロール名を選択します。
4. アンケート名を入力します。
5. タイムアウト時間（分）を指定します。
6. 開始するアンケート・プロセスのタイプをオンにします。
  - 「ロールの全参加者にアンケートを送信するシングル・プロセスを使用してください。」
  - 「全ロール参加者をループするマスター・プロセスを使用して、各参加者の詳細ワーク項目を作成してください。詳細ワーク項目別に一人ずつアンケートが参加者へ送信されます。」
7. 「送信」を選択して製品アンケート・プロセスを開始し、「アンケートが作成されました」確認ページにナビゲートします。



8. 確認ページには、どのロールでログインすると、プロセスの通知を参照できるかという情報の他に、ワークフロー・モニターのアクティビティ・リストへのHTMLリンクがあり、ここで、「ダイアグラム表示」を選択すると、ADMIN モードで送信したアンケートのプロセス・ダイアグラムが表示されます。11-2 ページの「ワークフロー・モニター」を参照してください。
9. 「プロセスのタイムアウト回数」HTML リンクを選択して、バックグラウンド・エンジンでタイムアウト通知を検索し、タイムアウト時に実行される予定の次のアクティビティを実行します。
10. 別のアンケートを「製品アンケート」Web 画面に入力して送信する場合は、「アンケートの作成」HTML リンクを選択します。

「製品アンケート」項目タイプ

「アンケート - シングル・プロセス」および「アンケート - マスター / ディテール・プロセス」はどちらも、「製品アンケート」という項目タイプに関連付けられています。この項目タイプによって、使用可能なすべての製品アンケート・ワークフロー・プロセスが識別されます。この 2 つのアンケートの実施をサポートするワークフロー・プロセスは、現在、3 つあります。

- アンケート - シングル・プロセス
- アンケート - マスター / ディテール・プロセス
- 詳細アンケート・プロセス

製品アンケートのプロパティ画面を見ると、維持タイプが「一時」に、維持日数が 0 になっています。これは、この項目タイプの項目インスタンスに関連付けられているランタイム・データが、終了直後に削除の対象となることを意味します。Web ベース・インタフェースから製品アンケートを開始した場合、開始するプロセスが指定されているため、セレクト関数の指定はありません。

「製品アンケート」項目タイプには、複数の属性も関連付けられています。これらの属性は、製品アンケートの開始時の情報を記録します。属性は、プロセス全体を通して通知アクティビティと関数アクティビティによって使用され保存されます。次の表は、「製品アンケート」項目タイプの属性を示しています。

表 15-3

表示名	説明	タイプ	長さ / 書式 / 選択肢タイプ
文書 ID	文書 ID (デフォルトは項目キー)	テキスト	30
アンケート名	アンケート名 (デフォルトはユーザー・キー)	テキスト	80
アンケートの参加者	アンケートの参加者のロール	ロール	
個別の参加者	アンケート通知の宛先として詳細アンケート・プロセスが使用するロール	ロール	
タイムアウト (分)	アンケートの返答に対する動的タイムアウト期間	数値	

## 「アンケート・シングル・プロセス」の概要

「アンケート・シングル・プロセス」のプロパティを表示するには、ナビゲータ・ツリーでプロセスを選択し、「編集」メニューから「プロパティ」を選択します。「アンケート・シングル・プロセス」は実行可能です。つまり、最上位レベルのプロセスとして開始できます。

プロセス・アクティビティの「詳細」プロパティ画面は、「アンケート・シングル・プロセス」に、WFERROR というエラー項目タイプと DEFAULT\_ERROR というエラー・プロセスが関連付けられていることを示します。「アンケート・シングル・プロセス」にエラーが起きると、項目タイプ WFERROR の DEFAULT\_ERROR プロセスが自動的に開始されます。現状では、DEFAULT\_ERROR プロセスによって管理者にエラーが通知され、エラーのプロセスを再試行、中止または続行するオプションが提供されます。

「アンケート・シングル・プロセス」の「プロセス」ウィンドウを表示すると、プロセスが4つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。

「ワークフロー・デモンストレーション」ホーム・ページからアクセスできる「製品アンケート」Web 画面を使用してアンケートを送信すると、「アンケート・シングル・プロセス」ワークフローが開始されます。アンケート申請者のロール、アンケート参加者のロール、アンケート名、タイムアウト値（分）を指定し、「ロールの全参加者にアンケートを送信するシングル・プロセスを使用してください。」をオンにする必要があります。

---

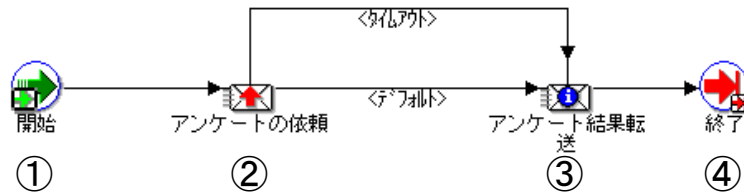
**注意：**「プロセスの開始」Web 画面からアンケート・プロセスを開始するように選択した場合は、プロセス名として「アンケート・シングル・プロセス」を選択してください。

---

ワークフローは、ノード1の「開始」アクティビティから開始します。

ノード2で、プロセスは、参加者のロールにアンケートを送信し、製品をランク付けして、その他のコメントを入力するよう求めます。

ワークフロー・エンジンが応答をすべて受け取るか、アンケート要求が（ワークフローの開始時に指定した分単位のタイムアウト期間に基づいて）タイムアウトになると、プロセスはノード3へ移り、アンケート結果付きの通知を参加者のロールに送ります。この時点でプロセスは終了します。



## 「アンケート・シングル・プロセス」のアクティビティ

### 開始（ノード1）

これは、単にプロセスの開始をマークする「標準」関数アクティビティです。

- 関数： WF\_STANDARD.NOOP
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

### アンケートの依頼（ノード2）

このアクティビティは、「アンケートの参加者」ロールのメンバーに、製品アンケートに入力が必要であることを通知します。メッセージには、アンケート名とタイムアウト期間（分）を表示する、2つの「送信」属性が含まれます。

メッセージには、ランク付けとコメントを要求する2つの「応答」属性も含まれます。

ノードのプロパティ画面を表示すると、「アンケートの参加者」という項目属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。また、このアクティビティに関連付けられているタイムアウトが、「タイムアウト」という項目属性に格納されていることもわかります。

- メッセージ： アンケート
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし

- ロールの拡張: Yes
- 通知関数: PLSQL:WF\_SURVEYDEMO.PROCESS\_SURVEY

### アンケート結果転送（ノード3）

このアクティビティは、アンケートの参加者にアンケート結果を通知します。メッセージには、アンケート名と結果スクリプトを表示する2つの「送信」属性が含まれます。結果スクリプトは「PL/SQL 文書」タイプで、アンケートの結果を要約した PL/SQL 文書を作成します。

ノードのプロパティ画面を表示すると、「アンケートの参加者」という項目属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。

- メッセージ: アンケートの結果
- 結果タイプ: なし
- 必須: No
- 前提条件アクティビティ: アンケートの依頼
- ロールの拡張: Yes
- 通知関数: なし

### 終了（ノード4）

これは、単にプロセスの終了をマークする「標準」関数アクティビティです。

- 関数: WF\_STANDARD.NOOP
- 結果タイプ: なし
- 必須: Yes
- 前提条件アクティビティ: なし
- 関数によって設定される項目属性: なし
- 関数によって取り出される項目属性: なし

## 「アンケート・マスター/ディテール・プロセス」の概要

「アンケート・マスター/ディテール・プロセス」のプロパティを表示するには、ナビゲータ・ツリーでプロセスを選択し、「編集」メニューから「プロパティ」を選択します。「アンケート・マスター/ディテール・プロセス」は実行可能です。つまり、最上位レベルのプロセスとして開始できます。

プロセス・アクティビティの「詳細」プロパティ画面は、「アンケート・マスター/ディテール・プロセス」に、WFERROR というエラー項目タイプと DEFAULT\_ERROR というエラー・プロセスが関連付けられていることを示します。「アンケート・マスター/ディテール・プロセス」にエラーが起きると、項目タイプ WFERROR の DEFAULT\_ERROR プロセスが自動的に開始されます。現状では、DEFAULT\_ERROR プロセスによって管理者にエラーが通知され、エラーのプロセスを再試行、中止または続行するオプションが提供されます。

「アンケート・マスター/ディテール・プロセス」の「プロセス」ウィンドウを表示すると、プロセスが 5 つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。

「ワークフロー・デモンストレーション」ホーム・ページからアクセスできる「製品アンケート」Web 画面を使用してアンケートを送信すると、「アンケート・マスター/ディテール・プロセス」ワークフローが開始します。アンケート申請者のロール、アンケート参加者のロール、アンケート名およびタイムアウト時間（分）を指定し、「全ロール参加者をループするマスター・プロセス」を使用して、各参加者の詳細ワーク項目を作成してください。詳細ワーク項目別に一人ずつアンケートが参加者へ送信されます。」をオンにする必要があります。

---

---

**注意：**「プロセスの開始」Web 画面からアンケート・プロセスを開始するように選択した場合は、プロセス名として「アンケート・マスター/ディテール・プロセス」を選択してください。

---

---

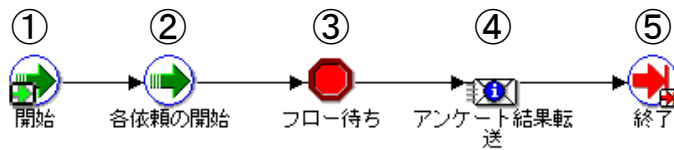
ワークフローは、ノード 1 の「開始」アクティビティから開始します。

ノード 2 で、アンケート参加者ロールのメンバーである各ユーザーを決め、各ユーザーにアンケートを送る詳細作業項目を開始します。

ノード 3 で、プロセスはすべての詳細作業項目の終了を待機してから再開します。

すべての詳細作業項目が終了すると、プロセスはノード 4 へ移り、ここで参加者ロールにアンケートの結果付き通知を送ります。この時点でプロセスは終了します。





## 「アンケート・マスター/ディテール・プロセス」のアクティビティ

### 開始（ノード 1）

これは、単にプロセスの開始をマークする「標準」関数アクティビティです。

- 関数： WF\_STANDARD.NOOP
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

### 各依頼の開始（ノード 2）

この関数アクティビティは、「アンケートの参加者」ロールの各メンバーに対して詳細作業項目を開始し、各ユーザーについて個々のアンケートを作成します。関数は、すべての関連項目属性値を「アンケート・マスター/ディテール・プロセス」から各「詳細アンケート・プロセス」の作業項目にコピーします。さらに、「マスター/ディテール・プロセス」の項目キーが、各「詳細アンケート・プロセス」の作業項目の親項目として設定されます。

- 関数： WF\_SURVEYDEMO.START\_CHILDREN
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって取り出される項目属性： USERKEY、TIMEOUT\_MINUTES
- 関数によって設定される項目属性： 各「詳細アンケート・プロセス」作業項目に対し、USERKEY、TIMEOUT\_MINUTES

### フロー待ち（ノード 3）

これは、該当する詳細プロセスが指定したアクティビティを終了するまで、フローを休止するために使用される「標準」関数アクティビティです。

- 関数: WF\_STANDARD.WAITFORFLOW
- 結果タイプ: なし
- 必須: Yes
- 前提条件アクティビティ: 各依頼の開始
- 関数によって取り出されるアクティビティ属性
  - 継続アクティビティ・ラベル: 定数、CONTINUEFLOW
  - 継続フロー: 定数、ディテール
- 関数によって設定される項目属性: なし
- 関数によって取り出される項目属性: なし

### アンケート結果転送（ノード 4）

このアクティビティは、アンケートの参加者にアンケート結果を通知します。メッセージには、アンケート名と結果スクリプトを表示する 2 つの「送信」属性が含まれます。結果スクリプトは「PL/SQL 文書」タイプで、アンケートの結果を要約した PL/SQL 文書を作成します。

ノードのプロパティ画面を表示すると、「アンケートの参加者」という項目属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。

- メッセージ: アンケートの結果
- 結果タイプ: なし
- 必須: No
- 前提条件アクティビティ: フロー待ち
- ロールの拡張: Yes
- 通知関数: なし

### 終了（ノード 5）

これは、単にプロセスの終了をマークする「標準」関数アクティビティです。

- 関数: WF\_STANDARD.NOOP
- 結果タイプ: なし
- 必須: Yes

- 前提条件アクティビティ： なし
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

## 「詳細アンケート・プロセス」の概要

「詳細アンケート・プロセス」のプロパティを表示するには、ナビゲータ・ツリーでプロセスを選択し、「編集」メニューから「プロパティ」を選択します。「詳細アンケート・プロセス」は、実行可能ではありません。つまり、最上位レベルのプロセスとして開始できず、上位レベルのプロセスからコールする必要があります。

プロセス・アクティビティの「詳細」プロパティ画面では、「詳細アンケート・プロセス」には関連するエラー項目タイプおよびエラー・プロセスがないことがわかります。エラーが起きた場合、開始されるエラー・プロセスは、詳細作業項目の親プロセスである「アンケート - マスター / ディテール・プロセス」に関連付けられているエラー項目タイプとエラー・プロセスによって決定されます。

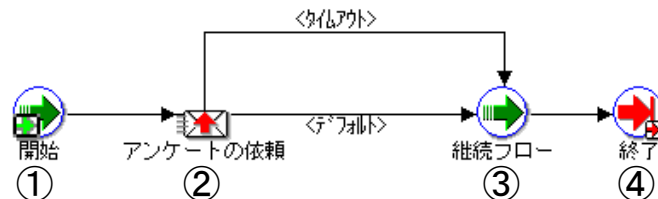
「詳細アンケート・プロセス」の「プロセス」ウィンドウを表示すると、プロセスが4つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。

親プロセスの「アンケート - マスター / ディテール・プロセス」が、「各依頼の開始」アクティビティで詳細作業項目を作成すると、「詳細アンケート・プロセス」ワークフローが開始します。

ワークフローは、ノード1の「開始」アクティビティから開始します。

ノード2で、プロセスは、個々のロールにアンケートを送信し、製品のランク付けとコメントを要求します。

すべての詳細作業項目から応答を受け取るか、アンケート要求がすべて（ワークフローの開始時に指定した分単位のタイムアウト期間に基づいて）タイムアウトになると、プロセスはノード3へ移り、親プロセスである「アンケート - マスター / ディテール・プロセス」のフローを続けます。この時点でプロセスは終了します。



## 「詳細アンケート・プロセス」のアクティビティ

### 開始（ノード 1）

これは、単にプロセスの開始をマークする「標準」関数アクティビティです。

- 関数： WF\_STANDARD.NOOP
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

### アンケートの依頼（ノード 2）

このアクティビティは、個々のロールに、製品アンケートに回答が必要であることを通知します。メッセージには、アンケート名とタイムアウト期間（分）を表示する、2つの「送信」属性が含まれます。

メッセージには、製品のランク付けとコメントを要求する2つの「応答」属性も含まれます。

ノードのプロパティ画面を表示すると、「個別の参加者」という項目属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。また、このアクティビティに関連付けられているタイムアウトが、「タイムアウト」という項目属性に格納されていることもわかります。

- メッセージ： アンケート
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- ロールの拡張： No
- 通知関数： PLSQL:WF\_SURVEYDEMO.PROCESS\_SURVEY

### 継続フロー（ノード 3）

これは、詳細プロセスの位置をマークする「標準」関数アクティビティです。詳細プロセスが終了すると、対応する停止中のマスター・プロセスはこの位置から続行します。

- 関数： WF\_STANDARD.CONTINUEFLOW
- 結果タイプ： なし

- 必須： Yes
- 前提条件アクティビティ： アンケートの依頼
- 関数によって取り出されるアクティビティ属性
  - － 待機中のアクティビティ・ラベル： 定数、WAITFORFLOW
  - － 待機中フロー： 定数、マスター
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

### 終了（ノード4）

これは、単にプロセスの終了をマークする「標準」関数アクティビティです。

- 関数： WF\_STANDARD.NOOP
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

## ドキュメント・レビュー・プロセス

---

**注意：** 文書管理機能は今後使用する目的で確保されています。ドキュメント・レビュー・プロセスに関するこの説明は、単なる参考として記載されています。

---

ドキュメント・レビュー・プロセスは、通知とドキュメント管理システムを統合し、添付文書をレビューして承認するように、承認者に要求します。Oracle Workflow 管理ロール内のユーザーは、Oracle Workflow の「プロセスの開始」Web 画面または「ワークフロー・デモンストレーション」Web 画面からドキュメント・レビュー・プロセスを開始できます。プロセスを開始するには、「項目キー」、「ユーザー・キー」、「プロセス所有者」、「文書の送信」、「文書の所有者」および「文書のレビューア」の項目属性値を指定します。

レビュー・プロセスのプロセス定義は、Oracle Workflow のスタンドアロン版では、Workflow Configuration Assistant によって自動的にインストールされます。Oracle Applications embedded Workflow では、AutoUpgrade によって自動的にインストールされます。

このデモンストレーションで、ドキュメント・レビュー要求を送信すると、プロセスは、指定したレビュー担当に文書の承認を求める通知を送ります。オプションで、レビュー担当が応答時に代替文書を提供することもできます。レビュー担当が文書を承認した場合、プロセスは「承認」という結果で終了します。レビュー担当が文書を却下した場合、申請者は文書の承認を求めて再送信できます。申請者が文書の再送信を選択すると、プロセスはループ・バックし、「文書のレビュー」通知を送ります。再送信しなければ、プロセスは「却下」という結果で終了します。

### 関連項目：

15-2 ページ [「サンプル・ワークフロー・プロセス」](#)

12-2 ページ [「ワークフロー定義のテスト」](#)

## 「文書管理」項目タイプ

ドキュメント・レビュー・プロセスは、「文書管理」という項目タイプに関連付けられています。この項目タイプは、ドキュメント管理システムの統合に関連するデモンストレーション・ワークフロー・プロセスをすべて識別します。「文書管理」に関連付けられているワークフロー・プロセスは、現在、「レビュー」1つのみです。

ドキュメント管理のプロパティ画面を見ると、維持タイプが「一時」に、維持日数が0になっています。これは、この項目タイプの作業項目に関連付けられているランタイム・データは、終了直後に削除の対象となることを意味します。Web ベース・インタフェースからドキュメント管理プロセスを開始したときに、開始するプロセスが指定されていたため、項目タイプはセレクト関数を持ちません。

「文書管理」項目タイプには、複数の属性も関連付けられています。これらの属性は、ドキュメント管理プロセスの開始時の情報を記録します。属性は、プロセス全体を通して通知アクティビティと関数アクティビティによって使用され保存されます。次の表は、「文書管理」項目タイプの属性を示しています。

表 15-4

表示名	説明	タイプ	長さ / 書式 / 選択肢タイプ
送信ドキュメント	レビューのために送信されるドキュメント	ドキュメント	フレーム・ターゲット: 新規ウィンドウ
文書の所有者	レビューのために送信されるドキュメントの所有者	ロール	
文書のレビューア	ドキュメントのレビュー担当のロール	ロール	
コメント	レビュー担当が入力するコメント	テキスト	
応答ドキュメント	レビュー後に編集を経たドキュメント	ドキュメント	フレーム・ターゲット - 新規ウィンドウ 表 15-4

## ドキュメント・レビュー・プロセスの概要

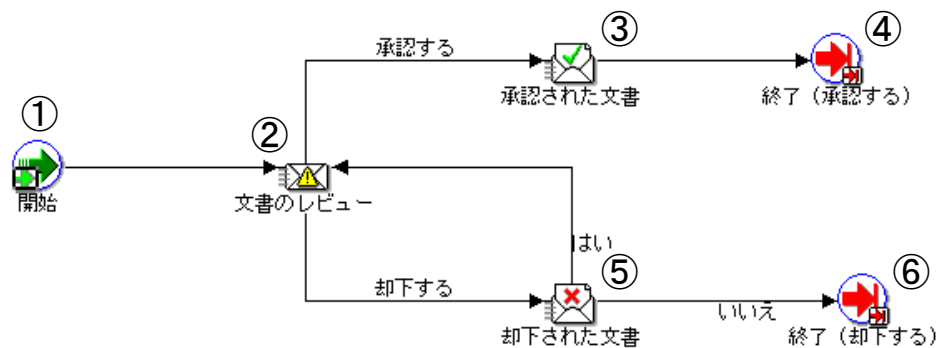
ドキュメント・レビュー・プロセスのプロパティを表示するには、ナビゲータ・ツリーでプロセスを選択し、「編集」メニューから「プロパティ」を選択します。ドキュメント・レビュー・プロセスには、プロセスの完了時に結果が「承認」になるか「却下」になるか（標準項目タイプに関連付けられた「承認」選択肢タイプの選択肢コード）を示す「承認」という結果タイプがあります。このプロセス・アクティビティも実行可能です。つまり、最上位レベルのプロセスとして開始できます。

プロセス・アクティビティの「詳細」プロパティ画面は、「ドキュメント・レビュー」に、WFERROR というエラー項目タイプと DEFAULT\_ERROR というエラー・プロセスが関連付けられていることを示します。「ドキュメント・レビュー」にエラーが起きると、項目タイプ WFERROR の DEFAULT\_ERROR プロセスが自動的に開始されます。現状では、DEFAULT\_ERROR プロセスによって管理者にエラーが通知され、エラーのプロセスを再試行、中止または続行するオプションが提供されます。

ドキュメント・レビュー・プロセスのプロセス・ウィンドウを表示すると、プロセスが6つのアクティビティ・ノードから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。

ワークフローは、ノード1の「開始」アクティビティから開始します。

ノード2の「文書のレビュー」では、プロセスは、ドキュメントのレビュー担当に通知を送り、「文書の送信」のレビューと承認を求めます。レビュー担当がドキュメントを承認すると、プロセスはノード3の「承認された文書」に移り、申請者に承認を通知します。この場合、プロセスは「承認する」という結果で終了します。レビュー担当がドキュメントを却下すると、プロセスはノード5の「却下された文書」に移り、申請者に却下を通知します。申請者は、ドキュメントが承認されるように再送信するか、結果を受け入れるかを選択できます。ドキュメントが再送信される場合、プロセスはノード2の「文書のレビュー」に戻ります。申請者が結果を受け入れた場合、プロセスは「却下する」という結果で終了します。





## ドキュメント・レビュー・プロセスのアクティビティ

### 開始（ノード1）

これは、単にプロセスの開始をマークする「標準」関数アクティビティです。

- 関数： WF\_STANDARD.NOOP
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって取り出されるアクティビティ属性： なし
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

### 文書のレビュー（ノード2）

これは、ドキュメントのレビュー担当に承認を求めるメッセージを送る通知アクティビティです。オプションで、応答時にドキュメントを編集して返すように求めることもできます。メッセージには、2つの「送信」属性が含まれます。1つは、ドキュメントの所有者を表示するもので、もう1つは、通知に添付情報として表示される文書属性です。添付アイコンを選択すると、ウィンドウが新たに開き、ドキュメントが表示されます。メッセージには、「コメント」と「応答ドキュメント」を要求する2つの「応答」属性も含まれます。

- メッセージ： ドキュメント送信
- 結果タイプ： 承認
- 必須： Yes
- 前提条件アクティビティ： なし
- 通知によって取り出されるアクティビティ属性： なし

### 承認された文書（ノード3）

これは、ドキュメント・レビューの申請者に、ドキュメントが承認されたことを伝えるメッセージを送る通知アクティビティです。メッセージには、「文書の所有者」、「文書のレビュー担当」、「文書の送信」へのリンク、レビュー担当からの「コメント」およびオプションの「応答ドキュメント」添付を表示する、5つの「送信」属性が含まれます。「応答ドキュメント」添付を選択すると、ウィンドウが新たに開かれ、ドキュメントが表示されます。

- メッセージ： ドキュメントの応答： 承認
- 結果タイプ： 承認
- 必須： Yes

- 前提条件アクティビティ： ドキュメントのレビュー
- 通知によって取り出されるアクティビティ属性： なし

### 終了（ノード4および6）

これは、単にプロセスの終了をマークする「標準」関数アクティビティです。

- 関数： WF\_STANDARD.NOOP
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって取り出されるアクティビティ属性： なし
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

### 却下された文書（ノード5）

これは、ドキュメント・レビューの申請者に、ドキュメントが却下されたことを伝えるメッセージを送る通知アクティビティです。メッセージには、「文書の所有者」、「文書のレビュー担当」、「文書の送信」へのリンクおよびオプションの「応答ドキュメント」添付を表示する、4つの「送信」属性が含まれます。添付アイコンを選択すると、ウィンドウが新たに開き、ドキュメントが表示されます。メッセージには、レビュー担当からのコメントを表示する1つの「応答」属性も含まれており、ドキュメント・レビュー要求が再送信される場合に、申請者からのコメントを要求します。

- メッセージ： ドキュメントの応答： 却下
- 結果タイプ： Yes/No
- 必須： Yes
- 前提条件アクティビティ： ドキュメントのレビュー
- 通知によって取り出されるアクティビティ属性： なし

## エラー・チェック・プロセス

エラー・チェック・プロセスは、Oracle Workflow の項目アクティビティ・ステータス表でエラーのアクティビティを検索します。このプロセスの主な目的は、Oracle Alert の定期警告と同じ機能を持つワークフロー・プロセスを設計する場合の、Oracle Workflow の使用方法を示すことです。エラー・チェック・プロセスは、指定した間隔でデータベース例外のチェックを行うように設定できます。

エラー・チェック・プロセスのプロセス定義は、Oracle Workflow のスタンドアロン版では、Workflow Configuration Assistant によって自動的にインストールされます。Oracle Applications embedded Workflow では、AutoUpgrade によって自動的にインストールされます。

このデモンストレーションでエラー・チェック・プロセスを開始すると、プロセスはワークフロー項目アクティビティ・ステータス表をチェックし、エラーのあるアクティビティを検索する関数を実行します。エラーが見つかったら、指定されたアラートの宛先にエラーをリストした通知を送ります。1 回のみ実行するか、指定した間隔で実行するかの設定に応じて、エラー・チェック・プロセスは続行または終了します。特定の間隔で実行するように指定した場合、プロセスは必要な時間だけ待機し、ワークフロー項目アクティビティ・ステータス表を再チェックします。待機 / チェックのループが続き、プロセスは指定した終了日に終了します。

エラー・チェック・プロセスは、「プロセスの開始」Web 画面または「ワークフロー・デモンストレーション」Web 画面から開始できます。「項目キー」、「ユーザー・キー」、「プロセス所有者」、「警告の宛先」、プロセスの「開始日付」と「終了日付」およびエラーをチェックする「頻度」（曜日、日付、時刻、日数または 1 回のみ）を指定する必要があります。

### 関連項目：

15-2 ページ [「サンプル・ワークフロー・プロセス」](#)

12-2 ページ [「ワークフロー定義のテスト」](#)

「定期警告」項目タイプ

エラー・チェック・プロセスは、「定期警告」という項目タイプに関連付けられています。この項目タイプは、Oracle Workflow 全体で、定期警告機能の実装に関連するプロセスをすべて識別します。定期警告に関連するワークフロー・プロセスは、現在、2 つあります。エラー・チェックとユーザー定義の警告処理です。

定期警告のプロパティ画面を見ると、維持タイプが「一時」に、維持日数が 0 になっています。これは、この項目タイプの項目インスタンスに関連付けられているランタイム・データが、終了直後に削除の対象となることを意味します。Web ベース・インタフェースから定期警告プロセスを開始した場合、開始するプロセスが指定されているため、セレクト関数の指定はありません。

「定期警告」項目タイプには、複数の属性も関連付けられています。これらの属性は、定期警告プロセスの開始時の情報を記録します。属性は、プロセス全体を通して通知アクティビティと関数アクティビティによって使用され保存されます。次の表は、「定期警告」項目タイプの属性を示しています。

表 15-5

表示名	説明	タイプ	長さ / 書式 / 選択肢タイプ
警告の宛先	アラート例外が検出された場合に通知するロール	ロール	
開始日	アラート・チェックの開始日 (デフォルトは今日)	日付	DD-MON-YYYY HH24:MI:SS
終了日	アラート・チェックの終了日 (デフォルトは、2010 年 12 月 31 日 12:12:12 PM)	日付	DD-MON-YYYY HH24:MI:SS
頻度	アラートをチェックする頻度	選択肢	待機モード
頻度： 毎月何日	頻度が「毎月何日」の場合、日付が必須	選択肢	毎月何日
頻度： 毎週何曜日	頻度が「毎週何曜日」の場合、曜日が必須	選択肢	毎週何曜日
頻度： 毎日何時	時間はオプション、ただし「待機」アクティビティで「頻度」の値に使用	日付	HH24:MI
頻度： 日	頻度が「相対時間」の場合、数値が必須	数値	
1 回のみ	アラートを 1 回のみ実行	選択肢	はい / いいえ

## エラー・チェック・プロセスの概要

エラー・チェック・プロセスのプロパティを表示するには、ナビゲータ・ツリーでプロセスを選択し、「編集」メニューから「プロパティ」を選択します。エラー・チェック・プロセスは実行可能です。つまり、最上位レベルのプロセスとして開始できます。

プロセス・アクティビティの「詳細」プロパティ画面は、「エラー・チェック」に、WFERROR というエラー項目タイプと DEFAULT\_ERROR というエラー・プロセスが関連付けられていることを示します。「エラー・チェック」にエラーが起きると、項目タイプ WFERROR の DEFAULT\_ERROR プロセスが自動的に開始されます。現状では、DEFAULT\_ERROR プロセスによって管理者にエラーが通知され、エラーのプロセスを再試行、中止または続行するオプションが提供されます。

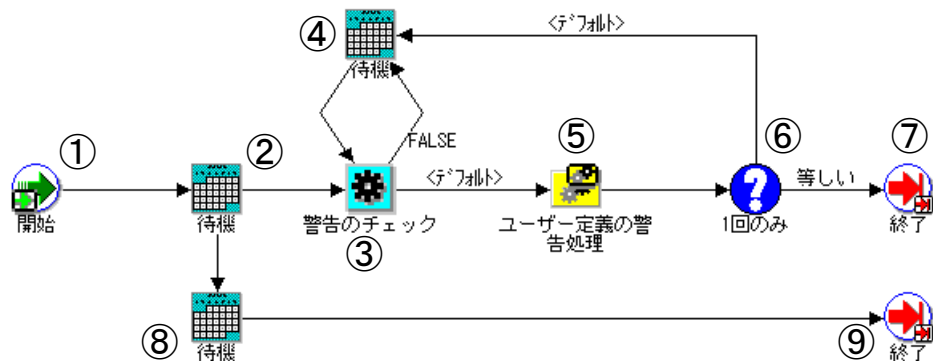
エラー・チェック・プロセスの「プロセス」ウィンドウを表示すると、このプロセスが 6 個の一意のアクティビティで構成されていることがわかります。そのうちのいくつかが再利用され、ワークフロー・ダイアグラムに表示される 9 個のアクティビティ・ノードを構成しています。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。

「プロセスの開始」Web ベース・インタフェースからプロセスを起動すると、エラー・チェック・ワークフローが開始します。「項目キー」、「ユーザー・キー」、「プロセス所有者」、「警告の宛先」、プロセスの「開始日付」と「終了日付」およびエラーをチェックする「頻度」（曜日、日付、時刻、日数または 1 回のみ）を指定する必要があります。

ワークフローは、ノード 1 の「開始」アクティビティから開始します。

ノード 2 でプロセスは休止し、開始日まで待機します。待機時間が経過すると、プロセスはノード 3 で、ワークフロー項目アクティビティ・ステータス表でエラーを検索する関数アクティビティを実行します。関数アクティビティがエラーを検出できなければ、プロセスはノード 4 を実行します。ここで、プロセスの開始時に指定された頻度に基づいた期間だけ休止します。頻度をベースにした待機時間が経過すると、プロセスはステータス表でエラーを再検索します。エラーを検出した場合、プロセスはノード 5 で、アラートの宛先にエラーの通知を送るプロセス・アクティビティを実行します。その後、プロセスはノード 6 を実行し、エラー・チェック・プロセスを 1 回のみ実行するかどうかを評価します。プロセスの実行が 1 回のみの場合、プロセスはノード 7 で終了します。1 回のみでない場合、プロセスはノード 4 の頻度ベースの「待機」アクティビティに戻ります。

ノード 2～5 の一連のアクティビティが実行される間に、プロセスは並行してノード 8 へも移ります。これは、指定した終了日に達するまで、作業項目のスキャンをループさせたままにする別の「待機」アクティビティです。指定の終了日になると、プロセスはノード 9 で終了します。



## エラー・チェック・プロセスのアクティビティ

### 開始（ノード 1）

これは、単にプロセスの開始をマークする「標準」関数アクティビティです。

- 関数： WF\_STANDARD.NOOP
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって取り出されるアクティビティ属性： なし
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

### 待機（ノード 2）

これは、指定した時間プロセスを休止させる「標準」関数アクティビティです。

プロセスで「待機」アクティビティを使用するには、「待機」アクティビティを終了させるために、待機時間が経過したかどうかを評価するバックグラウンド・エンジンを、1つ以上設定する必要があります。

- 関数： WF\_STANDARD.WAIT
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし

- 関数によって取り出されるアクティビティ属性
  - － 待機モード： 定数、絶対日付
  - － 絶対日付： 項目属性、開始日
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

### 警告のチェック（ノード3）

これは、ワークフロー項目アクティビティ・ステータス表の行をスキャンして、ERROR ステータスのアクティビティを検索する関数アクティビティです。

- 関数： WF\_ALERT.CHECKALERT
- 結果タイプ： ブール
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって取り出されるアクティビティ属性： なし
- 関数によって作成される項目属性： LAST\_CHECKED
- 関数によって取り出される項目属性： LAST\_CHECKED
- 関数によって設定される項目属性： LAST\_CHECKED

### 待機（ノード4）

これは、指定した時間プロセスを休止させる「標準」関数アクティビティです。

プロセスで「待機」アクティビティを使用するには、「待機」アクティビティを終了させるために、待機時間が経過したかどうかを評価するバックグラウンド・エンジンを、1つ以上設定する必要があります。

- 関数： WF\_STANDARD.WAIT
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって取り出されるアクティビティ属性
  - － 待機モード： 項目属性、頻度
  - － 絶対日付： 項目属性、開始日
  - － 毎月何日： 項目属性、毎月何日

- － 毎週何曜日： 項目属性、毎週何曜日
- － 相対時間： 項目属性、頻度： 日
- － 毎日何時： 項目属性、頻度： 毎日何時
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

### ユーザー定義の警告処理（ノード5）

このアクティビティは、アラート例外が検出されるたびに一連のアクティビティを実行するサブプロセスです。このサブプロセスを参照するには、ナビゲータ・ツリーで「プロセス」のブランチの下にある「ユーザー定義の警告処理」をダブルクリックします。現状では、サブプロセスは、検出されたエラーに関する通知をアラートの宛先に送ります。

- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって取り出されるアクティビティ属性： なし

### 1 回のみ（ノード6）

これは、値を別の値と比較する「標準」関数アクティビティです。

- 関数： WF\_STANDARD.COMPARE
- 結果タイプ： 比較
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって取り出されるアクティビティ属性
  - － テスト値： 項目属性、1 回のみ
  - － 参照値： 定数、Y
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

### 終了（ノード7および9）

これは、単にプロセスの終了をマークする「標準」関数アクティビティです。

- 関数： WF\_STANDARD.NOOP
- 結果タイプ： なし



- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって取り出されるアクティビティ属性： なし
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

## 待機（ノード 8）

これは、指定した時間プロセスを休止させる「標準」関数アクティビティです。

プロセスで「待機」アクティビティを使用するには、「待機」アクティビティを終了させるために、待機時間が経過したかどうかを評価するバックグラウンド・エンジンを、1つ以上設定する必要があります。

- 関数： WF\_STANDARD.WAIT
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって取り出されるアクティビティ属性
  - － 待機モード： 定数、絶対日付
  - － 絶対日付： 項目属性、終了日
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

## ユーザー定義の警告処理プロセスの概要

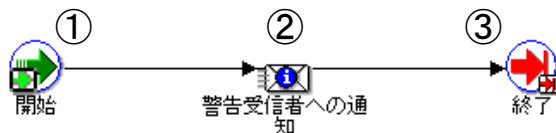
ユーザー定義の警告処理プロセスのプロパティを表示するには、ナビゲータ・ツリーでプロセスを選択し、「編集」メニューから「プロパティ」を選択します。ユーザー定義の警告処理プロセスは、実行可能ではありません。つまり、最上位レベルのプロセスとして開始できず、上位レベルのプロセスからコールする必要があります。

プロセス・アクティビティの「詳細」プロパティ画面では、このプロセス・アクティビティに関連するエラー項目タイプおよびエラー・プロセスがないことがわかります。エラーが起きた場合、開始されるエラー・プロセスは、親プロセスであるエラー・チェックに関連付けられているエラー項目タイプとエラー・プロセスによって決定されます。

ユーザー定義の警告処理プロセスの「プロセス」ウィンドウを表示すると、プロセスが3つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。

ユーザー定義の警告処理プロセスは、エラー・チェック・プロセスのサブプロセスとして開始されます。

ワークフローは、ノード1の「開始」アクティビティから開始します。ノード2で、プロセスは検出されたエラーをリストした通知をアラートの宛先へ送ります。この時点でプロセスは終了します。



## ユーザー定義の警告処理プロセスのアクティビティ

### 開始（ノード 1）

これは、単にプロセスの開始をマークする「標準」関数アクティビティです。

- 関数： WF\_STANDARD.NOOP
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって取り出されるアクティビティ属性： なし
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

### 警告受信者への通知（ノード 2）

これは、指定したアラートの宛先にエラー・レポートを送る通知アクティビティです。

メッセージには、エラー・レポートという 1 つの送信属性が含まれます。これは、PL/SQL プロシージャ WF\_ALERT.ErrorReport によって作成される値を持つ PL/SQL 文書属性です。

- メッセージ： 例外の検出された FYI
- 結果タイプ： なし
- 必須： No
- 前提条件アクティビティ： なし

### 終了（ノード 3）

これは、単にプロセスの終了をマークする「標準」関数アクティビティです。

- 関数： WF\_STANDARD.NOOP
- 結果タイプ： なし
- 必須： Yes
- 前提条件アクティビティ： なし
- 関数によって取り出されるアクティビティ属性： なし
- 関数によって設定される項目属性： なし
- 関数によって取り出される項目属性： なし

## イベント・システム・デモンストレーション

イベント・システム・デモンストレーションは、2つのシステム間でビジネス・ドキュメントを送信するイベントの使用例です。一方のシステムで発注を入力して、デモンストレーション・プロセスを開始します。Oracle Workflow は発注 XML 文書を生成して、この文書をもう一方のシステムに送信します。発注先のシステムは発注を処理して、発注承認、アドバンスト出荷通知および請求書に相当する3つのXML文書を発注元のシステムに返送します。

Oracle Workflow のスタンドアロン版を使用している場合は、このサンプル・プロセスを開始できます。Oracle Applications embedded Workflow を使用している場合、このプロセスは、主にデモンストレーションではなく説明のための一例と考えてください。Oracle Applications embedded Workflow には、このデモンストレーションの設定と実行に必要なファイルが用意されていません。

---

---

**注意：** Oracle Applications または Oracle Self-Service Web Applications と統合された実行可能ワークフロー・プロセスの詳細は、該当する Oracle Applications のユーザーズ・ガイドまたはオンライン・マニュアルを参照してください。B-2 ページの「[Oracle E-Business Suite 埋込みの事前定義済のワークフロー](#)」を参照してください。

---

---

イベント・システム・デモンストレーションを実行する前に、デモンストレーションで使用するパイヤー・システムとサプライヤ・システムを設定する必要があります。

---

---

**注意：** 2つのシステムは、個別に設定したり、同じシステムをパイヤーおよびサプライヤとして使用することができます。

---

---

イベント・システム・デモンストレーション・プロセスは、「ワークフロー・デモンストレーション」Web 画面のパイヤー・システムから開始できます。プロセスを開始するときに、発注についてオーダー番号、項目番号、項目の説明、納品日、合計金額およびオーダー・リクエスト・ロールを指定する必要があります。

発注を送信すると、注文情報がデータベース表に挿入され、オーダー番号をイベント・キーとして発注イベントが呼び出されます。発注イベントを呼び出すと、このイベントへのサブスクリプションから、「ローカル」ソース・タイプを持つ2つのサブスクリプションがトリガーされます。一方のサブスクリプションでは、関連 ID がイベント・メッセージに追加されます。関連 ID は、接頭辞 PO とイベント・キー（オーダー番号）で構成されます。

発注イベントへのもう一方のサブスクリプションには、すべてのイベント・データが必要です。このため、イベント・マネージャはイベントのジェネレート関数を実行して、有効な発注 XML 文書を作成します。もう一方のサブスクリプションでは、「イベント・システムのデモ」項目タイプのパイヤー：最上位の発注プロセスにイベントが送信されます。ワークフロー・エンジンにより、関連 ID をイベント・キーとしてこのプロセスの新しいインスタンスが作成されます。

バイヤー:最上位の発注プロセスで発注进行处理しているときに、標準の外部 Java 関数アクティビティによってオーダー・リクエストの名前が発注 XML 文書から取得されます。発注プロセスは、この情報を基にしてオーダー・リクエストに通知を送信します。発注プロセスは、発注イベント・メッセージをサプライヤ・システムに送信し、サプライヤからの応答を待機します。

発注イベントがサプライヤ・システムに着信すると、このイベントへのサブスクリプションのうち、「外部」ソース・タイプを持つ2つのサブスクリプションがトリガーされます。一方のサブスクリプションでは、イベント・メッセージの相関 ID の構成が、接頭辞 SO とイベント・キー（オーダー番号）に変更されます。もう一方のサブスクリプションでは、「イベント・システムのデモ」項目タイプのサプライヤ:最上位のオーダー・プロセスにイベントが送信されます。ワークフロー・エンジンにより、新しい相関 ID をイベント・キーとしてこのプロセスの新しいインスタンスが作成されます。

サプライヤ:最上位のオーダー・プロセスで発注が処理されているときに、標準の外部 Java 関数アクティビティによってオーダー番号と説明が発注 XML 文書から取得されます。次の XML 文書を含むイベント・メッセージが、オーダー・プロセスからバイヤー・システムに送信されます。

- 発注承認
- アドバンスト出荷通知
- 請求書

バイヤー・システムでは、これらのイベントによって「外部」ソース・タイプのサブスクリプションがトリガーされます。各イベントには、2つのサブスクリプションが割り当てられています。一方のサブスクリプションでは、接頭辞 PO とイベント・キー（オーダー番号）で構成される相関 ID が追加されます。もう一方のサブスクリプションでは、バイヤー:最上位の発注プロセスにイベント・メッセージが送信されます。このとき、イベント・メッセージの送信元プロセスは、相関 ID によって照合されます。発注プロセスは、イベント・メッセージを受信すると、オーダー・リクエストに通知します。3つの応答文書がすべて着信すると、プロセスは完了します。

## イベント・システム・デモンストレーション・データ・モデルのインストール

イベント・システム・データ・モデルは、Oracle Workflow のスタンドアロン版でのみインストールされます。データ・モデルは、Workflow Configuration Assistant によって自動的にインストールされます。インストールで使用されるファイルは、Oracle Workflow Server のディレクトリ構造の demo および demo//<language> サブディレクトリにコピーされます。

---

---

**注意：** イベント・システム・デモンストレーションを正常に機能させるには、インストール後に、Oracle Workflow の必須設定手順を実行する必要があります。2-6 ページの「[設定の概要](#)」を参照してください。

---

---

インストール時には、次の処理が行われます。

- スクリプト wfevdemc.sql がコールされ、WF\_EVENTDEMO\_ITEMS および WF\_EVENTDEMO\_PO という 2 つの表が作成されます。表 WF\_EVENTDEMO\_ITEMS には、発注時に選択可能な項目が格納されます。イベント・システム・デモンストレーション・プロセスでは、発注情報によって表 WF\_EVENTDEMO\_PO が更新されます。
- スクリプト wfevdems.sql および wfevdemb.sql がコールされ、WF\_EVENTDEMO というパッケージの PL/SQL 仕様と本体が作成されます。このパッケージの内容は次のとおりです。
  - PL/SQL ストアド・プロシージャ。イベント・システム・デモンストレーション・ワークフローで使用される関数アクティビティによってコールされます。
  - PL/SQL プロシージャの WF\_EVENTDEMO.Create\_Order および WF\_EVENTDEMO.Track\_Order。Oracle Workflow の Web Agent によってコールされ、イベント・システム・デモンストレーション・プロセスのデモンストレーションに対して、Web ベースのインタフェース・ページを生成します。
- イベント・システム・デモンストレーション・ワークフロー定義が、wfevdeme.wft からデータベースにロードされます。このプロセスは、Oracle Workflow Builder で表示できます。
- イベント・システム・デモンストレーション・プロセスのデータ・モデルには、サンプルの購買申請プロセスで使用するデモンストレーション・ディレクトリ・サービスも含まれます。15-6 ページの「[購買申請データ・モデルのインストール](#)」を参照してください。

---

**注意：** セキュリティ上の理由から、インストール・プロセスでは、デモンストレーション・ディレクトリ・サービスのユーザーのデータベース・アカウントは作成後に自動的にロックされます。アカウントを使用する前に、wfdemoul.sql というスクリプトを使用して、それらのアカウントをロック解除する必要があります。15-6 ページの「[購買申請データ・モデルのインストール](#)」を参照してください。

---

## イベント・システム・デモンストレーション・ワークフローの開始

イベント・システム・デモンストレーションには、2つのワークフロー対応システム（バイヤー・システムとサプライヤ・システム）が必要です。2つのシステムは、個別に設定したり、同じシステムをバイヤーおよびサプライヤとして使用することができます。イベント・システム・デモンストレーションを実行する前に、使用するシステムを設定する必要があります。15-65 ページの「[イベント・システム・デモンストレーション・ワークフローの設定](#)」を参照してください。

2つのシステムを設定したら、「イベント: Buyer Workbench」デモンストレーション Web 画面を使用して、イベント・システム・デモンストレーション・ワークフローを開始できます。ワークフローを開始したら、サプライヤ・システムの「オーダーの追跡」デモンストレーション Web 画面を使用して、イベント・システム・デモンストレーション・ワークフローの処理を継続できます。15-66 ページの「[Buyer Workbench からのイベント・システム・デモンストレーション・ワークフローの開始](#)」および 15-68 ページの「[サプライヤ・システムでのイベント・システム・デモンストレーション・ワークフローの継続](#)」を参照してください。

### ▶ イベント・システム・デモンストレーション・ワークフローの設定

1. Oracle Workflow の 2 つのインストレーションを個別に使用する場合は、一方をバイヤー・システム、もう一方をサプライヤ・システムとして指定します。
2. Oracle Workflow の 2 つのインストレーションを個別に使用する場合は、2 つのシステムを相互にサインアップして、イベント・メッセージを交換します。13-73 ページの「[システムのサインアップ](#)」を参照してください。
3. Oracle Workflow の 2 つのインストレーションを個別に使用する場合は、バイヤー・システム上で、「ローカル」ソース・タイプおよびフェーズ 2 を持つ demo.oracle.apps.wf.po.create イベントへの事前定義済サブスクリプションを検索します。サプライヤ・システムの標準 WF\_IN エージェントを宛先エージェントとして選択して、このサブスクリプションを編集します。13-48 ページの「[イベント・サブスクリプションの定義](#)」を参照してください。
4. Oracle Workflow の 2 つのインストレーションを個別に使用する場合は、サプライヤ・システム上で、「外部」ソース・タイプおよびフェーズ 2 を持つ demo.oracle.apps.wf.po.create イベントへの事前定義済サブスクリプションを検索します。バイヤー・システムの標準 WF\_IN エージェントを宛先エージェントとし

て選択して、このサブスクリプションを編集します。13-48 ページの「[イベント・サブスクリプションの定義](#)」を参照してください。

5. Java 関数アクティビティ・エージェントが、システムで動作していることを確認します。2-81 ページの「[手順 WF-13 Java 関数アクティビティ・エージェントの設定](#)」を参照してください。
6. オプションで、バックグラウンド・エンジンがシステムで 10 ～ 30 秒おきに実行されるようにスケジュールします。2-40 ページの「[手順 WF-8 バックグラウンドのワークフロー・エンジンの設定](#)」を参照してください。

「イベント: オーダーの追跡」画面の「プロセス・オーダー」リンクを選択して、デモンストレーション中にバックグラウンド・エンジンを手動で実行することもできます。

### ► Buyer Workbench からのイベント・システム・デモンストレーション・ワークフローの開始

1. バイヤー・システム上の Web ブラウザに次の URL を入力して、「ワークフロー・デモンストレーション」Web 画面にアクセスします。

```
<webagent>/wf_demo.home
```

<webagent> は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。2-13 ページの「[グローバル・ワークフロー・プリファレンスの設定](#)」を参照してください。

「イベント: Buyer Workbench」リンクを選択して、「Buyer Workbench」Web 画面を表示します。

または、次の URL を入力して「Buyer Workbench」Web 画面を直接表示することもできます。

```
<webagent>/wf_eventdemo.create_order
```

---

---

**注意：** どちらのページにもセキュリティが適用されるため、現行 Web セッションで有効なワークフロー・ユーザーとしてログオンしていない場合は、ページが表示される前に有効なワークフロー・ユーザーとしてのログオンを求めるプロンプトが表示されます。

---

---



Business Event Systemデモンストレーション - Netscape

ファイル(F) 編集(E) 表示(V) ジャンプ(G) Communicator(C) ヘルプ(H)

Buyer Workbench

ORACLE

このデモンストレーションでは、発注書がBuyer Systemに入力されます。送信後、PO XML文書はSupplier Systemと呼ばれる別のシステムに伝播されます。Supplier Systemでは、Buyer Systemに対して、次の内容のメッセージを返信します。1)発注の確認 2)納品日の確認(ASN) 3)請求書の送信

発注を入力してください:

オーダー番号

項目番号

項目の説明

納品日

合計金額

申請者

ドキュメント: 完了。

2. 発注に対して一意のオーダー番号を入力します。オーダー番号は、呼び出される発注イベントのイベント・キーになります。
3. 項目番号を選択します。
4. 項目の説明を半角英数字 40 文字以内で入力します。
5. 納品日を「DD-MON-YYYY」の形式で入力します。
6. 発注の合計金額を入力します。この金額は書式なしの数値として入力する必要があります。
7. オーダー・リクエストのロール名を選択します。
8. 「送信」ボタンを選択して発注を送信し、「イベント: オーダーの追跡」画面を表示します。「取消」ボタンを選択し、発注を送信しないで「ワークフロー・デモンストレーション」画面に戻ることができます。



9. 「イベント: オーダーの追跡」画面で、「キュー・メッセージ」リンクを選択し、「イベント・システム・ローカル・キュー」画面を表示して、ビジネス・イベント・システムのキューにあるメッセージを確認します。13-79 ページの「ローカル・キューの確認」を参照してください。
10. 「ワークフロー・モニター」リンクを選択し、ワークフロー・モニターでバイヤー: 最上位の発注ワークフロー・プロセスの進行状況を確認します。11-2 ページの「ワークフロー・モニター」を参照してください。
11. 「プロセス・オーダー」リンクを選択し、バックグラウンド・エンジンを実行して、バイヤー・ワークフローが発注の処理を継続できるように遅延アクティビティを処理します。

## ▶ サプライヤ・システムでのイベント・システム・デモンストレーション・ワークフローの継続

1. サプライヤ・システムで、Web ブラウザに次の URL を入力して「ワークフロー・デモンストレーション」Web 画面にアクセスします。

`<webagent>/wf_demo.home`

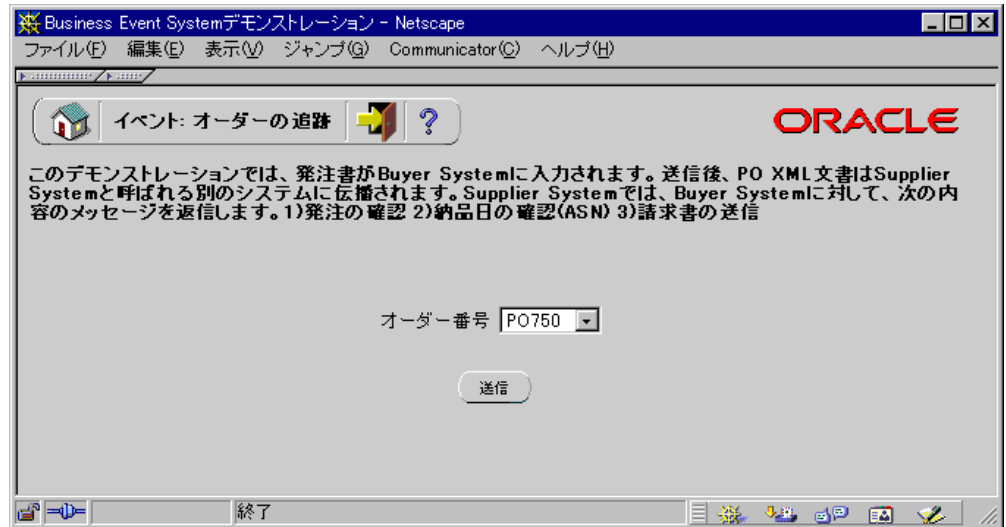
`<webagent>` は、Web サーバーで Oracle Workflow 用に構成された Web エージェントのベース URL を表します。2-13 ページの「グローバル・ワークフロー・プリファレンスの設定」を参照してください。

「イベント: オーダーの追跡」リンクを選択して、「イベント: オーダーの追跡」Web 画面を表示します。

または、次の URL を入力して「Buyer Workbench」Web 画面を直接表示することもできます。

`<webagent>/wf_eventdemo.track_order`

**注意：** どちらのページにもセキュリティが適用されるため、現行 Web セッションで有効なワークフロー・ユーザーとしてログオンしていない場合は、ページが表示される前に有効なワークフロー・ユーザーとしてのログオンを求めるプロンプトが表示されます。



2. 発注のオーダー番号を選択して、「送信」ボタンを選択します。



3. 表示される「イベント:オーダーの追跡」画面で、「キュー・メッセージ」リンクを選択し、「イベント・システム・ローカル・キュー」画面を表示して、ビジネス・イベント・システムのキューにあるメッセージを確認します。13-79 ページの「[ローカル・キューの確認](#)」を参照してください。
4. 「ワークフロー・モニター」リンクを選択し、ワークフロー・モニターでサプライヤ:最上位のオーダー・プロセスの進行状況を確認します。11-2 ページの「[ワークフロー・モニター](#)」を参照してください。
5. 「プロセス・オーダー」リンクを選択し、バックグラウンド・エンジンを実行して、発注の処理を続けます。この手順を 2 回実行して、サプライヤ・ワークフローの両方の遅延アクティビティを処理します。

## 「イベント・システムのデモ」項目タイプ

イベント・システム・デモンストレーションは、「イベント・システムのデモ」という項目タイプに関連付けられています。現在、次の 11 のワークフロー・プロセスがイベント・システム・デモンストレーションに関連付けられています。

- バイヤー:最上位の発注
- バイヤー:サプライヤへの発注送信
- バイヤー:サプライヤの発注承認の受信
- バイヤー:アドバンスト出荷通知
- バイヤー:サプライヤの請求の受信
- サプライヤ:最上位のオーダー
- サプライヤ:オーダー詳細の取得
- サプライヤ:クレジット・チェック
- サプライヤ:在庫チェック
- サプライヤ:アドバンスト出荷通知
- サプライヤ:サプライヤの請求書の送信

「イベント・システムのデモ」項目タイプの詳細を Workflow Builder に表示するには、「ファイル」メニューから「オープン」を選択します。次に、データベースに接続して、「イベント・システムのデモ」項目タイプを選択するか、ファイル・システムの <ORACLE\_HOME>/wf/Data/<language> サブディレクトリにある wfefdeme.wft というファイルに接続します。

「イベント・システムのデモ」のプロパティ画面を見ると、維持タイプが「一時」に、維持日数が 0 になっています。これは、この項目タイプの作業項目に関連付けられているランタイム・データは、終了直後に削除の対象となることを意味します。

「イベント・システムのデモ」項目タイプには、複数の属性も関連付けられています。これらの属性は、ワークフローのアプリケーション表にある情報を参照します。属性は、プロセス全体を通して関数アクティビティ、通知アクティビティおよびイベント・アクティビティによって使用され保存されます。次の表は、「イベント・システムのデモ」項目タイプの属性を示しています。

表 15-6

表示名	説明	タイプ	長さ/書式/選択肢 タイプ/デフォルト 値
イベント名	元のイベントの内部名	テキスト	

表 15-6 (続き)

表示名	説明	タイプ	長さ / 書式 / 選択肢 タイプ / デフォルト 値
イベント・キー	元のイベントの特定のインスタンスを一意に識別するイベント・キー	テキスト	
イベント・メッセージ	元のイベントのイベント・メッセージ	イベント	
発注状況	発注の状況	テキスト	CREATED
発注承認イベント	発注承認イベントの名前	テキスト	
発注承認イベント・キー	発注承認イベントの特定のインスタンスを一意に識別するイベント・キー	テキスト	
発注 ASN イベント	アドバンスト出荷通知イベントの名前	テキスト	
発注 ASN イベント・キー	アドバンスト出荷通知イベントの特定のインスタンスを一意に識別するイベント・キー	テキスト	
発注 ASN イベント・メッセージ	アドバンスト出荷通知イベント・メッセージ	イベント	
オーダー・リクエスト	オーダーをリクエストしたユーザーの名前	テキスト	BLEWIS
宛先エージェント / システム 1	イベント・メッセージを受信するバイヤー・システム上のインバウンド・エージェント (書式は <agent>@<system>)	テキスト	
送信元エージェント / システム 1	イベント・メッセージを送信するバイヤー・システム上のアウトバウンド・エージェント (書式は <agent>@<system>)	テキスト	
宛先エージェント / システム 2	イベント・メッセージを受信するサプライヤ・システム上のインバウンド・エージェント (書式は <agent>@<system>)	テキスト	
送信元エージェント / システム 2	イベント・メッセージを送信するサプライヤ・システム上のアウトバウンド・エージェント (書式は <agent>@<system>)	テキスト	
項目番号	項目番号	テキスト	

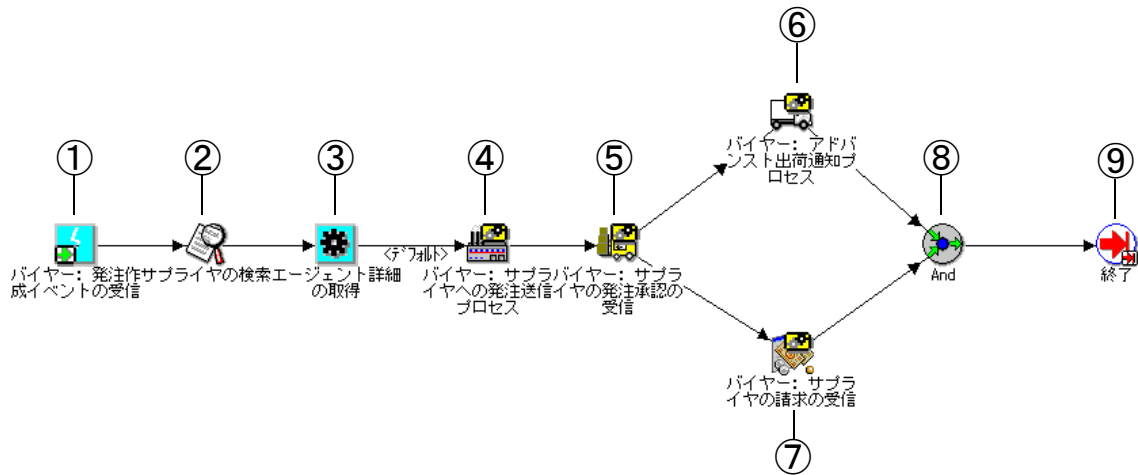
表 15-6（続き）

表示名	説明	タイプ	長さ / 書式 / 選択肢 タイプ / デフォルト 値
項目の説明	項目の説明	テキスト	
サブスクリプション GUID	サブスクリプションのグローバル 一意識別子	テキスト	

バイヤー：最上位の発注プロセスの概要

バイヤー：最上位の発注プロセスのプロパティを表示するには、ナビゲータ・ツリーでプロセスを選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは実行可能です。つまり、最上位レベルのプロセスとして実行できます。

バイヤー：最上位の発注プロセスの「プロセス」ウィンドウを表示すると、プロセスが9つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



バイヤー：最上位の発注ワークフローは、「Buyer Workbench」デモンストレーション Web 画面から発注を送信して、発注イベントを呼び出すと開始します。事前定義のサブスクリプションにより、接頭辞 PO とイベント・キー（オーダー番号）で構成される関連 ID が追加され、イベント・メッセージがバイヤー：最上位の発注プロセスに送信されます。15-66 ページの「Buyer Workbench からのイベント・システム・デモンストレーション・ワークフローの開始」を参照してください。

発注プロセスは、ノード1の「バイヤー:発注作成イベントの受信」アクティビティから開始します。この発注のサプライヤを検索し、目的のアウトバウンド・エージェントおよびインバウンド・エージェントのエージェント詳細を取得したら、発注をサプライヤに送信します。

ノード4は、発注イベント・メッセージからオーダー・リクエストの名前を取得し、発注をサプライヤに送信して、発注がサプライヤに送信されたことをオーダー・リクエストに通知するサブプロセスです。

ノード5は、サプライヤからの発注承認イベント・メッセージを待機するサブプロセスです。発注承認が指定した期間内に着信しない場合は、タイムアウト・アクティビティを実行し、発注承認を受信するまで、サプライヤが応答しないことをオーダー・リクエストに繰り返し通知します。承認が着信すると、サブプロセスはオーダー・リクエストに承認を通知します。

発注承認が着信すると、最上位プロセスは、サプライヤからのアドバンスト出荷通知および請求書を待機します。ノード6は、アドバンスト出荷通知を受信し、オーダーが出荷されたことをオーダー・リクエストに通知するサブプロセスです。ノード7は、請求書を受信し、そのことをオーダー・リクエストに通知するサブプロセスです。これらのサブプロセスが完了すると、プロセスは終了します。

バイヤー:最上位の発注プロセスのアクティビティ

ここでは、プロセスの各アクティビティについて、アクティビティの表示名別に説明します。

バイヤー:発注作成イベントの受信 (ノード1)

このイベント・アクティビティは、イベント・マネージャによってバイヤー:最上位の発注プロセスに送信された発注イベント・メッセージを受信して、新しい項目を開始します。

イベント・アクション	受信
イベント・フィルタ	demo.oracle.wf.b2b.po.create
前提条件アクティビティ	なし
アクティビティによって設定される項目属性	イベント名、イベント・キー、イベント・メッセージ

サプライヤの検索 (ノード2)

現在、このアクティビティでは何も処理されません。このアクティビティに対して、発注されている項目などの条件に基づいて、発注のサプライヤを検索する関数を統合できます。

関数	WF_EVENTDEMO.FINDSUPPLIER
結果タイプ	なし



前提条件アクティビティ バイヤー：発注作成イベントの受信

### エージェント詳細の取得（ノード3）

この関数アクティビティは、イベント・メッセージをワークフロー・プロセスに送信するサブスクリプションから、エージェント詳細を取得します。このアクティビティは、メッセージを送信するバイヤー・システム上のアウトバウンド・エージェントおよびメッセージを受信するサプライヤ・システム上のインバウンド・エージェントの、エージェント詳細を取得します。

関数	WF_STANDARD.GETAGENTS
結果タイプ	ブール
前提条件アクティビティ	サプライヤの検索
関数によって取得される項目属性	サブスクリプション GUID
関数によって設定される項目属性	送信元エージェント / システム 1、宛先エージェント / システム 2

### バイヤー：サプライヤへの発注送信プロセス（ノード4）

このアクティビティは、発注イベント・メッセージからオーダー・リクエストの名前を取得し、発注をサプライヤに送信して、発注がサプライヤに送信されたことをオーダー・リクエストに通知するサブプロセスです。このサブプロセスを参照するには、ナビゲータ・ツリーで「プロセス」のブランチの下にある「バイヤー：サプライヤへの発注書の送信」をダブルクリックします。15-77 ページの「[バイヤー：サプライヤへの発注送信サブプロセスの概要](#)」を参照してください。

結果タイプ	なし
前提条件アクティビティ	エージェント詳細の取得

### バイヤー：サプライヤの発注承認の受信（ノード5）

このアクティビティは、サプライヤから発注承認イベント・メッセージを待機するサブプロセスです。発注承認が指定した期間内に着信しない場合は、タイムアウト・アクティビティを実行し、発注承認を受信するまで、サプライヤが応答しないことをオーダー・リクエストに繰り返し通知します。承認が着信すると、サブプロセスはオーダー・リクエストに承認を通知します。

このサブプロセスを参照するには、ナビゲータ・ツリーで「プロセス」のブランチの下にある「バイヤー：サプライヤの発注承認の受信」をダブルクリックします。15-79 ページの「[バイヤー：サプライヤの発注承認の受信サブプロセスの概要](#)」を参照してください。

結果タイプ                      なし  
 前提条件アクティビティ    バイヤー：サプライヤへの発注送信

### バイヤー：アドバンスト出荷通知プロセス（ノード 6）

このアクティビティは、サプライヤからアドバンスト出荷通知を受信し、オーダーが出荷されたことをオーダー・リクエストに通知するサブプロセスです。このサブプロセスを表示するには、ナビゲータ・ツリーの「プロセス」のブランチの下にある「バイヤー：アドバンスト出荷通知」をダブルクリックします。15-82 ページの「[バイヤー：アドバンスト出荷通知サブプロセスの概要](#)」を参照してください。

結果タイプ                      なし  
 前提条件アクティビティ    バイヤー：サプライヤの発注承認の受信

### バイヤー：サプライヤの請求の受信（ノード 7）

このアクティビティは、サプライヤから請求書を受信し、請求書が届いたことをオーダー・リクエストに通知するサブプロセスです。このサブプロセスを参照するには、ナビゲータ・ツリーで「プロセス」のブランチの下にある「バイヤー：サプライヤの請求の受信」をダブルクリックします。15-84 ページの「[バイヤー：サプライヤの請求の受信サブプロセスの概要](#)」を参照してください。

結果タイプ                      なし  
 前提条件アクティビティ    バイヤー：サプライヤの発注承認の受信

### And（ノード 8）

この「標準」関数アクティビティでは、フロー内の並行する 2 つ以上の分岐のすべてのアクティビティが完了した場合にのみ、分岐がマージされます。

関数                                  WF\_STANDARD.ANDJOIN  
 結果タイプ                      なし  
 前提条件アクティビティ    バイヤー：アドバンスト出荷通知、バイヤー：サプライヤの請求の受信

### 終了（ノード 9）

この「標準」関数アクティビティは、プロセスの終了をマークします。

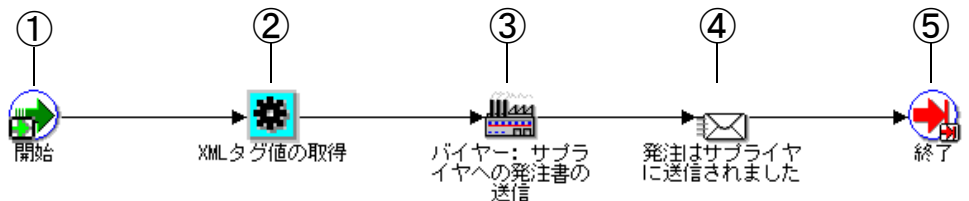
関数                                  WF\_STANDARD.NOOP  
 結果タイプ                      なし

## 前提条件アクティビティ And

## バイヤー：サプライヤへの発注送信サブプロセスの概要

バイヤー：サプライヤへの発注送信サブプロセスのプロパティを表示するには、そのプロセス・アクティビティをナビゲータ・ツリーで選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは、実行可能ではありません。つまり、最上位レベルのプロセスとして開始できず、上位レベルのプロセスからコールする必要があります。

バイヤー：サプライヤへの発注送信サブプロセスの「プロセス」ウィンドウを表示すると、サブプロセスが5つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



このサブプロセスは、ノード1の「開始」アクティビティから開始します。ノード2で、発注イベント・メッセージからオーダー・リクエストの名前を取得します。次に、発注をサプライヤに送信し、発注が送信されたことをオーダー・リクエストに通知します。この時点でサブプロセスは終了します。

## バイヤー：サプライヤへの発注送信サブプロセスのアクティビティ

ここでは、サブプロセスの各アクティビティについて、アクティビティの表示名別に説明します。

### 開始（ノード 1）

この「標準」関数アクティビティは、プロセスの開始をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	なし

### XML タグ値の取得（ノード 2）

この「標準」外部 Java 関数アクティビティは、イベント・メッセージに設定されている特定の XML タグの値を取得します。このノードで、プロセスは発注イベント・メッセージからオーダー・リクエストの名前を取得します。

関数	oracle.apps.fnd.wf.XMLGetTagValue
結果タイプ	なし
前提条件アクティビティ	開始
関数によって取得される項目属性	イベント・メッセージ
関数によって設定される項目属性	オーダー・リクエスト

### バイヤー：サプライヤへの発注書の送信（ノード 3）

このイベント・アクティビティは、バイヤー・システム上のアウトバウンド・エージェントおよびサプライヤ・システム上のインバウンド・エージェントを使用して、発注イベント・メッセージをサプライヤに送信します。これらのエージェントは、バイヤー：最上位の発注プロセスの「エージェント詳細の取得」アクティビティによって識別されます。

イベント・アクション	送信
前提条件アクティビティ	XML タグ値の取得
アクティビティによって取得される項目属性	イベント・メッセージ、送信元エージェント / システム 1、宛先エージェント / システム 2

## 発注はサプライヤに送信されました（ノード4）

このアクティビティは、発注がサプライヤに送信されたことをオーダー・リクエストに通知します。メッセージには、通知の送信時に発注番号を表示する「送信」属性が含まれます。

このアクティビティ・ノードのプロパティ画面を表示すると、「オーダー・リクエスト」という項目タイプ属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。

メッセージ	発注はサプライヤに送信されました
結果タイプ	なし
前提条件アクティビティ	バイヤー：サプライヤへの発注送信

## 終了（ノード5）

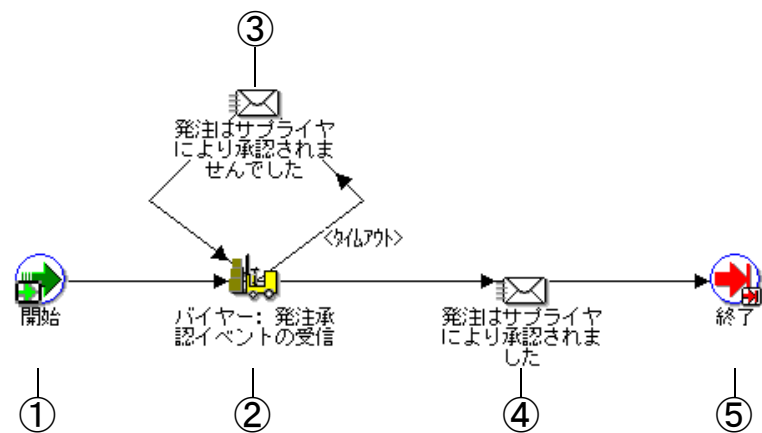
この「標準」関数アクティビティは、プロセスの終了をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	発注はサプライヤに送信されました

## バイヤー：サプライヤの発注承認の受信サブプロセスの概要

バイヤー：サプライヤの発注承認の受信サブプロセスのプロパティを表示するには、そのプロセス・アクティビティをナビゲータ・ツリーで選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは、実行可能ではありません。つまり、最上位レベルのプロセスとして開始できず、上位レベルのプロセスからコールする必要があります。

バイヤー：サプライヤの発注承認の受信サブプロセスの「プロセス」ウィンドウを表示すると、サブプロセスが5つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



このサブプロセスは、ノード1の「開始」アクティビティから開始します。ノード2で、サプライヤから発注承認イベント・メッセージを待機します。発注承認が指定した期間内に着信しない場合は、タイムアウト・アクティビティを実行し、発注承認を受信するまで、サプライヤが応答しないことをオーダー・リクエストに繰り返し通知します。承認が着信すると、サブプロセスはオーダー・リクエストに承認を通知します。この時点でサブプロセスは終了します。

## バイヤー：サプライヤの発注承認の受信サブプロセスのアクティビティ

ここでは、サブプロセスの各アクティビティについて、アクティビティの表示名別に説明します。

### 開始（ノード1）

この「標準」関数アクティビティは、プロセスの開始をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	なし

### バイヤー：発注承認イベントの受信（ノード2）

このイベント・アクティビティは、サプライヤから発注承認イベント・メッセージを待機します。このアクティビティは1日以内に完了しないとタイムアウトになります。

イベント・アクション	受信
イベント・フィルタ	demo.oracle.wf.b2b.po.ack

前提条件アクティビティ 開始

アクティビティによって イベント名、イベント・キー、イベント・メッセージ  
設定される項目属性

### 発注はサプライヤにより承認されませんでした（ノード3）

このアクティビティは、「バイヤー：発注承認イベントの受信」アクティビティがタイムアウトになり、完了しなかった場合にのみ発生します。このアクティビティは、サプライヤがまだ発注を承認していないことをオーダー・リクエストに通知します。メッセージには、通知の送信時に発注番号を表示する「送信」属性が含まれます。

このアクティビティ・ノードのプロパティ画面を表示すると、「オーダー・リクエスト」という項目タイプ属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。

メッセージ 発注未承認

結果タイプ なし

前提条件アクティビティ バイヤー：発注承認イベントの受信

### 発注はサプライヤにより承認されました（ノード4）

このアクティビティは、発注がサプライヤによって承認されたことをオーダー・リクエストに通知します。メッセージには、通知の送信時に発注番号を表示する「送信」属性が含まれます。

このアクティビティ・ノードのプロパティ画面を表示すると、「オーダー・リクエスト」という項目タイプ属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。

メッセージ 発注承認済

結果タイプ なし

前提条件アクティビティ バイヤー：発注承認イベントの受信

### 終了（ノード5）

この「標準」関数アクティビティは、プロセスの終了をマークします。

関数 WF\_STANDARD.NOOP

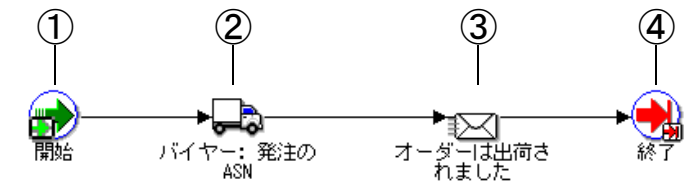
結果タイプ なし

前提条件アクティビティ 発注はサプライヤにより承認されました

## バイヤー：アドバンスト出荷通知サブプロセスの概要

バイヤー：アドバンスト出荷通知サブプロセスのプロパティを表示するには、そのプロセス・アクティビティをナビゲータ・ツリーで選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは、実行可能ではありません。つまり、最上位レベルのプロセスとして開始できず、上位レベルのプロセスからコールする必要があります。

バイヤー：アドバンスト出荷通知サブプロセスの「プロセス」ウィンドウを表示すると、サブプロセスが4つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



このサブプロセスは、ノード1の「開始」アクティビティから開始します。ノード2で、サプライヤからアドバンスト出荷通知を待機します。アドバンスト出荷通知が着信すると、オーダーが出荷されたことをオーダー・リクエストに通知します。この時点でサブプロセスは終了します。

## バイヤー：アドバンスト出荷通知サブプロセスのアクティビティ

ここでは、サブプロセスの各アクティビティについて、アクティビティの表示名別に説明します。

### 開始（ノード1）

この「標準」関数アクティビティは、プロセスの開始をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	なし

### バイヤー：発注のASN（ノード2）

このイベント・アクティビティは、サプライヤからアドバンスト出荷通知イベント・メッセージを待機します。

イベント・アクション	受信
------------	----



イベント・フィルタ demo.oracle.wf.b2b.po.asn

前提条件アクティビティ 開始

アクティビティによって イベント名、イベント・キー、イベント・メッセージ  
設定される項目属性

### オーダーは出荷されました（ノード3）

このアクティビティは、アドバンスト出荷通知が着信したことをオーダー・リクエストに通知します。つまり、オーダーがサプライヤから出荷されたことを通知します。メッセージには、通知の送信時に発注番号を表示する「送信」属性が含まれます。

このアクティビティ・ノードのプロパティ画面を表示すると、「オーダー・リクエスト」という項目タイプ属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。

メッセージ 発注のアドバンスト出荷通知

結果タイプ なし

前提条件アクティビティ バイヤー：発注の ASN

### 終了（ノード4）

この「標準」関数アクティビティは、プロセスの終了をマークします。

関数 WF\_STANDARD.NOOP

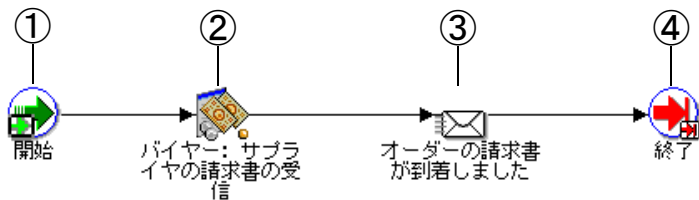
結果タイプ なし

前提条件アクティビティ オーダーは出荷されました

## バイヤー：サプライヤの請求の受信サブプロセスの概要

バイヤー：サプライヤの請求の受信サブプロセスのプロパティを表示するには、そのプロセス・アクティビティをナビゲータ・ツリーで選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは、実行可能ではありません。つまり、最上位レベルのプロセスとして開始できず、上位レベルのプロセスからコールする必要があります。

バイヤー：サプライヤの請求の受信サブプロセスの「プロセス」ウィンドウを表示すると、サブプロセスが4つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



このサブプロセスは、ノード1の「開始」アクティビティから開始します。ノード2で、サプライヤから請求書を待機します。請求書が着信したときに、請求が届いたことをオーダー・リクエストに通知します。この時点でサブプロセスは終了します。

## バイヤー：サプライヤの請求の受信サブプロセスのアクティビティ

ここでは、サブプロセスの各アクティビティについて、アクティビティの表示名別に説明します。

### 開始（ノード1）

この「標準」関数アクティビティは、プロセスの開始をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	なし

### バイヤー：サプライヤの請求書の受信（ノード2）

このイベント・アクティビティは、サプライヤから請求書イベント・メッセージを待機します。

イベント・アクション	受信
------------	----

イベント・フィルタ      demo.oracle.wf.b2b.po.invoice

前提条件アクティビティ    開始

アクティビティによって    イベント名、イベント・キー、イベント・メッセージ  
設定される項目属性

### オーダーの請求書が到着しました（ノード3）

このアクティビティは、サプライヤから発注の請求書が着信したことをオーダー・リクエストに通知します。メッセージには、通知の送信時に発注番号を表示する「送信」属性が含まれます。

このアクティビティ・ノードのプロパティ画面を表示すると、「オーダー・リクエスト」という項目タイプ属性に名前が保存されている実行者に、このアクティビティが割り当てられていることがわかります。

メッセージ                  サプライヤ発注請求書

結果タイプ                  なし

前提条件アクティビティ    バイヤー：サプライヤの請求書の受信

### 終了（ノード4）

この「標準」関数アクティビティは、プロセスの終了をマークします。

関数                          WF\_STANDARD.NOOP

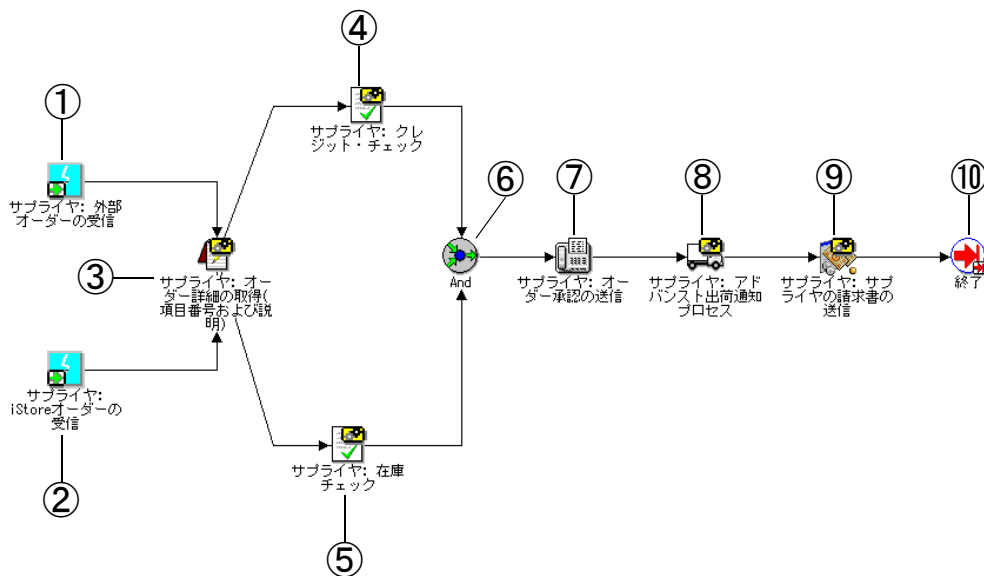
結果タイプ                  なし

前提条件アクティビティ    オーダーの請求書が到着しました

## サプライヤ:最上位のオーダー・プロセスの概要

サプライヤ:最上位のオーダー・プロセスのプロパティを表示するには、ナビゲータ・ツリーでプロセスを選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは実行可能です。つまり、最上位レベルのプロセスとして実行できます。

サプライヤ:最上位のオーダー・プロセスの「プロセス」ウィンドウを表示すると、プロセスが 10 の異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



サプライヤ:最上位のオーダー・ワークフローは、サプライヤ・システムがバイヤー・システムから発注イベントを受信したときに開始します。事前定義のサブスクリプションにより、相関 ID の構成が接頭辞 PO とイベント・キー（オーダー番号）に変更され、イベントがサプライヤ:最上位のオーダー・プロセスに送信されます。

このプロセスは、ノード 1 またはノード 2 に発注が着信したときに開始します。着信先は、発注のソースによって異なります。

ノード 3 は、項目番号および項目の説明とともに、目的のアウトバウンドおよびインバウンド・エージェントのエージェント詳細を取得して、応答をバイヤーに送信するサブプロセスです。

ノード 4 は、オーダーの合計コストが \$2000 を超える場合にクレジット・チェックを実行するサブプロセスです。ノード 5 は、在庫チェックを実行して、オーダー項目の在庫を確認す

るサブプロセスです。クレジット・チェックと在庫チェックが完了すると、発注の承認をバイヤーに送信します。

ノード8は、オーダーの出荷詳細をチェックして、アドバンスト出荷通知をバイヤーに送信するサブプロセスです。ノード9は、請求書を作成し、請求書をバイヤーに送信するサブプロセスです。請求書の送信後に、プロセスは終了します。

## サプライヤ: 最上位のオーダー・プロセスのアクティビティ

ここでは、プロセスの各アクティビティについて、アクティビティの表示名別に説明します。

### サプライヤ: 外部オーダーの受信 (ノード1)

このイベント・アクティビティは、発注イベント・メッセージを受信します。このメッセージは、イベント・マネージャがバイヤーからイベントを受信したときに、サプライヤ: 最上位のオーダー・プロセスに送信されます。

イベント・アクション 受信

イベント・フィルタ demo.oracle.wf.b2b.po.create

前提条件アクティビティ なし

アクティビティによって イベント名、イベント・キー、イベント・メッセージ  
設定される項目属性

### サプライヤ: iStore オーダーの受信 (ノード2)

このイベント・アクティビティは、サプライヤの iStore アプリケーションによってオーダーが作成されたときに送信される、発注イベント・メッセージを受信します。現在、このイベントは定義されていません。このアクティビティに対して、別のソースから受信した発注を統合できます。

イベント・アクション 受信

イベント・フィルタ demo.oracle.wf.istore.po.create

前提条件アクティビティ なし

アクティビティによって イベント名、イベント・キー、イベント・メッセージ  
設定される項目属性

### サプライヤ: オーダー詳細の取得 (ノード3)

このアクティビティは、項目番号およびオーダー項目の説明とともに、目的のアウトバウンドおよびインバウンド・エージェントのエージェント詳細を取得して、応答をバイヤーに送信するサブプロセスです。

このサブプロセスを参照するには、ナビゲータ・ツリーで「プロセス」のブランチの下にある「サプライヤ: オーダー詳細の取得」をダブルクリックします。15-90 ページの「[サプライヤ: オーダー詳細の取得サブプロセスの概要](#)」を参照してください。

結果タイプ                      なし

前提条件アクティビティ    サプライヤ: 外部オーダーの受信またはサプライヤ: iStore オーダーの受信

### サプライヤ: クレジット・チェック (ノード 4)

このアクティビティは、オーダーの合計コストが \$2000 を超える場合にクレジット・チェックを実行するサブプロセスです。

このサブプロセスを表示するには、ナビゲータ・ツリーの「プロセス」のブランチの下にある「サプライヤ: クレジット・チェック」をダブルクリックします。15-93 ページの「[サプライヤ: クレジット・チェック・サブプロセスの概要](#)」を参照してください。

結果タイプ                      なし

前提条件アクティビティ    サプライヤ: オーダー詳細の取得

### サプライヤ: 在庫チェック (ノード 5)

このアクティビティは、在庫チェックを実行して、オーダー項目の在庫を確認するサブプロセスです。

このサブプロセスを表示するには、ナビゲータ・ツリーの「プロセス」のブランチの下にある「サプライヤ: 在庫チェック」をダブルクリックします。15-95 ページの「[サプライヤ: 在庫チェック・サブプロセスの概要](#)」を参照してください。

結果タイプ                      なし

前提条件アクティビティ    サプライヤ: オーダー詳細の取得

### And (ノード 6)

この「標準」関数アクティビティでは、フロー内の並行する 2 つ以上の分岐のすべてのアクティビティが完了した場合にのみ、分岐がマージされます。

関数                              WF\_STANDARD.ANDJOIN

結果タイプ                      なし

前提条件アクティビティ    サプライヤ: クレジット・チェック、サプライヤ: 在庫チェック

## サプライヤ: オーダー承認の送信 (ノード 7)

このイベント・アクティビティは、サプライヤ・システム上のアウトバウンド・エージェントおよびバイヤー・システム上のインバウンド・エージェントを使用して、発注承認イベント・メッセージをバイヤーに送信します。これらのエージェントは、サプライヤ: オーダー詳細の取得サブプロセスの「エージェント詳細の取得」アクティビティによって識別されます。

イベント・アクション      送信

前提条件アクティビティ    And

アクティビティによって    イベント・メッセージ、送信元エージェント / システム 2、宛  
取得される項目属性      先エージェント / システム 1

## サプライヤ: アドバンスト出荷通知プロセス (ノード 8)

このアクティビティは、オーダーの出荷詳細をチェックして、アドバンスト出荷通知をバイヤーに送信するサブプロセスです。

このサブプロセスを表示するには、ナビゲータ・ツリーの「プロセス」のブランチの下にある「サプライヤ: アドバンスト出荷通知」をダブルクリックします。15-96 ページの「[サプライヤ: アドバンスト出荷通知サブプロセスの概要](#)」を参照してください。

結果タイプ                  なし

前提条件アクティビティ    サプライヤ: オーダー承認の送信

## サプライヤ: サプライヤの請求書の送信 (ノード 9)

このアクティビティは、発注の請求書を作成し、請求書をバイヤーに送信するサブプロセスです。

このサブプロセスを参照するには、ナビゲータ・ツリーで「プロセス」のブランチの下にある「サプライヤ: サプライヤの請求書の送信」をダブルクリックします。15-98 ページの「[サプライヤ: サプライヤの請求書の送信サブプロセスの概要](#)」を参照してください。

結果タイプ                  なし

前提条件アクティビティ    サプライヤ: アドバンスト出荷通知

## 終了 (ノード 10)

この「標準」関数アクティビティは、プロセスの終了をマークします。

関数                          WF\_STANDARD.NOOP

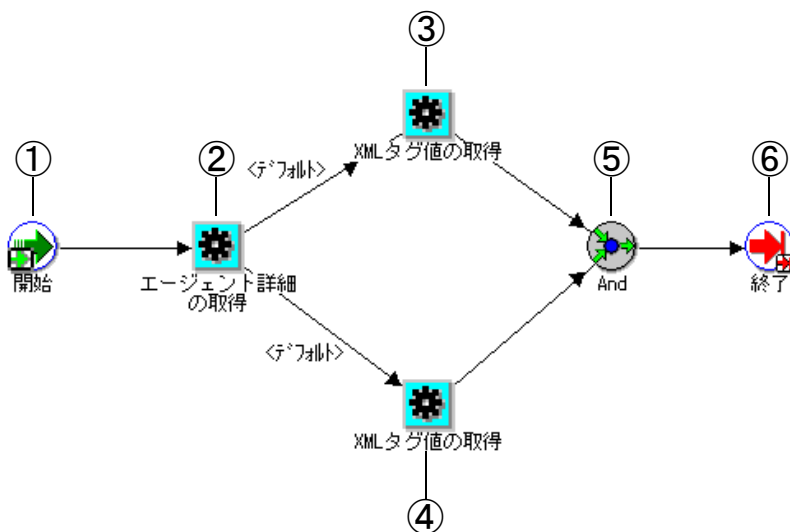
結果タイプ                  なし

前提条件アクティビティ サプライヤ: サプライヤの請求書の送信

## サプライヤ: オーダー詳細の取得サブプロセスの概要

サプライヤ: オーダー詳細の取得サブプロセスのプロパティを表示するには、そのプロセス・アクティビティをナビゲータ・ツリーで選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは、実行可能ではありません。つまり、最上位レベルのプロセスとして開始できず、上位レベルのプロセスからコールする必要があります。

サプライヤ: オーダー詳細の取得サブプロセスの「プロセス」ウィンドウを表示すると、サブプロセスが5つの異なるアクティビティで構成されていることがわかります。そのうちの1つが再利用され、ワークフロー・ダイアグラムに表示される6つのアクティビティ・ノードを構成しています。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



このサブプロセスは、ノード1の「開始」アクティビティから開始します。ノード2で、目的のアウトバウンド・エージェントおよびインバウンド・エージェントのエージェント詳細を取得して、応答をバイヤーに送信します。次に、オーダー項目の項目番号と項目の説明を発注から取得します。項目番号と項目の説明の取得が完了すると、サブプロセスは終了します。



## サプライヤ: オーダー詳細の取得サブプロセスのアクティビティ

ここでは、サブプロセスの各アクティビティについて、アクティビティの表示名別に説明します。

### 開始（ノード 1）

この「標準」関数アクティビティは、プロセスの開始をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	なし

### エージェント詳細の取得（ノード 2）

この関数アクティビティは、イベント・メッセージをワークフロー・プロセスに送信するサブスクリプションから、エージェント詳細を取得します。このアクティビティは、応答メッセージを送信するサプライヤ・システム上のアウトバウンド・エージェントおよび応答メッセージを受信するパイヤー・システム上のインバウンド・エージェントの、詳細を取得します。

関数	WF_STANDARD.GETAGENTS
結果タイプ	ブール
前提条件アクティビティ	開始
関数によって取得される項目属性	サブスクリプション GUID
関数によって設定される項目属性	送信元エージェント / システム 2、宛先エージェント / システム 1

### XML タグ値の取得（ノード 3）

この「標準」外部 Java 関数アクティビティは、イベント・メッセージに設定されている特定の XML タグの値を取得します。このノードで、プロセスは発注イベント・メッセージから項目番号を取得します。

関数	oracle.apps.fnd.wf.XMLGetTagValue
結果タイプ	なし
前提条件アクティビティ	エージェント詳細の取得
関数によって取得される項目属性	イベント・メッセージ

関数によって設定される 項目番号  
項目属性

**XML タグ値の取得（ノード 4）**

この「標準」外部 Java 関数アクティビティは、イベント・メッセージに設定されている特定の XML タグの値を取得します。このノードで、発注イベント・メッセージから項目の説明を取得します。

関数 oracle.apps.fnd.wf.XMLGetTagValue  
結果タイプ なし  
前提条件アクティビティ エージェント詳細の取得  
関数によって取得される イベント・メッセージ  
項目属性  
関数によって設定される 項目の説明  
項目属性

**And（ノード 5）**

この「標準」関数アクティビティでは、フロー内の並行する 2 つ以上の分岐のすべてのアクティビティが完了した場合にのみ、分岐がマージされます。

関数 WF\_STANDARD.ANDJOIN  
結果タイプ なし  
前提条件アクティビティ XML タグ値の取得（ノード 3）、XML タグ値の取得（ノード 4）

**終了（ノード 6）**

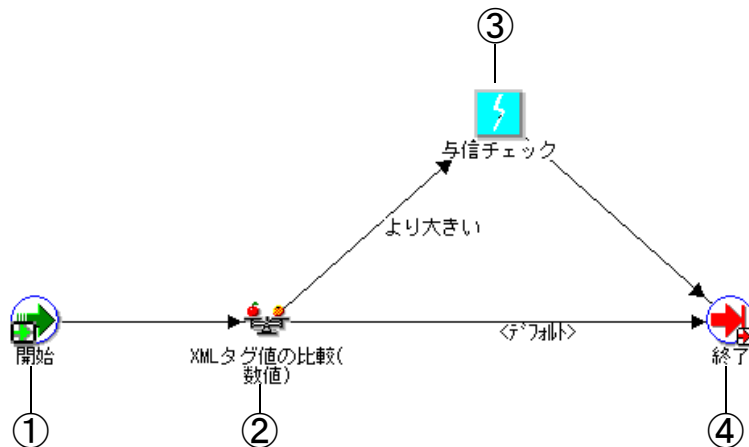
この「標準」関数アクティビティは、プロセスの終了をマークします。

関数 WF\_STANDARD.NOOP  
結果タイプ なし  
前提条件アクティビティ And

## サプライヤ:クレジット・チェック・サブプロセスの概要

サプライヤ:クレジット・チェック・サブプロセスのプロパティを表示するには、そのプロセス・アクティビティをナビゲータ・ツリーで選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは、実行可能ではありません。つまり、最上位レベルのプロセスとして開始できず、上位レベルのプロセスからコールする必要があります。

サプライヤ:クレジット・チェック・サブプロセスの「プロセス」ウィンドウを表示すると、サブプロセスが4つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



このサブプロセスは、ノード1の「開始」アクティビティから開始します。ノード2で、オーダーの合計コストが\$2000を超えるかどうかをチェックします。合計コストが\$2000以下の場合、プロセスはこの時点で終了します。合計コストが\$2000を超える場合、バイヤーに対してクレジット・チェックを実行した後でプロセスは終了します。

## サプライヤ:クレジット・チェック・サブプロセスのアクティビティ

ここでは、サブプロセスの各アクティビティについて、アクティビティの表示名別に説明します。

### 開始（ノード 1）

この「標準」関数アクティビティは、プロセスの開始をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	なし

### XML タグ値の比較（数値）（ノード 2）

この「標準」外部 Java 関数アクティビティは、イベント・メッセージに設定されている特定の XML タグの値とテスト値を比較します。このノードで、発注イベント・メッセージの合計コストとテスト値 \$2000 を比較します。

関数	oracle.apps.fnd.wf.XMLCompareTag
結果タイプ	比較
前提条件アクティビティ	開始
関数によって取得される項目属性	イベント・メッセージ

### 与信チェック（ノード 3）

このアクティビティは、発注の合計コストが \$2000 を超える場合にのみ発生します。このイベント・アクティビティは、クレジット・チェック・イベントを呼び出します。現在、このイベントは定義されていません。このアクティビティに対して、クレジット・チェックを続けます。

イベント・アクション	呼出し
前提条件アクティビティ	XML タグ値の比較（数値）
アクティビティによって取得される項目属性	イベント・キー

### 終了（ノード 4）

この「標準」関数アクティビティは、プロセスの終了をマークします。

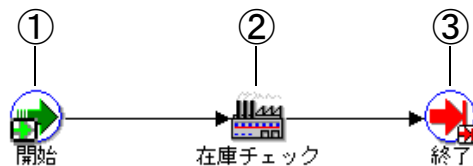
関数	WF_STANDARD.NOOP
----	------------------

結果タイプ	なし
前提条件アクティビティ	XML タグ値の比較 (数値)

## サプライヤ:在庫チェック・サブプロセスの概要

サプライヤ:在庫チェック・サブプロセスのプロパティを表示するには、そのプロセス・アクティビティをナビゲータ・ツリーで選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは、実行可能ではありません。つまり、最上位レベルのプロセスとして開始できず、上位レベルのプロセスからコールする必要があります。

サプライヤ:在庫チェック・サブプロセスの「プロセス」ウィンドウを表示すると、サブプロセスが3つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



このサブプロセスは、ノード1の「開始」アクティビティから開始します。ノード2で、在庫チェックを実行して、オーダー項目の在庫を確認します。この時点でサブプロセスは終了します。

## サプライヤ:在庫チェック・サブプロセスのアクティビティ

ここでは、サブプロセスの各アクティビティについて、アクティビティの表示名別に説明します。

### 開始 (ノード1)

この「標準」関数アクティビティは、プロセスの開始をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	なし

### 在庫チェック (ノード2)

現在、このアクティビティでは何も処理されません。このアクティビティに対して、在庫チェックを実行する関数を統合できます。

関数	WF_EVENTDEMO.STOCKCHECK
結果タイプ	なし
前提条件アクティビティ	開始

終了（ノード3）

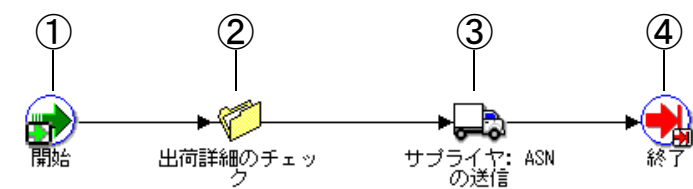
この「標準」関数アクティビティは、プロセスの終了をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	在庫チェック

サプライヤ：アドバンスト出荷通知サブプロセスの概要

サプライヤ：アドバンスト出荷通知サブプロセスのプロパティを表示するには、そのプロセス・アクティビティをナビゲータ・ツリーで選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは、実行可能ではありません。つまり、最上位レベルのプロセスとして開始できず、上位レベルのプロセスからコールする必要があります。

サプライヤ：アドバンスト出荷通知サブプロセスの「プロセス」ウィンドウを表示すると、サブプロセスが4つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



このサブプロセスは、ノード1の「開始」アクティビティから開始します。ノード2で、発注の出荷詳細をチェックします。次に、アドバンスト出荷通知をバイヤーに送信します。この時点でサブプロセスは終了します。

## サプライヤ: アドバンスト出荷通知サブプロセスのアクティビティ

ここでは、サブプロセスの各アクティビティについて、アクティビティの表示名別に説明します。

### 開始（ノード 1）

この「標準」関数アクティビティは、プロセスの開始をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	なし

### 出荷詳細のチェック（ノード 2）

現在、このアクティビティでは何も処理されません。このアクティビティに対して、発注の出荷詳細をチェックする関数を統合できます。

関数	WF_EVENTDEMO.CHECKSHIPPING
結果タイプ	なし
前提条件アクティビティ	開始

### サプライヤ: ASN の送信（ノード 3）

このイベント・アクティビティは、サプライヤ・システム上のアウトバウンド・エージェントおよびバイヤー・システム上のインバウンド・エージェントを使用して、アドバンスト出荷通知・メッセージをバイヤーに送信します。これらのエージェントは、サプライヤ: オーダー詳細の取得サブプロセスの「エージェント詳細の取得」アクティビティによって識別されます。

イベント・アクション	送信
前提条件アクティビティ	出荷詳細のチェック
アクティビティによって取得される項目属性	イベント・メッセージ、送信元エージェント / システム 2、宛先エージェント / システム 1

### 終了（ノード 4）

この「標準」関数アクティビティは、プロセスの終了をマークします。

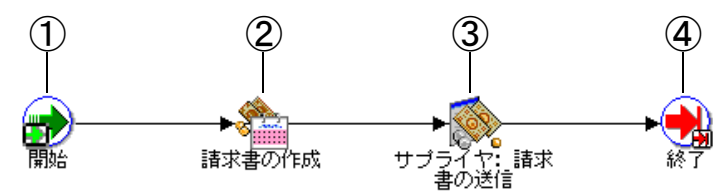
関数	WF_STANDARD.NOOP
結果タイプ	なし

前提条件アクティビティ サプライヤ: ASN の送信

サプライヤ: サプライヤの請求書の送信サブプロセスの概要

サプライヤ: サプライヤの請求書の送信サブプロセスのプロパティを表示するには、そのプロセス・アクティビティをナビゲータ・ツリーで選択し、「編集」メニューから「プロパティ」を選択します。このプロセス・アクティビティは、実行可能ではありません。つまり、最上位レベルのプロセスとして開始できず、上位レベルのプロセスからコールする必要があります。

サプライヤ: サプライヤの請求書の送信サブプロセスの「プロセス」ウィンドウを表示すると、サブプロセスが 4 つの異なるアクティビティから構成されていることがわかります。プロセスのアクティビティを詳しく説明するために、次のように各ノードに番号を付けて参照しやすくしてあります。番号自体は、プロセス・ダイアグラムには含まれません。



このサブプロセスは、ノード 1 の「開始」アクティビティから開始します。ノード 2 で、発注のオーダー項目について請求書を作成します。次に、請求書をバイヤーに送信します。この時点でサブプロセスは終了します。

サプライヤ: サプライヤの請求書の送信サブプロセスのアクティビティ

ここでは、サブプロセスの各アクティビティについて、アクティビティの表示名別に説明します。

開始（ノード 1）

この「標準」関数アクティビティは、プロセスの開始をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	なし



## 請求書の作成（ノード 2）

現在、このアクティビティでは何も処理されません。このアクティビティに対して、発注のオーダー項目について請求書を作成する関数を統合できます。

関数	WF_EVENTDEMO.CREATEINVOICE
結果タイプ	なし
前提条件アクティビティ	開始

## サプライヤ: 請求書の送信（ノード 3）

このイベント・アクティビティは、サプライヤ・システム上のアウトバウンド・エージェントおよびバイヤー・システム上のインバウンド・エージェントを使用して、請求書イベント・メッセージをバイヤーに送信します。これらのエージェントは、サプライヤ: オーダー詳細の取得サブプロセスの「エージェント詳細の取得」アクティビティによって識別されます。

イベント・アクション	送信
前提条件アクティビティ	請求書の作成
アクティビティによって 取得される項目属性	イベント・メッセージ、イベント・キー、送信元エージェント / システム 2、宛先エージェント / システム 1

## 終了（ノード 4）

この「標準」関数アクティビティは、プロセスの終了をマークします。

関数	WF_STANDARD.NOOP
結果タイプ	なし
前提条件アクティビティ	サプライヤ: 請求書の送信

発注イベント

このイベントは、「Buyer Workbench」デモンストレーション画面から発注を送信したときに、パイヤー・システム上で呼び出されます。

内部名 demo.oracle.wf.b2b.po.create  
ジェネレート関数 wf\_eventdemo.generatexml

Oracle Workflow には、発注イベントに対して、4 つのデフォルト・サブスクリプションが用意されています。最初のサブスクリプションでは、発注イベントがローカルで呼び出されたときに、**相関 ID** がイベント・メッセージに追加されます。**相関 ID** は、接頭辞 **PO** とイベント・キー（オーダー番号）で構成されます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 15-7

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	ローカル
イベント・フィルタ	demo.oracle.wf.b2b.po.create
フェーズ	1
ステータス	使用可能
ルール・データ	キー
ルール関数	wf_eventdemo.derivecorrelationid

2 番目のサブスクリプションでは、発注イベントがローカルで呼び出されたときに、「イベント・システムのデモ」項目タイプのパイヤー：最上位の発注プロセスにイベント・メッセージが送信されます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 15-8

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	ローカル
イベント・フィルタ	demo.oracle.wf.b2b.po.create
フェーズ	2
ステータス	使用可能

表 15-8 (続き)

サブスクリプションのプロパティ	値
ルール・データ	メッセージ
ルール関数	wf_rule.workflow_protocol
ワークフロー項目タイプ	WFEVDEME
ワークフロー・プロセス名	RCVPOCRT
送信エージェント	WF_OUT@<local system>
宛先エージェント	WF_IN@<local system>

3 番目のサブスクリプションでは、発注イベントが外部ソースから着信したときに、イベント・メッセージ内の相関 ID が変更されます。相関 ID は、接頭辞 SO とイベント・キー（オーダー番号）で構成されます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 15-9

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	外部
イベント・フィルタ	demo.oracle.wf.b2b.po.create
フェーズ	1
ステータス	使用可能
ルール・データ	キー
ルール関数	wf_eventdemo.derivecorrelationid

4 番目のサブスクリプションでは、発注イベントが外部ソースから呼び出されたときに、「イベント・システムのデモ」項目タイプのサブライヤ: 最上位のオーダー・プロセスにイベント・メッセージが送信されます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 15-10

サブスクリプションのプロパティ	値
システム	<local system>

表 15-10 (続き)

サブスクリプションのプロパティ	値
ソース・タイプ	外部
イベント・フィルタ	demo.oracle.wf.b2b.po.create
フェーズ	2
ステータス	使用可能
ルール・データ	キー
ルール関数	wf_rule.workflow_protocol
ワークフロー項目タイプ	WFEVDEME
ワークフロー・プロセス名	SUPPORCV
送信エージェント	WF_OUT@<local system>
宛先エージェント	WF_IN@<local system>

発注承認イベント

このイベントは、発注イベントが受信および処理された後で、サプライヤ・システムからバイヤー・システムに発注承認として送信されます。

内部名                      demo.oracle.wf.b2b.po.ack  
ジェネレート関数        なし

Oracle Workflow には、発注承認イベントに対して、2 つのデフォルト・サブスクリプションが用意されています。最初のサブスクリプションでは、発注承認イベントが外部ソースから着信したときに、相関 ID がイベント・メッセージに追加されます。相関 ID は、接頭辞 PO とイベント・キー（オーダー番号）で構成されます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 15-11

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	外部
イベント・フィルタ	demo.oracle.wf.b2b.po.ack
フェーズ	1
ステータス	使用可能

表 15-11 (続き)

サブスクリプションのプロパティ	値
ルール・データ	キー
ルール関数	wf_eventdemo.derivecorrelationid

2 番目のサブスクリプションでは、発注承認イベントが外部ソースから呼び出されたときに、「イベント・システムのデモ」項目タイプのバイヤー：最上位の発注プロセスにイベント・メッセージが送信されます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 15-12

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	外部
イベント・フィルタ	demo.oracle.wf.b2b.po.ack
フェーズ	2
ステータス	使用可能
ルール・データ	キー
ワークフロー項目タイプ	WFEVDEME
ワークフロー・プロセス名	RCVPOCRT

## アドバンスト出荷通知イベント

このイベントは、発注イベントが受信および処理された後で、サプライヤ・システムからバイヤー・システムにアドバンスト出荷通知として送信されます。

内部名 demo.oracle.wf.b2b.po.asn  
ジェネレート関数 なし

Oracle Workflow には、アドバンスト出荷通知イベントに対して、2つのデフォルト・サブスクリプションが用意されています。最初のサブスクリプションでは、アドバンスト出荷通知イベントが外部ソースから着信したときに、相関 ID がイベント・メッセージに追加されます。相関 ID は、接頭辞 PO とイベント・キー（オーダー番号）で構成されます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 15-13

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	外部
イベント・フィルタ	demo.oracle.wf.b2b.po.asn
フェーズ	1
ステータス	使用可能
ルール・データ	キー
ルール関数	wf_eventdemo.derivecorrelationid

2 番目のサブスクリプションでは、アドバンスト出荷通知イベントが外部ソースから呼び出されたときに、「イベント・システムのデモ」項目タイプのバイヤー：最上位の発注プロセスにイベント・メッセージが送信されます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 15-14

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	外部
イベント・フィルタ	demo.oracle.wf.b2b.po.asn
フェーズ	2

表 15-14 (続き)

サブスクリプションのプロパティ	値
ステータス	使用可能
ルール・データ	キー
ワークフロー項目タイプ	WFEVDEME
ワークフロー・プロセス名	RCVPOCRT

## 請求書イベント

このイベントは、発注イベントが受信および処理された後で、サプライヤ・システムからバイヤー・システムに請求書として送信されます。

内部名 demo.oracle.wf.b2b.po.invoice

ジェネレート関数 なし

Oracle Workflow には、請求書イベントに対して、2つのデフォルト・サブスクリプションが用意されています。最初のサブスクリプションでは、請求書イベントが外部ソースから着信したときに、相関 ID がイベント・メッセージに追加されます。相関 ID は、接頭辞 PO とイベント・キー（オーダー番号）で構成されます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 15-15

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	外部
イベント・フィルタ	demo.oracle.wf.b2b.po.invoice
フェーズ	1
ステータス	使用可能
ルール・データ	キー
ルール関数	wf_eventdemo.derivecorrelationid

2 番目のサブスクリプションでは、請求書イベントが外部ソースから着信したときに、「イベント・システムのデモ」項目タイプのバイヤー:最上位の発注プロセスにイベント・メッセージが送信されます。このサブスクリプションは、デフォルトで使用可能になっています。次の表は、このサブスクリプションに対して定義されているプロパティを示しています。

表 15-16

サブスクリプションのプロパティ	値
システム	<local system>
ソース・タイプ	外部
イベント・フィルタ	demo.oracle.wf.b2b.po.invoice
フェーズ	2
ステータス	使用可能
ルール・データ	キー
ワークフロー項目タイプ	WFEVDEME
ワークフロー・プロセス名	RCVPOCRT



---

## Oracle Workflow の管理スクリプト

この章では、ワークフロー管理者が Oracle Workflow Server に対して実行できる SQL スクリプトについて説明します。

## 様々な SQL スクリプト

次の管理スクリプトを使用すると、Oracle Workflow の各種機能を設定して保存できます。Oracle Workflow のスタンドアロン版の場合、各スクリプトは使用しているサーバーの Oracle Workflow の `admin/sql` サブディレクトリにあります。Oracle Applications embedded Workflow の場合は、`$FND_TOP` の `sql` サブディレクトリにあります。

- 変換表の更新 : 16-6 ページ「[WFNLADD.sql](#)」
- 言語の有効化 / 無効化 : 16-12 ページ「[wfnlena.sql](#)」
- ワークフロー・プロセスの実行 : 16-16 ページ「[wfrun.sql](#)」
- バックグラウンド・エンジンの起動 : 16-7 ページ「[wfbkg.sql](#)」
- バックグラウンド・エンジンの次の実行のために繰り延べられたアクティビティの表示 : 16-7 ページ「[wfbkgchk.sql](#)」
- 項目のステータス・レポートの表示
  - 16-16 ページ「[wfstatus.sql](#)」
  - 16-16 ページ「[wfstat.sql](#)」
- 通知のステータスの表示 : 16-13 ページ「[wfntfsh.sql](#)」
- オブジェクトの保護レベルのリセット : 16-13 ページ「[wfprot.sql](#)」
- エラーが発生したアクティビティの処理 : 16-14 ページ「[wfretry.sql](#)」
- バージョン・エラーおよびプロセス定義エラーのチェック
  - 16-17 ページ「[wfverchk.sql](#)」
  - 16-17 ページ「[wfverupd.sql](#)」
  - 16-16 ページ「[wfstdchk.sql](#)」
- 無効な外部キーのチェック : 16-14 ページ「[wfrefchk.sql](#)」
- ディレクトリ・サービスのデータ・モデルのチェック : 16-10 ページ「[wfdirchk.sql](#)」
- システム表でのワークフロー・キューのクリーン・アップ : 16-13 ページ「[wfqclean.sql](#)」
- ワークフロー・オブジェクトの内部名の変更

---

**注意：** 通常、Oracle Workflow Builder では、ワークフロー・オブジェクトの内部名は更新できません。ただし、プロセス定義をデータベースにロードする場合に、オブジェクトに対するランタイム・データが存在しなければ、スクリプトの 1 つを使用して、ワークフロー・オブジェクトの内部名を更新できます。このスクリプトは、設計時にオブジェクトの内部名に関するエラーを訂正する場合にのみ使用してください。プロセスの実行中のインスタンスに関連するオブジェクトの名前の変更には、使用しないでください。

---

- 16-8 ページ [「wfchact.sql」](#)
  - 16-8 ページ [「wfchacta.sql」](#)
  - 16-8 ページ [「wfchita.sql」](#)
  - 16-9 ページ [「wfchitt.sql」](#)
  - 16-9 ページ [「wfchluc.sql」](#)
  - 16-9 ページ [「wfchlut.sql」](#)
  - 16-10 ページ [「wfchmsg.sql」](#)
  - 16-10 ページ [「wfchmsga.sql」](#)
- Oracle Workflow の表からのデータ削除
- 16-14 ページ [「wfrmall.sql」](#)
  - 16-15 ページ [「wfrmitms.sql」](#)
  - 16-15 ページ [「wfrmitt.sql」](#)
  - 16-16 ページ [「wfrmtime.sql」](#)
  - 16-15 ページ [「wfrmita.sql」](#)

---

**注意：** Oracle Applications では、「ワークフローの不要ランタイム・データのページ」と呼ばれる標準コンカレント・プログラムも使用できます。16-5 ページの [「FNDWFPR」](#) を参照してください。

また、Oracle9i リリース 2 で利用可能なスタンドアロン版の Oracle Workflow を使用している場合は、Oracle Enterprise Manager から利用可能なスタンドアロン版の Oracle Workflow Manager コンポーネントを使用して、ワークフローのページ・データベース・ジョブを管理できます。詳細は、Oracle Workflow Manager のオンライン・ヘルプを参照してください。

---

- 通知メーラーのアウトバウンド・メッセージ・キューの削除および再作成 : 16-12 ページ  
「[wfmqupd.sql](#)」

---

**警告：** このスクリプトは、オラクル社カスタマ・サポートから指示があった場合にのみ使用してください。オラクル社カスタマ・サポートから指示がない限り、このスクリプトを実行しないでください。

---

- Oracle Workflow Server のバージョンの表示 : 16-17 ページ 「[wfver.sql](#)」
- Java 関数アクティビティ・エージェントの停止 : 16-12 ページ 「[wfjvstop.sql](#)」
- オーバーライド・エージェントによるキューへのイベント・メッセージのエンキュー : 16-11 ページ 「[wfevteng.sql](#)」
- インバウンド・イベント・メッセージに関してエージェントを監視するリスナーの実行 : 16-6 ページ 「[wfgtltst.sql](#)」

---

**注意：** Oracle Applications では、「ワークフロー・エージェント・リスナー」と呼ばれる標準コンカレント・プログラムも使用できます。16-5 ページの「[ENDWFLST](#)」を参照してください。

また、Oracle9i リリース 2 で利用可能なスタンドアロン版の Oracle Workflow を使用している場合は、Oracle Enterprise Manager から利用可能なスタンドアロン版の Oracle Workflow Manager コンポーネントを使用して、ワークフロー・エージェント・リスナーのデータベース・ジョブを管理できます。詳細は、Oracle Workflow Manager のオンライン・ヘルプを参照してください。

---

## FNDWFLST

Oracle Applications embedded Workflow では、標準コンカレント・プログラム FNDWFLST「ワークフロー・エージェント・リスナー」を使用して、インバウンド・イベント・メッセージのエージェントを監視してください。

Oracle Applications の「Submit Requests」フォームにナビゲートし、「ワークフロー・エージェント・リスナー」コンカレント・プログラムを発行します。Oracle Applications と Oracle Workflow をインストールして設定するときに、システム管理者はこのコンカレント・プログラムを実行する職責の要求セキュリティ・グループに追加する必要があります。『Oracle Applications System Administrator's Guide』の「Overview of Concurrent Programs and Requests」および『Oracle Applications User's Guide』の「Submit Requests」を参照してください。

監視するエージェントの名前を、「ワークフロー・エージェント・リスナー」コンカレント・プログラムのパラメータとして渡す必要があります。

---

**注意：** Oracle Applications embedded Workflow を使用し、Oracle Applications Manager を実装している場合は、Oracle Workflow Manager を使用して「ワークフロー・エージェント・リスナー」コンカレント・プログラムを発行および管理することができます。詳細は、Oracle Applications Manager のオンライン・ヘルプを参照してください。

---

## FNDWFPR

Oracle Applications embedded Workflow では、標準コンカレント・プログラム FNDWFPR「ワークフローの不要ランタイム・データのページ」を使用して、古くなったデータを Oracle Workflow ランタイム表から定期的に削除してください。

Oracle Applications の「Submit Requests」フォームにナビゲートし、「ワークフローの不要ランタイム・データのページ」コンカレント・プログラムを発行します。Oracle Applications と Oracle Workflow をインストールして設定するときに、システム管理者はこのコンカレント・プログラムを実行する職責の要求セキュリティ・グループに追加する必要があります。『Oracle Applications System Administrator's Guide』の「Overview of Concurrent Programs and Requests」および『Oracle Applications User's Guide』の「Submit Requests」を参照してください。

「ワークフローの不要ランタイム・データのページ」コンカレント・プログラムには、次のパラメータを渡すことができます。

- 項目タイプ： 削除する項目タイプ。このフィールドを空のままにすると、デフォルトですべての項目タイプのランタイム・データが削除されます。
- 項目キー： 削除する項目キー。このフィールドを空のままにすると、デフォルトですべての項目キーのランタイム・データが削除されます。
- 持続日数： 削除するデータの最小経過日数。

- 維持タイプ: 削除の維持タイプとして、「TEMP」(一時)または「PERM」(永続)を指定します。デフォルトは「TEMP」です。

---

**注意:** Oracle Applications embedded Workflow を使用し、Oracle Applications Manager を実装している場合は、Oracle Workflow Manager を使用して「ワークフローの不要ランタイム・データのパージ」コンカレント・プログラムを発行および管理することができます。詳細は、Oracle Applications Manager のオンライン・ヘルプを参照してください。

---

## WFNLADD.sql

Oracle インストールで新しい言語を使用可能にする場合は、WFNLADD.sql を使用して Oracle Workflow の変換表にその言語用の欠落行を追加します。2-36 ページの「[手順 WF-5 WF\\_LANGUAGES ビューの作成](#)」および 16-12 ページの「[wfnlana.sql](#)」を参照してください。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @WFNLADD
```

## wfagtlst.sql

wfagtlst.sql を使用してリスナーを実行すれば、インバウンド・イベント・メッセージのエージェントを監視できます。メッセージが着信すると、イベント・マネージャは、「外部」ソース・タイプを持つそのイベントの有効なサブスクリプションを、ローカル・システム単位に検索および実行します。また、「外部」ソース・タイプを持つ Any イベントの有効なサブスクリプションを、ローカル・システム単位に検索および実行します。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfagtlst <agent_name>
```

<agent\_name> は、監視するインバウンド・イベント・メッセージのエージェントの内部名に置き換えてください。

---

**注意:** このスクリプトは、主にデバッグ用として使用してください。

---

### 関連項目:

8-277 ページ [「Listen」](#)

## wfbkg.sql

Oracle Workflow のスタンドアロン版を使用している場合は、wfbkg.sql を使用してバックグラウンド・エンジンを起動できますこのスクリプトは、WF\_ENGINE バックグラウンド API をコールし、指定した時間（分）だけバックグラウンド・エンジンを実行します。「適格」アクティビティの現行セットの処理が終了すると、バックグラウンド・プロセスは、指定した時間（秒）だけ待機してから、他のバックグラウンド・エンジンを起動します。このサイクルは、指定した時間（分）が経過するまで続きます。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfbkg <minutes> <seconds>
```

<minutes> をバックグラウンド・エンジンを実行する時間（分）に置き換え、<seconds> を間合せ間でバックグラウンド・エンジンを待機させる秒数に置き換えます。

### 関連項目：

8-44 ページ [「Background」](#)

2-40 ページ [「手順 WF-8 バックグラウンドのワークフロー・エンジンの設定」](#)

## wfbkgchk.sql

wfbkgchk.sql を使用すると、バックグラウンド・エンジンを次回に実行するまでに処理待ちになっているすべてのアクティビティのリストを表示できます。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfbkgchk
```

### 関連項目：

8-44 ページ [「Background」](#)

2-40 ページ [「手順 WF-8 バックグラウンドのワークフロー・エンジンの設定」](#)

## wfchact.sql

wfchact.sql を使用して、アクティビティの内部名の変更やアクティビティのすべての参照の更新ができます。16-2 ページの「[ワークフロー・オブジェクトの内部名の変更](#)」を参照してください。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfchact <act_type> <old_act> <new_act>
```

<act\_type> を、更新するアクティビティに関連付けられている項目タイプに、<old\_act> をアクティビティの現行の内部名に、<new\_act> をアクティビティの新規の内部名に置き換えてください。

## wfchacta.sql

wfchacta.sql を使用して、アクティビティ属性の内部名の変更や、アクティビティ属性のすべての参照の更新ができます。16-2 ページの「[ワークフロー・オブジェクトの内部名の変更](#)」を参照してください。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfchacta <act_type> <old_acta> <new_acta>
```

<act\_type> を、更新するアクティビティ属性に関連付けられている項目タイプに、<old\_acta> をアクティビティ属性の現行の内部名に、<new\_acta> をアクティビティ属性の新規の内部名に置き換えてください。

## wfchita.sql

wfchita.sql を使用して、項目属性の内部名の変更や、項目属性のすべての参照の更新ができます。16-2 ページの「[ワークフロー・オブジェクトの内部名の変更](#)」を参照してください。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfchita <item_type> <old_attr> <new_attr>
```

<item\_type> を、更新する項目属性に関連付けられている項目タイプに、<old\_attr> を項目属性の現行の内部名に、<new\_acta> を項目属性の新規の内部名に置き換えてください。



## wfchitt.sql

wfchitt.sql を使用して、項目タイプの内部名の変更や、項目タイプのすべての参照の更新ができます。16-2 ページの「[ワークフロー・オブジェクトの内部名の変更](#)」を参照してください。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfchitt <old_type> <new_type>
```

<old\_type> を項目属性の現行の内部名に、<new\_type> を項目属性の新規の内部名に置き換えてください。

## wfchluc.sql

wfchluc.sql を使用して、選択肢コードの内部名の変更や、選択肢コードのすべての参照の更新ができます。16-2 ページの「[ワークフロー・オブジェクトの内部名の変更](#)」を参照してください。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfchluc <lookup_type> <old_luc> <new_luc>
```

<lookup\_type> を、更新する選択肢コードの選択肢タイプに、<old\_luc> を選択肢コードの現行の内部名に、<new\_luc> を選択肢コードの新規の内部名に置き換えてください。

## wfchlut.sql

wfchlut.sql を使用して、選択肢タイプの内部名の変更や、選択肢タイプのすべての参照の更新ができます。16-2 ページの「[ワークフロー・オブジェクトの内部名の変更](#)」を参照してください。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfchlut <old_lut> <new_lut>
```

<old\_lut> を選択肢タイプの現行の内部名に、<new\_lut> を選択肢タイプの新規の内部名に置き換えてください。

## wfchmsg.sql

wfchmsg.sql を使用して、メッセージの内部名の変更や、メッセージのすべての参照の更新ができます。16-2 ページの「[ワークフロー・オブジェクトの内部名の変更](#)」を参照してください。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfchmsg <msg_type> <old_msg> <new_msg>
```

<msg\_type> を、更新するメッセージの項目タイプに、<old\_msg> をメッセージの現行の内部名に、<new\_msg> をメッセージの新規の内部名に置き換えてください。

## wfchmsga.sql

wfchmsga.sql を使用して、メッセージ属性の内部名を変更できます。このスクリプトでは、メッセージの件名や本文のメッセージ属性への参照は更新されません。メッセージ属性の参照は、手動で更新する必要があります。16-2 ページの「[ワークフロー・オブジェクトの内部名の変更](#)」を参照してください。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfchmsga <msg_type> <msg_name> <old_attr> <new_attr>
```

<msg\_type> を、更新するメッセージ属性の項目タイプに、<msg\_name> をメッセージ属性が属するメッセージの内部名に、<old\_attr> をメッセージ属性の現行の内部名に、<new\_attr> をメッセージ属性の新規の内部名に置き換えてください。

## wfdirchk.sql

wfdirchk.sql を使用すると、ディレクトリ・サービスのデータ・モデル内で次の条件をチェックできます。

- WF\_USERS 内の文字「#」、「:」、または「/」を含む無効な内部名
- WF\_USERS または WF\_ROLES 内の無効な複合名
- WF\_USERS または WF\_ROLES 内の重複名
- WF\_USERS または WF\_ROLES 内で元のリポジトリ内の同じ行にリンクされている複数の名前
- WF\_USERS または WF\_ROLES 内の不明な表示名
- WF\_USERS または WF\_ROLES 内で「通知環境設定」が MAILTEXT、MAILHTML または SUMMARY の場合の無効な「通知環境設定」または未入力の電子メール・アドレス
- WF\_USERS 内の無効なステータス
- WF\_ROLES 内に対応する行がない WF\_USERS 内の行

- WF\_ROLES 内で文字「#」または「/」を含むか、あるいは長さ 30 文字（半角英数字）を超える無効な内部名
- WF\_USER\_ROLES 内の無効なユーザー / ロール外部キー
- WF\_USER\_ROLES 内の不明なユーザー / ロール（すべてのユーザーは、各自のロールに参加する必要があります。）
- WF\_USER\_ROLES 内の重複行

wfdirchk.sql では、ディレクトリ・サービスのデータ・モデルが正しいかどうかを確認する行は戻されません。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfdirchk
```

## wfevttenq.sql

wfevttenq.sql では、オーバーライド・エージェントを使用してイベント・メッセージをローカル・キューにエンキューできます。このスクリプトでは、指定したイベント名、イベント・キー、イベント・データ、送信元エージェントおよび宛先エージェントを使用してイベント・メッセージを作成します。作成したイベント・メッセージは、指定したオーバーライド・エージェントに関連付けられているキューにエンキューされます。オーバーライド・エージェントには、イベント・メッセージの送信元エージェント以外のエージェントを指定できます。オーバーライド・エージェントを指定しない場合は、イベント・メッセージはメッセージの送信元エージェントにデフォルトでエンキューされます。

---

**注意：** このスクリプトでは、ローカル・システム上のエージェントのキューにのみイベント・メッセージをエンキューできます。

---

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfevttenq <overrideagent> <overridesystem> <fromagent>  
<fromsystem> <toagent> <tosystem> <eventname> <eventkey> <message>
```

次のように、各変数を適切なパラメータに置き換えてください。

- <overrideagent>: イベント・メッセージをエンキューするキューのエージェント
- <overridesystem>: オーバーライド・エージェントが配置されているシステム
- <fromagent>: イベント・メッセージに追加する送信元エージェント
- <fromsystem>: 送信元エージェントが配置されているシステム
- <toagent>: イベント・メッセージを受信する宛先エージェント
- <tosystem>: 宛先エージェントが配置されているシステム

- `<eventname>`: イベントの内部名
- `<eventkey>`: イベントのインスタンスを一意に識別するイベント・キー
- `<message>`: イベント・データ

### wfjvstop.sql

wfjvstop.sql を使用すると、停止メッセージを「Outbound」キューに配置して、Java 関数アクティビティ・エージェントを停止できます。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfjvstop
```

### wfmqupd.sql

wfmqupd.sql を使用すると、通知メーラーのアウトバウンド・メッセージ・キューを削除し、WF\_NOTIFICATIONS 表からキューを再作成できます。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfmqupd
```

---

---

**警告：** このスクリプトは、オラクル社カスタマ・サポートから指示があった場合にのみ使用してください。オラクル社カスタマ・サポートから指示がない限り、このスクリプトを実行しないでください。

---

---

### wfnlena.sql

Oracle インストールで新しい言語を定義する場合は、wfnlena.sql を使用すると、その言語を Oracle Workflow で使用可能または使用不可にすることができます。16-6 ページの「WFNLADD.sql」を参照してください。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfnlena <language_code> <enable_flag>
```

`<language_code>` を有効な言語コードに置き換え、指定した言語を使用可能にする場合は `<enable_flag>` を Y に、使用不可にする場合は N に置き換えます。

## wfntfsh.sql

wfntfsh.sql を使用すると、通知 ID を指定した特定の通知のステータスに関する情報を表示できます。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfntfsh <notification_id>
```

## wfprot.sql

wfprot.sql を使用すると、指定した項目タイプに関連したすべてのオブジェクトの保護レベルをリセットできます。

---

---

**注意：** ある項目タイプのすべてのオブジェクトの保護レベルをリセットすると、新規の保護レベルより上位のアクセス・レベルで操作するユーザーは、その項目タイプの各オブジェクトをカスタマイズできなくなります。

---

---

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfprot <item_type> <protection_level>
```

<Item\_type> を、保護レベルをリセットする項目タイプに置き換え、<protection\_level> を新しい保護レベルに置き換えます。

## wfqclean.sql

wfqclean.sql を使用して、システム表のワークフロー・キューのクリーン・アップができます。

---

---

**注意：** このスクリプトが必要になるのは、Oracle8i リリース 8.1.5 以前の Oracle8 で、wfqued.sql を使用して事前にワークフロー・キューを削除せずに、ユーザーまたは表領域を削除した場合のみです。wfqued.sql スクリプトは、Oracle Workflow の sql サブディレクトリにあります。以前のバージョンの Oracle8 で、DROP USER CASCADE コマンドや DROP TABLESPACE INCLUDING CONTENTS コマンドを使用すると、システム表にキュー・データが残り、その結果、キューを再作成すると、ORA-00600 エラーになります。これを回避するには、常に wfqued.sql を実行してキューを削除してから、ユーザーまたは表領域を削除してください。

---

---

wfqclean.sql スクリプトの使用方法は、次のとおりです。

```
sqlplus system/manager @wfgclean <un>
```

<un> を、ORA-00600 エラーが発生するスキーマのユーザー名に置き換えてください。

### wfrefchk.sql

wfrefchk.sql を使用して、外部キーに対する主キー・データの無い、無効なワークフロー・データを検索できます。

```
sqlplus <user/pwd> @wferfchk
```

### wfretry.sql

wfretry.sql を使用すると、特定プロセスのインスタンスに関してエラーが発生したアクティビティのリストを表示し、そのアクティビティをスキップするか、再試行するか、またはリセットするかを指定できます。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfretry <item_type> <item_key>
```

項目タイプと項目キーを指定して、特定の項目またはプロセスのインスタンスを識別します。このスクリプトでは、最初に、エラーが発生したアクティビティのラベル名別リストが戻されます。次に、スキップ、再試行またはリセットするアクティビティのラベル名を求めるプロンプトが表示されます。スキップするように選択した場合は、スキップするアクティビティの結果も指定する必要があります。

---

---

**注意：** このスクリプトでは、WF\_ENGINE HandleError API がコールされるため、実際には指定した項目タイプと項目キーに関連した任意のアクティビティのラベル名を指定して、ロールバックを実行できます。8-79 ページの「[HandleError](#)」を参照してください。

---

---

### wfrmall.sql

wfrmall.sql を使用すると、Oracle Workflow の設計時と実行時のすべての表の全データを削除できます。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfrmall
```

---

---

**警告：** このスクリプトを実行すると、すべてのワークフロー定義が削除されます。実行時と設計時の表からすべてのワークフロー・データを削除することに絶対的な確信がない限り、このスクリプトは使用しないでください。

このスクリプトを実行した後に、「標準」、「システム：メーラー」および「システム：エラー」項目タイプのワークフロー定義も再ロードする必要があります。各ワークフロー定義は、それぞれファイル wfstd.wft、wfmail.wft および wferror.wft に格納されています。

---

---

## wfrmita.sql

wfrmita.sql を使用して、指定した項目タイプ属性のワークフロー・データをすべて削除できます。このスクリプトを実行すると、削除対象の項目タイプと属性名の入力を求めるプロンプトが表示されます。または、**Oracle Workflow Builder** を使用して、ファイルまたはデータベースに格納されたワークフロー定義から、項目タイプ属性を削除することもできます。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfrmita
```

## wfrmitms.sql

wfrmitms.sql を使用すると、**Oracle Workflow** の実行時の表から特定項目のステータス情報を削除できます。このスクリプトを実行すると、指定した項目タイプと項目キーに関連したすべてのデータを削除するか、指定した項目タイプと項目キーの完了したアクティビティのデータのみを削除するかを選択を求めるプロンプトが表示されます。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfrmitms <item_type> <item_key>
```

## wfrmitt.sql

wfrmitt.sql を使用すると、**Oracle Workflow** の設計時と実行時のすべての表から、特定の項目タイプの全データを削除できます。このスクリプトを実行すると、有効な項目タイプのリストから項目タイプを選択するように求めるプロンプトが表示されます。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfrmitt
```

---

---

**警告：** このスクリプトを実行すると、指定した項目タイプのワークフロー・データがすべて削除されます。

---

---

## wfrmttype.sql

wfrmttype.sql を使用すると、特定の項目タイプに関連したランタイム・データを削除できます。このスクリプトを実行すると、削除する項目タイプを有効な値リストから選択するように求めるプロンプトが表示されます。次に、指定した項目タイプに関連したすべてのランタイム・データを削除するか、または指定した項目タイプの完了したアクティビティと項目のランタイム・データのみ削除するかが確認されます。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfrmttype
```

## wfrun.sql

wfrun.sql を使用すると、指定したプロセスを作成して開始できます。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfrun <item_type> <item_key> <process_name>
```

## wfstat.sql

wfstat.sql を使用すると、指定した項目の開発者用ステータス・レポートを表示できます。出力は半角英数字で 1 行当り 132 文字です。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfstat <item_type> <item_key>
```

## wfstatus.sql

wfstatus.sql を使用すると、指定した項目のエンド・ユーザー用ステータス・レポートを表示できます。出力は半角英数字で 1 行当り 132 文字です。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfstatus <item_type> <item_key>
```

## wfstdchk.sql

wfstdchk.sql を使用して、Oracle Workflow データ・モデルで検出された問題のチェックとレポートができます。たとえば、このスクリプトは、無効な関数を参照する関数アクティビティをレポートし、各ワークフロー・プロセス定義オブジェクト表を検索して、各行に有効な内部名と表示名が入っているかどうかを検証します。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfstdchk
```



## wfver.sql

wfver.sql を使用すると、インストールされている Oracle Workflow Server のバージョン、Oracle Workflow PL/SQL パッケージのステータスとバージョン、および Oracle Workflow ビューのバージョンを表示できます。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfver
```

## wfverchk.sql

wfverchk.sql を使用するのには、同時にアクティブになっている複数バージョンのアクティビティが、ワークフロー・プロセスで起きている問題の原因と思われる場合です。このスクリプトは、複数バージョンが同時にアクティブであるように見える原因となっている、アクティビティのエラーを識別します。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfverchk
```

## wfverupd.sql

wfverupd.sql を使用すると、複数バージョンのアクティビティが同時にアクティブになっていることが原因で、ワークフロー・プロセスで起きている問題を解決できます。このスクリプトは、複数バージョンが同時にアクティブであるように見える原因となっている、アクティビティのエラーを識別し、修正します。

このスクリプトの使用方法は、次のとおりです。

```
sqlplus <user/pwd> @wfverupd
```



# A

---

## Oracle Workflow Builder のメニューと ツールバー

付録 A では、Oracle Workflow Builder のメニューとツールバーについて説明します。

## Oracle Workflow Builder のメニュー

Oracle Workflow Builder のメイン・メニュー・バーには、次のメニューが含まれています。

- ファイル
- 編集
- 表示
- ウィンドウ
- ヘルプ

### 「ファイル」メニュー

「ファイル」メニューを使用すると、次のような処理を実行できます。

<b>新規作成</b>	項目タイプを定義するための新規作業領域を作成します。
<b>クイック・スタート・ウィザード</b>	ワークフロー・プロセス定義の設計を開始できるフレームワークを作成します。3-19 ページの「 <a href="#">クイック・スタート・ウィザードの概要</a> 」を参照してください。
<b>オープン ...</b>	データベースやファイルへの接続を求めるプロンプトが表示され、データ・ストアが開きます。3-13 ページの「 <a href="#">項目タイプのオープンと保存</a> 」を参照してください。
<b>ストアを閉じる</b>	選択したデータ・ストアを閉じます。このメニュー・オプションを選択できるのは、ナビゲータがアクティブ・ウィンドウになっている場合に限られます。
<b>保存</b>	現在接続されているデータベースまたはファイルに変更内容を保存します。3-13 ページの「 <a href="#">項目タイプのオープンと保存</a> 」を参照してください。
<b>別名保存</b>	変更内容を、指定したファイルまたはデータベースに任意の有効日付で保存します。

- ショートカットの作成** Oracle Workflow Builder の現行セッションのデスクトップにショートカット・アイコンを作成します。ショートカット名の入力を求めるプロンプトが表示されます。ショートカットを選択すると、Oracle Workflow Builder が実行され、ショートカットの作成時に選択したデータ・ストアに自動的に接続して項目タイプをロードし、その時点でロードされて開いていたプロセス・ウィンドウが開きます。データ・ストアがデータベースの場合は、ショートカットを実行すると、Oracle Workflow Builder の起動前に、データベースのパスワード入力を求めるプロンプトが表示されます。この機能を使用できるのは、Windows NT 4.0 以上で Oracle Workflow Builder を実行する場合のみです。Microsoft Windows NT の旧バージョンでは、ショートカットの概念がサポートされていません。5-22 ページの「ワークフロー・プロセスへのショートカット・アイコンの作成」を参照してください。
- 検証** 現行のデータ・ストア内のすべてのプロセス定義を検証します。「更新」を使用して、プロセスの最新の検証レポートを表示します。5-21 ページの「[プロセス定義の検証](#)」を参照してください。
- ダイアグラムの印刷** アクティブなプロセス・ウィンドウに表示されているプロセス・ダイアグラムを印刷します。5-21 ページの「[プロセスの印刷](#)」を参照してください。
- 項目タイプの表示 / 非表示** 「項目タイプの表示」ウィンドウが表示され、現行データ・ストアのどの項目タイプをナビゲータ・ツリーで表示または非表示にするかを指定します。
- データベースからロールをロード** Oracle Workflow ディレクトリ・サービスのロールが、現行データベース・ストアから Oracle Workflow Builder にロードされ、ナビゲータ・ツリーの「ディレクトリ・サービス」ブランチや、ロールを参照するプロパティ画面のドロップダウン・フィールドから表示できるようになります。このメニュー・オプションを選択できるのは、現行データ・ストアがデータベースの場合に限られます。5-24 ページの「[ロール](#)」を参照してください。
- 終了** Oracle Workflow Builder を終了します。

## 「編集」メニュー

「編集」メニューは、「ナビゲータ」ウィンドウを選択した場合とプロセス・ウィンドウを選択した場合で表示が異なります。次のメニュー・オプションは、「ナビゲータ」ウィンドウを選択した場合にのみ表示され、「ナビゲータ」ウィンドウにのみ適用されます。

<b>新規作成</b>	プロパティ画面を表示し、項目タイプ、関数アクティビティ、プロセス・アクティビティ、通知アクティビティ、イベント・アクティビティ、メッセージ、選択肢タイプ、選択肢コードまたは属性を新規作成します。
<b>コピー</b>	選択したオブジェクトをナビゲータ・ツリーにコピーします。
<b>貼り付け</b>	オブジェクトを、クリップボードからナビゲータ・ツリーで選択したブランチに貼り付けます。
<b>削除</b>	選択したオブジェクトをナビゲータ・ツリーから削除します。
<b>検索</b>	「検索」ウィンドウが表示され、検索基準を入力してナビゲータ・ツリーでオブジェクトを検索できます。3-6 ページの「ナビゲータ・ツリーでのオブジェクトの検索」を参照してください。
<b>再検索</b>	前に「検索」ウィンドウで定義したのと同じ基準を使用して、ナビゲータ・ツリーでオブジェクトを検索します。
<b>プロパティ</b>	選択したオブジェクトのプロパティ画面を表示します。
<b>プロセスの詳細</b>	選択したプロセス・アクティビティのプロセス・ウィンドウを開きます。
<b>属性の移動</b>	選択した属性をリスト内で上下に移動し、ナビゲータ・ツリーの現在のブランチに表示されている属性の順序を変更します。

次のメニュー・オプションが表示されるのは、プロセス・ウィンドウを選択している場合に限り、選択したプロセス・ウィンドウにのみ適用されます。

<b>選択の削除</b>	選択したオブジェクトをプロセス・ウィンドウから削除します。
<b>プロパティ</b>	選択したアクティビティ・ノードのプロパティ画面を表示します。
<b>ダイアグラムのコピー</b>	アクティブなプロセス・ウィンドウに表示されているプロセス・ダイアグラムを、クリップボードにコピーします。5-21 ページの「 <a href="#">プロセス・ダイアグラムをクリップボードにコピー</a> 」を参照してください。

## 「表示」メニュー

「表示」メニューを使用すると、Oracle Workflow Builder の表示を変更できます。

<b>フォント</b>	「フォント」プロパティ画面を表示します。このプロパティ画面を使用して、「ナビゲータ」ウィンドウやプロセス・ウィンドウに表示されるテキストのフォントの設定を変更します。変更は、それ以降のすべての Oracle Workflow Builder セッションに適用されます。5-21 ページの「 <a href="#">Oracle Workflow Builder のフォントの変更</a> 」を参照してください。
-------------	---

<b>ログ → 表示</b>	「ログ」ウィンドウの表示と非表示を切り替えます。「メッセージ・ログ」ウィンドウには、ワークフロー・ビルダーからのエラーと無関係なメッセージが表示されます。
<b>ログ → 詳細</b>	Oracle Workflow Builder のデバッグ・モードのオンとオフを切り替えます。「詳細」をオンにすると、デバッグ・モードがオンになり、より詳細なメッセージが「ログ」ウィンドウに書き込まれます。このモードをオンにすると Oracle Workflow Builder のパフォーマンスが大幅に低下するため、オラクル社カスタマ・サポート・センターの担当者から特に指示がない限り、「詳細」をオンにしないでください。
<b>ログ → ファイルに保存</b>	「メッセージ・ログ」ウィンドウの今後の内容がすべてファイルに書き込まれます。「表示」メニューから「Log Show」を選択して、ログ・ファイルの位置と名前を指定します。
<b>ログ → 前面に移動</b>	「メッセージ・ログ」ウィンドウがアクティブ・ウィンドウとして一番前面に移動にされます。
<b>スナップのグリッド</b>	すべてのプロセス・ウィンドウのグリッド表示のオンとオフを切り替えます。
<b>「デザイナーでラベルを表示」サブメニュー</b>	オプションのサブメニューを使用すると、アクティビティ・ノード・ラベルに表示される情報を制御できます。「インスタンス・ラベル」、「内部名」、「表示名」、「実行者」または「コメント」を選択します。
<b>デザイナーでラベルを表示 → インスタンス・ラベル</b>	プロセス・ダイアグラム内のアクティビティ・ノードのラベルとして、ノード・ラベルを使用します。この設定は、明示的に変更するまで、すべてのプロセス・ダイアグラムとすべての Oracle Workflow Builder セッションに適用されます。
<b>デザイナーでラベルを表示 → 内部名</b>	プロセス・ダイアグラム内のアクティビティ・ノードのラベルとして、アクティビティの内部名を使用します。この設定は、明示的に変更するまで、すべてのプロセス・ダイアグラムとすべての Oracle Workflow Builder セッションに適用されます。
<b>デザイナーでラベルを表示 → 表示名</b>	プロセス・ダイアグラム内のアクティビティ・ノードのラベルとして、アクティビティの表示名を使用します。この設定は、明示的に変更するまで、すべてのプロセス・ダイアグラムとすべての Oracle Workflow Builder セッションに適用されます。
<b>デザイナーでラベルを表示 → 実行者</b>	プロセス・ダイアグラム内のアクティビティのラベルとして、アクティビティの実行者を使用します。実行者が存在しない関数アクティビティとプロセス・アクティビティには、ラベルは表示されません。この設定は、明示的に変更するまで、すべてのプロセス・ダイアグラムとすべての Oracle Workflow Builder セッションに適用されます。

**ラベルの表示 → コメント**

プロセス・ダイアグラム内のアクティビティ・ノードのラベルとして、そのアクティビティのコメントを使用します。コメントが付いていないアクティビティには、ラベルは表示されません。この設定は、明示的に変更するまで、すべてのプロセス・ダイアグラムとすべての Oracle Workflow Builder セッションに適用されます。

**開発者モード**

表示を標準のプレゼンテーション・モードと開発者モードに切り替えます。開発者モードでは、アイコンはすべて特定のオブジェクト・タイプ / サブタイプのデフォルト・アイコンに戻り、サブプロセスにはトップ・レベルのプロセス・アイコンと異なるアイコンが表示され、ナビゲータ・ツリーでは、オブジェクトが内部名で表示およびソートされます。属性は内部名で表示されますが、ソートはされないので注意してください。

アクティブ・ウィンドウが「ナビゲータ」ウィンドウの場合は、次のメニュー・オプションも表示されます。

**ウィンドウを分割**

ナビゲータ・ウィンドウを縦方向または横方向に分割します。

アクティブ・ウィンドウがプロセス・ウィンドウの場合は、次のメニュー・オプションも表示されます。

**概要**

プロセスの「概要」ウィンドウを表示します。5-20 ページの「[プロセス概要の表示](#)」を参照してください。

**ナビゲータでプロセスを表示**

このメニュー・オプションを選択すると、プロセス・ダイアグラム・ウィンドウに表示されている現行プロセスに対応するプロセス・アクティビティが、「ナビゲータ」ウィンドウに表示されます。

**オーバーレイ・イメージの表示**

アイコンにオーバーレイ・イメージがある場合、その表示または非表示を切り替えます。たとえば、プロセスの「開始」アクティビティと「終了」アクティビティには、それぞれ緑と赤の矢印のオーバーレイ・イメージがあります。

**「ウィンドウ」メニュー**

「ウィンドウ」メニューには、開いているすべてのアプリケーション・ウィンドウの名前が表示されます。ウィンドウ名を選択して、ウィンドウをアクティブにします。また、次のメニュー項目も選択できます。

**重ねて表示**

開いているウィンドウがカスケード表示（重ねて表示）されます。

**並べて表示**

開いているウィンドウがタイル表示（並べて表示）されます。



## 「ヘルプ」メニュー

「ヘルプ」メニューを使用すると、Oracle Workflow Builder の使用方法に関するヘルプを表示できます。

### 目次

Oracle Workflow Builder の使用方法のヘルプを表示します。

### Oracle Workflow Builder のバージョン情報

Oracle Workflow Builder の現行バージョンとアクセス・レベルを表示します。「アクセス・レベル」フィールドでアクセス・レベルを編集し、「OK」を選択して変更を適用することもできます。

## Oracle Workflow Builder のツールバー

Oracle Workflow Builder では、「ナビゲータ」ウィンドウとプロセス・ウィンドウにツールバーが表示されます。

### ナビゲータのツールバー

ナビゲータのツールバーには、次のボタンが含まれ、ナビゲータ・ツリーから選択したオブジェクトにのみ適用されます。



**新規ストア:** ナビゲータ・ツリーに新規のデータ・ストアのブランチが作成されます。



**オープン:** 「オープン」ウィンドウを表示し、ファイルまたはデータベースから保存されている項目タイプを開きます。



**保存:** 選択したデータ・ストアの変更内容が、現在接続中のデータベースまたはファイルに保存されます。選択したデータ・ストアがデータベースやファイルに接続されていない場合は、「オープン」ウィンドウを表示してデータベースやファイルに接続します。



**削除:** 選択したオブジェクトを削除します。



**プロパティ:** 選択したオブジェクトのプロパティ画面を表示します。



**コピー:** 選択したオブジェクトをコピーします。



**貼り付け:** コピーしたオブジェクトを現行オブジェクト・ブランチに貼り付けます。



**検証:** プロセス定義を検証します。



**開発者モード:** 「開発者」と「プレゼンテーション」の表示を切り替えます。



**検索:** 「検索」ウィンドウを表示し、検索基準を指定してナビゲータ・ツリーでオブジェクトを検索します。



**クイック・スタート・ウィザード:** クイック・スタート・ウィザードを実行し、ワークフロー・プロセス定義の作成を開始します。



**ヘルプ:** Oracle Workflow Builder の使用方法のヘルプを表示します。



**新規オブジェクト:** 選択したオブジェクト・ブランチ（項目タイプ、「プロセス」、「通知」、「関数」、「メッセージ」または「選択肢タイプ」）に応じて、そのオブジェクト・タイプのプロパティ画面が表示され、オブジェクトを新規作成します。

## 「プロセス」ウィンドウのツールバー

プロセス・ウィンドウのツールバーには、次のボタンが含まれ、現行のプロセス・ウィンドウから選択したオブジェクトにのみ適用されます。



**オープン:** 「オープン」ウィンドウを表示し、ファイルまたはデータベースから保存されている項目タイプを開きます。



**保存:** 選択したデータ・ストアの変更内容が、現在接続中のデータベースまたはファイルに保存されます。選択したデータ・ストアがデータベースやファイルに接続されていない場合は、「オープン」ウィンドウを表示してデータベースやファイルに接続します。



**ダイアグラムの印刷:** 現行のプロセス・ダイアグラムを印刷します。



**新規プロセス:** プロセス・アクティビティ・ノードのプロパティ画面を表示し、新規プロセス・アクティビティを作成します。



**新規通知:** 通知アクティビティ・ノードのプロパティ画面を表示し、新規通知アクティビティを作成します。



**新規関数:** 関数アクティビティ・ノードのプロパティ画面を表示し、新規関数アクティビティを作成します。



**新規イベント:** イベント・アクティビティ・ノードのプロパティ画面を表示し、新規イベント・アクティビティを作成します。



**選択の削除:** 選択したオブジェクトを削除します。



**プロパティ:** 選択したオブジェクトのプロパティ画面を表示します。



**開発者モード:** 「開発者」と「プレゼンテーション」の表示を切り替えます。



**検索:** プロセスの「概要」ウィンドウを表示します。



**インスタンス・ラベルの表示:** プロセス・ウィンドウのノード・アクティビティ・ラベルとして、インスタンス・ラベルを表示します。



**内部名の表示:** プロセス・ウィンドウのノード・アクティビティ・ラベルとして、ノードの内部名を表示します。



**表示名の表示:** プロセス・ウィンドウのノード・アクティビティ・ラベルとして、ノードの表示名を表示します。



**コメントの表示:** プロセス・ウィンドウのノード・アクティビティ・ラベルとして、ノードのコメントを表示します。



**実行者の表示:** プロセス・ウィンドウのノード・アクティビティ・ラベルとして、ノードの実行者を表示します。



**ヘルプ:** Oracle Workflow Builder の使用方法のヘルプを表示します。

---

## 他の Oracle 製品への Oracle Workflow の実装

付録 B では、Oracle E-Business Suite および Oracle9i プラットフォームに埋め込まれているワークフローとビジネス・イベント・システムについて説明します。また、各ワークフロー、イベントおよびサブスクリプションのカスタマイズに関するオラクル社のサポート・ポリシーについて説明します。

## Oracle E-Business Suite 埋込みの事前定義済のワークフロー

次に挙げる事前定義済のワークフロー・プロセスは、Oracle Workflow を使用してカスタマイズできます。各ワークフローの詳細は、各製品のユーザーズ・ガイドまたは構成ガイドを参照してください。

---

**注意：** Oracle Applications の製品には、Account Generator の機能を使用して、動的に会計フレックスフィールドの組合せを作成するものもあります。Account Generator には、あらかじめ定義されている汎用的なワークフロー関数が用意されており、それぞれの Oracle Application 製品では、事前に定義した Account Generator のプロセス内でこれを使用します。ここでは、各製品に関する Account Generator のプロセスについては記載していません。詳細は各製品のユーザーズ・ガイドに記載されています。Account Generator 機能の一般的な内容については、『Oracle Applications Flexfields Guide』を参照してください。

---

### 関連項目：

[B-16 ページ「定義済のワークフロー、イベントおよびサブスクリプションに関するオラクル社のサポート・ポリシー」](#)

### Application Implementation Wizard

Application Implementation Wizard には、一連のワークフロー・プロセスが表示されるため、指示に従って Oracle Applications を設定し、実装できます。また、このウィザードを使用すると、各タスクを実行し、インストールに合せて相互に依存する Oracle Applications を構成できます。実装作業を容易に進行できるように、Application Implementation Wizard では類似する設定タスクが論理グループにまとめられています。

ウィザードに表示される手順の順序は、実装するアプリケーション・モジュールによって決まります。このため、複数のアプリケーション・モジュールを実装する場合に、重複する設定手順を実行する必要がなくなります。

ワークフロー・プロセスの詳細と使用方法は、『Application Implementation Wizard ユーザーズ・ガイド』を参照してください。

### Oracle Application Object Library

Oracle Application Object Library には、コンカレント・マネージャの処理を Oracle Applications のワークフロー・プロセスに統合するときに使用できるように、一連の「標準」関数アクティビティが用意されています。これらの「標準」関数アクティビティは、コンカレント・マネージャ関数の項目タイプに関連付けられています。6-20 ページの「[コンカレント・マネージャの標準アクティビティ](#)」を参照してください。

## Oracle Business Intelligence System

**BIS 例外管理：**汎用的なワークフロー・プロセスで、BIS カスタマが Performance Management Framework の一部として使用するためのテンプレートです。実際のパフォーマンスが、要求するパフォーマンス要件を満たさない場合は、このプロセスによって、埋込みレポート URL を使用した基本的な処置の通知が送信されます。OBIS Corrective Action の項目タイプの他のすべてのプロセスは、BIS 例外管理に類似しています。

## Oracle Internet Expenses

**AP Expense Report プロセス：**Oracle Internet Expenses では、AP Expense Report Workflow プロセスを使用し、Internet Expenses に入力された経費精算書に対してマネージャの承認および経理担当者の検査を行います。AP Expense Report プロセスは、ユーザーが経費精算書を送信した時点で起動され、経費精算書が否認された時点または経費精算書がマネージャによって承認され経理担当者によって検査された時点で終了します。承認および検査された場合は、AP Expense Report プロセスによって、経費精算書が買掛 / 未払金請求書インポートプログラムで使用できるようになります。AP Expense Report プロセスは、マネージャの承認および経理担当者の検査の際のキー・イベントで、従業員に通知を行います。

**AP Credit Card プロセス：**AP Credit Card Workflow プロセスでは、従業員に対して作成された支払、および T&E (travel and entertainment) クレジット・カードの請求額としてクレジット・カード発行者に対して作成された支払を、従業員およびマネージャに通知します。また、未処理のクレジット・カード取引のうち、経費計算書に送信されていないもの、または経費計算書に送信されているが経費計算書が AP Expense Report プロセスを完了していないものを、従業員およびマネージャに通知を送信します。支払通知は、支払が Oracle Payables で作成されたときに生成されます。未処理の請求通知は、Credit Card Outstanding Charges Report プログラムを Oracle Payables から実行したときに送信されます。

## Procurement Cards

次の調達カード・ワークフローでは、セルフサービス従業員が調達カード取引の検証および承認を行うことができます。

**AP Procurement Card Employee Verification ワークフロー・プロセス：**AP Procurement Card Employee Verification ワークフロー・プロセスでは、カード所有者との調達取引を通知および確認します。このワークフロー・プロセスは、Oracle Payables から Distribute Employee Card Transaction Verifications プログラムを送信したときに開始されます。このプロセスでは、従業員の調達カードに請求された取引をその従業員に通知し、オプションで従業員による検証を要求します。

**AP Procurement Card Manager Approval Transaction プロセス：**AP Procurement Card Manager Approval Transaction ワークフロー・プロセスでは、検証済のカード所有者との調達カード取引を通知および確認します。このワークフロー・プロセスは、Oracle Payables から Distribute Manager Card Transactions Approvals プログラムを送信したときに開始されます。マネージャ・ワークフロー・プロセスは、AP Procurement Card Employee Verification ワークフローが取引を完了していないと、使用できません。このプロセスでは、

従業員による調達カード取引をマネージャに通知し、オプションでマネージャによる承認を要求します。

## Oracle Self-Service Human Resources

**応募者内定承認プロセス：**「ライン・マネージャ・ダイレクト・アクセス」の「応募者内定」オプションを使用して提示された内定が、承認階層の該当するマネージャに送信されます。承認階層の最後の承認者が内定を承認すると、内定通知を印刷、署名し、応募者に送信して応募者の応答を待つようにという通知が、ワークフローによって人事部門に送られます。応募者が内定に応答すると、元のマネージャに通知され、ワークフローが完了します。このプロセスの実行中は、内定のステータスが元のマネージャに絶えず通知されます。

**キャリア・マネージメント・レビュー・プロセス：** レビュー担当者に「評価と査定」の通知が送信されます。

**360 度評価プロセス：** 特定の人に、グループの一員として評価を行う必要があるという通知を送信します。

**その他のプロセス：** Oracle Self-Service HR には、従業員およびマネージャが、承認された個人情報の詳細を Web 上で参照、更新および表示するプロセスが含まれています。具体的には、次の情報が対象となります。

- 基本詳細（名前、婚姻区分など）
- 住所
- 電話番号
- 連絡先
- 履歴書
- 資格
- 個人コンピテンス・プロフィール
- 学校と大学就学
- 勤務選択

Oracle Self-Service HR には、従業員がクラスに登録するプロセス、および応募するプロセスも含まれています。

## Oracle Self-Service Purchasing

**入金確認プロセス：** 申請者に対して、それぞれの申請が受理されたことを表す受領通知を送信します。このプロセスは、発注受入確認ワークフローとしても機能します。

**「購買承認申請」プロセス：** Web 購買から作成された購買申請が承認のために該当するマネージャに送信され、購買申請のステータスが更新されます。



## Oracle Internet Time

**PA Timecard Approval:** このプロセスは、従業員が Oracle Internet Time でタイムカードを送信したときに開始されます。このワークフローは、すべてのタイムカードを自動的に承認するか、事前に決定された承認プロセスを介してタイムカードを経路指定するように構成できます。この承認プロセスでは、マネージャおよび従業員に通知を送信し、タイムカードが会社の方針に準拠していることを確認し、マネージャ承認レベルをチェックします。送信されたタイムカードのステータスは、すべての承認プロセスにわたって監視できます。

## Oracle Web Suppliers

**Supplier Self-Service 登録承認プロセス:** Oracle Web Suppliers では、ゲストとしてログインしたユーザーを、会社の仕入先担当として登録できます。このプロセスでは、登録の検証と承認のために、該当するアカウント承認者に通知がルーティングされます。承認者が登録を承認すると、「Supplier Web ユーザー」アカウントがアクティブ化されます。アカウント承認者が登録を却下すると、そのアカウントは非アクティブ化されます。

## Oracle Configure To Order (CTO)

**CTO 変更オーダー:** 出荷組織のプランナに ATO オーダーに加えられた変更の詳細を通知します。このプロセスは、オーダー数量、予定出荷日、要求日または予定到着日の変更され、オーダーに対して個別のジョブ予約またはフロー・スケジュールが存在するときに開始されます。また、構成の変更や受注明細のキャンセルが発生したときは、予約が存在しない場合でも開始されます。

## Oracle Demand Planning

**MSD Demand Planning Cycle プロセス:** MSD Demand Planning Cycle では、Demand Planning Cycle のすべてのバックグラウンド処理が管理されます。Demand Planning Cycle は 4 つのステージで構成されています。それぞれのステージで、実行されるタスクを管理するワークフロー・プロセスが開始されます。各ステージは、「Demand Planning Administrator」画面から開始します。ワークフローが各ステージを処理すると、管理者とユーザーに対して関連するプロセスのステータスを通知します。

Demand Planning Processing Cycle は、次の 4 つのステージで構成されており、各ステージはプロセスに対応しています。

- Planning Server のデータをダウンロードする: Demand Planning Server のデータを転送および変換し、そのデータを分析するために Express Server Demand Planning エンジンに渡します。
- 需要プランナが利用するデータを予測し配布する: 予測エンジンを実行して、需要プランナが利用するデータを確保します。
- 需要プランナのデータを収集および統合する: 需要プランナから送信されたデータを収集し、Demand Planning エンジンでデータを統合します。

- 統合したデータを Planning Server にアップロードする： 変換したデータを Planning Server に転送します。

## Oracle Engineering

**設計変更プロセス：** 設計変更が適切な承認担当に送信されます。

## Oracle General Ledger

**仕訳承認プロセス：** 仕訳を転記する前に仕訳バッチの承認を要求できます。承認階層を作成し、ユーザーごとに承認限度を定義します。仕訳承認プロセスは、仕訳バッチを転記したときに開始されます。このプロセスは、承認階層に基づいて、承認を要求するユーザーに仕訳を自動的にルーティングします。

**自動配賦プロセス：** 戻し自動配賦を生成すると、このワークフロー・プロセスによって自動配賦プロセスが開始され、自動配賦に定義されている一括配賦または定型仕訳バッチが検証および生成されます。また、生成された各バッチに仕訳承認が必要かどうかを確認され、必要に応じて承認を要求するユーザーにバッチが送信され、承認結果が該当するユーザーに通知されます。自動配賦プロセスでエラーが発生した場合は、指定されたユーザーが、自動配賦プロセスをロールバックし、転記された仕訳を取り消すことができます。

**グローバル会社間システム：** グローバル会社間システム（以前の CENTRA）は、リリース 11i の拡張機能で、リリース 11 にバックポートされています。グローバル会社間システムは、複数の会社間の取引を交換するための環境を提供します。送信側の会社が会社間取引を開始して受信側の会社に承認を要求するか、送信側の会社が会社間取引を取り消すか破棄すると、ワークフロー・プロセスによって受信側の会社へ通知されます。送信側の会社によって開始された会社間取引を受信側の会社が承認または却下すると、送信側の会社へ通知されます。また、しきい金額を設定して、通知量を制限することもできます。このワークフロー・プロセスは、送信側の会社が会社間取引の送信、取消または破棄を行うか、受信側の会社が会社間取引を却下または承認したときに開始されます。

## Oracle Grants Accounting

**Grants Accounting ワークフロー・プロセス：** Grants Accounting ワークフロー・プロセスでは、賦払いが選択されるかレポートが必要なことを主要なメンバーに通知します。Budget サブプロセスでは、予算が送信され承認を必要としていることを、予算承認者または裁定マネージャに通知します。

このワークフロー・プロセスは、次の時点で開始されます。

- 賦払いが選択されたとき
- レポートが必要なとき
- 予算が送信されたとき
- 予算が承認 / ベースライン化されたとき

## Oracle Grants Proposal

**Proposal Approval プロセス：** Proposal Approval プロセスは、提案を送信して承認を要求したときに開始されます。

通知が承認者に送信されると、このワークフロー・プロセスは、その承認者からの回答を取得してから、提案承認階層マップ内の次の承認者に進みます。

すべての承認者が提案を承認すると、その提案は承認されます。いずれかの承認者が提案を却下すると、その提案は却下されます。提案を送信して承認を要求するユーザーには、承認プロセスのステージ単位に承認ステータスを通知します。

**Notify Approval サブプロセス：** Proposal Approval プロセスの Notify Approval サブプロセスは、承認階層マップ内の次の承認者が選択されているときに開始されます。

Notify Approval サブプロセスでは、承認を必要とする提案が保留中であることを承認者に通知します。承認者は、提案を承認または却下します。

承認者が所定の時間内に提案を承認または却下しなかった場合、その承認者は定期的に督促を受け取ります。組織は、タイムアウトを設定して、督促を送信する期間を定義します。デフォルトでは、タイムアウトは設定されていません。

**Notify Proposal Members プロセス：** Notify Proposal Members プロセスでは、提案の担当者には通知を送信します。

## Oracle Labor Distribution

**Effort Report Notification プロセス：** Labor Distribution のワークフロー機能は、成果レポートを組織全体に自動的にルーティングし、注意が必要な成果レポートまたは完了しているプロセスに関して、ユーザーに電子通知を配信します。

Effort Report Notification ワークフロー・プロセスには、次のサブプロセスが含まれます。

- 承認
- 通知

Effort Report Notification ワークフロー・プロセスは、成果レポートが作成されたときに、Labor Distribution で開始されます。

通知は、成果レポートの承認者に送信します。成果レポートが承認されると、成果レポートは管理者に送信し、証明を要求します。成果レポートの作成者は、成果レポートのステータスを監視できます。

**Distribution Adjustment Approval Notification プロセス：** Labor Distribution のワークフロー機能は、Distribution Adjustment Approval Notification を組織全体に自動的にルーティングし、注意が必要な配布調整または完了しているプロセスに関して、ユーザーに電子通知を配信します。このプロセスは、配布バッチが送信されたときに開始されます。

## Oracle Public Sector Budgeting

**Distribute Worksheet Workflow プロセス：** Distribute Worksheet Workflow プロセスでは、ワークシートを配布し、その事実をユーザーに通知します。このプロセスは、ワークシートが配布するときに開始します。

**Submit Worksheet Workflow プロセス：** Submit Worksheet Workflow プロセスでは、ワークシートを送信します。ユーザー定義のパラメータに基づいて、制約の検証、ワークシートの処理、コピーおよびマージを実行します。ワークシートは、予算ステージを経過してから、必要な承認を得るために承認プロセスにルーティングされます。また、このプロセスは、ワークシートの凍結および凍結解除も行います。通知は、プロセスを開始するユーザーおよび承認者に送信します。

このプロセスは、次の時点で開始されます。

- ワークシートの制約を検証するとき
- ワークシートを凍結するとき
- ワークシートの凍結を解除するとき
- ワークシートを次のステージに移動するとき
- ワークシートをコピーするとき
- ワークシートをマージするとき
- ワークシートを送信するとき

**Distribute Budget Revision Workflow プロセス：** Distribute Budget Revision Workflow では、予算改訂を配布し、その事実をユーザーに通知します。このプロセスは、予算改訂を配布するときに開始されます。

**Submit Budget Revisions Workflow:** Submit Budget Revisions Workflow プロセスでは、予算改訂を送信します。ユーザー定義のパラメータに基づいて、制約の検証およびその他の予算改訂処理を実行します。Submit Budget Revisions プロセスでは、予算改訂を承認プロセスにルーティングし、予算改訂のステータスおよび基準値を更新します。また、予算引当および総勘定元帳への改訂の転記も行います。予算改訂の凍結および凍結解除も実行します。通知は、プロセスを開始するユーザーおよび承認者に送信します。

このプロセスは、次の時点で開始されます。

- 予算改訂の制約を検証するとき
- 予算改訂を凍結するとき
- 予算改訂の凍結を解除するとき
- 予算改訂を送信するとき

## Oracle Federal Human Resources

**GHR 人事アクション・プロセス：** データ・エントリの人事申請処理 RPA フォームをルーティングできるようにし、署名、および最終的な承認前の検討を行って、データベースを更新します。ユーザーは、エージェントの実行に基づいて、個人、グループボックスまたはルーティング・グループのルーティング・リストに RPA を送信できます。RPA がルーティングされると、処理の履歴が記録されます。履歴を参照すると、実行された処理、実行者および実行日がわかります。

**GHR 職階の摘要プロセス：** データ・エントリの職階摘要フォームをルーティングできるようにし、署名、検討および分類を行います。ユーザーは、エージェントの実行に基づいて、個人、グループボックスまたはルーティング・グループのルーティング・リストに職階摘要フォームを送信できます。職階摘要フォームがルーティングされると、処理の履歴が記録されます。履歴を参照すると、実行された処理、実行者および実行日がわかります。

**GHR 等級変更プロセス：** 職階変更 (WGI) の処理を、手動の作業を介さずに自動化できるようにします。デフォルトの WGI プロセスは、WGI が承認されたことを Personnel Office に自動的に通知するため、ユーザーは応答する必要がありません。WGI プロセスは、導入の際にエージェントの実行に基づいて様々な方法で設定できます。

## Oracle Human Resources

**タスク・フロー項目タイプ：** Oracle Human Resources には、あらかじめ定義されているワークフロー (HR Task Flow) の項目タイプが用意されており、これを使用してタスク・フローを設定できます。HR Task Flow の項目タイプには、HR アプリケーションの各ウィンドウの関数アクティビティが含まれており、このアクティビティによって 1 つのタスク・フローに統合できます。これらの定義済の関数アクティビティを使用して、それぞれのタスク・フローに対するワークフロー・プロセスをモデル化できます。また、関数アクティビティにはアクティビティ属性が含まれており、この属性の設定によって、対象のアプリケーション・ウィンドウで、ボタン・ラベルの作成やボタンの位置定義を行うことができます。

HR Task Flow の項目タイプでは、タスク・フローを設定および管理するためのフォームを使用することもできます。Oracle Workflow と統合すると、グラフィカルな Oracle Workflow Builder を使用し、一連のウィンドウを簡単に設計および作成できます。

## Oracle Order Management

**Order and Line Runnable プロセスおよび Functional サブプロセス：** Oracle Order Management には、Seeded Order and Line Runnable プロセスおよび Functional サブプロセスが用意されています。注文ヘッダー・ワークフロー・データは、項目タイプ「OM 注文ヘッダー (OECH)」に定義されています。シード済ヘッダー実行可能プロセスは、標準の承認のある受注と返品を処理するために用意されています。Booking and Close Order 機能サブプロセスもシードされています。Order Line ワークフロー・データは、項目タイプ「OM 受注明細 (OEOL)」に定義されています。Oracle Order Management には、明細レベルの実行可能プロセスが用意されており、標準項目、構成、サービス項目、直送およびその他の項目を処理することができます。受注明細のスケジュール、出荷、履行、請求書インタフェースおよびクローズに関する機能サブプロセスもシードされています。また、シード済

機能サブプロセスおよびカスタム・アクティビティを使用して、特定のビジネス要件に合わせてカスタム・プロセスを構成することもできます。シード済プロセスおよびカスタム実行可能プロセスは、特定の受注および明細取引タイプに割り当てることにより、オーダー処理に使用できます。

**Change Order Notification:** 特定の申請フォームからオーダー変更通知を送信します。職責を選択し、内容を指定すると、受信者およびメッセージの内容が動的に設定されます。項目タイプ「OM 発注変更 (OECHORD)」を使用します。

**COGS:** 項目タイプ「Generate Cost of Goods Sold Account (OECOGS)」を使用して、売上原価勘定を生成します。

## Oracle Payables

**AP オープン・インタフェース・インポート・プロセス:** オープン・インタフェース表のデータの検証が自動化されます。たとえば、このプロセスを修正し、オープン・インタフェース表内ですべての会計コード組合せを検証できます。無効なコード組合せがある場合は、その通知を指定したユーザーに送信して訂正させることができます。必要であれば、無効なコード組合せを明示したデフォルト値で上書きするように、このプロセスを設定できます。Oracle Workflow を使用すると、特定の業務要件にあわせてワークフロー・ルールを追加することも可能です。請求書は、このプロセスを通過すると、Oracle Payables Application の表にインポートする準備ができたこととなります。オープン・インタフェース・インポート・プロセスを開始するには、「要求の発行」ウィンドウから「Payables Open Interface Workflow」を発行します。

経費精算書および調達カードを含む、Oracle Internet Expenses の関連ワークフローの詳細は、B-3 ページの「[Oracle Internet Expenses](#)」を参照してください。

## Oracle Advanced Supply Chain Planning

**事前計画例外メッセージ・プロセス:** 仕入先、得意先の担当者または社内の担当者に対して、事前計画の例外を通知し、通知を受信した担当者が、対象の計画例外に対して適切な処置を開始できるようにします。

**Allocated ATP:** 注文予約要求を満たすために異なる供給ソース間で補充が発生した場合または注文スケジュール・プロセスが失敗した場合に、プランナに通知を送信します。

## Oracle Projects

**プロジェクト承認とステータス変更プロセス:** プロジェクトがルーティングされ、該当するユーザーにプロジェクト・ステータスの変更が通知されます。たとえば、プロジェクトを承認処理のために送信したり、プロジェクトの完了時に該当する担当者に通知できます。該当するステータスの変更のみでなく、プロジェクトのルーティング先となる担当者の決定に使用するワークフローを選択します。

**予算承認プロセス:** プロジェクト予算が承認処理とベースライン化のためにルーティングされます。予算タイプのみでなく、予算のルーティング先となる担当者の決定に使用するワークフローを選択します。

**Step Down Allocations プロセス：** 戻し処理自動配賦セットの実行を自動化します。つまり、配賦実行の作成、配賦トランザクションの生成、配賦トランザクションの解放（プロセスの続行に必要な承認の要求）、コストの配布、およびプロジェクト要約金額の更新を行います。

## Oracle Project Manufacturing

**間接 / 資本プロジェクト定義プロセス：** このプロセスは、プロジェクト製造プロジェクト定義プロセスのナビゲータ・フローの一部として機能します。このプロセスでは、Oracle Project Manufacturing で使用する間接または資本タイプのプロジェクトを設定するときに必要な一連の手順の説明が表示されます。

**契約プロジェクト定義プロセス：** このプロセスは、プロジェクト製造プロジェクト定義プロセスのナビゲータ・フローの一部として機能します。このプロセスでは、Oracle Project Manufacturing で使用する契約タイプのプロジェクトを設定するときに必要な一連の手順に関する説明が表示されます。

## Oracle Process Manufacturing

**品質管理サンプル作成通知プロセス：** 特定のトランザクションのパラメータ（Organization、Warehouse、Item など）に関連付けられている適切なユーザーに対して通知を行い、Oracle Process Manufacturing の Product Development Module に対して品質保証のためのサンプルを作成するように要求します。このワークフロー・プロセスでは、Oracle Process Manufacturing の特別な在庫トランザクションが開始されます。ユーザーは、通知からサンプル作成フォームを直接起動して、品質管理のサンプルを作成できます。

**品質管理サンプル検収プロセス：** 品質管理分析者に対して、新しく作成されたサンプルでテストを実施するように通知する詳細プロセスを生成し、最終的なサンプル検収のテスト結果を管理します。このワークフローは、Oracle Process Manufacturing の Product Development Module でサンプルが作成されたときに開始されます。このワークフローはマスター・プロセスとして機能し、あらかじめ定義された仕様に基づいてサンプルで実行する一連のテストを決定し、テストを実行するための分析を通知する Quality Control Assay Testing の詳細プロセスの該当番号を導出します。このマスター・プロセスは、すべての詳細プロセスが、サンプル承認者に対してサンプル処置の通知を送信するまで待機します。この通知によって、承認者は、結果フォームから結果を直接参照し、サンプルに対して最終的な処置を行うことができます。このプロセスは、在庫承認者に対して最終的なサンプル処置が通知されたタイミングで終了します。

**品質管理分析テスト・プロセス：** 品質管理分析者に対して、新しく作成されたサンプルでテストを実行するように通知します。このプロセスは、品質管理サンプル検収プロセスによって開始されます。ここでは、テストを実行する必要のある分析者に対して通知が送信されます。分析者は、この通知から結果フォームを直接開いて通知に応答し、テストの結果を入力できます。

**品目有効化プロセス：** 承認者に対して、Oracle Process Manufacturing で作成された品目を承認するように通知します。この項目は、承認者によって承認されるまで無効になっています。

**ロット期限切れおよびロット再テスト・プロセス：** 品目のロットまたはサブロットが失効日に達したとき、または品目の再テストの準備ができたときに、品目に関連付けられている対象ロールに通知します。

## Process Manufacturing Intelligence

**プロセス製造在庫回転率プロセス：** 在庫品の回転の実績値が、在庫回転率レポートで定義されている目標値に満たない場合に、指定された責任者に対して通知が送信されます。在庫回転率レポートは、Process Manufacturing BIS で使用されます。

## Oracle Purchasing

**調達ワークフロー：** 調達ワークフローは、サプライ・チェーンのメンバー全員のための、完全に自動化された柔軟で拡張性のあるトランザクション処理システムです。このワークフローは、より戦略的な部品外注および調達アクティビティに対する主要な実現手段の一つです。このワークフローは、文書承認、自動文書作成、変更オーダー、勘定科目生成、通知の送信、価格 / 販売カタログ通知および入金確認 (Self-Service Purchasing のみで使用) のワークフロー・プロセスによって構成されます。

**文書承認プロセス：** Oracle Purchasing の承認に関するすべてのアクティビティを実行します。具体的には、文書の発行、承認、転送、承認通知、却下などがあります。このプロセスには、購買発注を承認するための発注承認ワークフロー・プロセスと、購買申請を承認するための発注依頼の承認ワークフロー・プロセスが含まれています。

**自動文書作成プロセス：** 購買申請明細に必要なソース情報が含まれている場合は、承認済の購買申請明細を使用して、包括契約に対する標準発注または発注リリースが自動的に作成されます。このプロセスは、発注文書作成ワークフローとも呼ばれます。

**変更オーダー・プロセス：** 変更によって、手動による再承認が必要であるか、自動的に再承認されるかを制御します。手動または自動で再承認されたすべての文書では、文書の改訂を表す番号が増加します。このプロセスは、発注承認ワークフローの一部として機能します。

**通知の送信プロセス：** 不完全な文書、却下された文書または再承認が必要な文書をロックし、この文書ステータスに関する通知を該当メンバーに通知します。このプロセスは、購買文書ワークフローの発注送信通知としても機能します。

**価格 / 販売カタログ通知プロセス：** 購買文書オープン・インタフェースを介して送信された価格 / 販売カタログ情報に、設定した金額範囲を超える金額の増分が検出された場合に、購買担当に対して通知を送信します。このプロセスは、発注カタログの価格許容範囲超過通知ワークフローとしても機能します。

## Oracle Receivables

**クレジット・メモ申請承認プロセス：** 組織の内部管理階層、または Oracle Receivables で定義されている承認制限を使用して、承認するクレジット・メモの申請をルーティングします。この申請が承認されると、Oracle Receivables にクレジット・メモが自動的に作成されます。申請が却下された場合は、申請者に対して承認されなかった理由を通知します。



クレジット・メモ申請ワークフローは、iReceivables から開始します。iReceivables は、Web ベースのセルフサービス・アプリケーションです。iReceivables に登録されたユーザーは、標準 Web ブラウザを使用して、Receivables アカウント情報にアクセスできます。iReceivables ユーザーが「Dispute a Bill」を選択すると、Receivables によって、指定した額が係争中になり、クレジット・メモ申請プロセスが開始され、申請が承認プロセスに経路指定されます。

## Oracle Service

**サービス要求プロセス：** サービス要求が組織内の個人にルーティングされ、処理されます。このプロセスをカスタマイズして、サービス担当を選択して通知するのみでなく、組織のサービス・ルールとガイドラインに基づいてサービス要求を自動的に転送し、エスカレートさせることができます。

**サービス要求処理と手配プロセス：** サービス要求処理が解決のために組織内の個人にルーティングされるだけでなく、現場で作業手配が必要な該当のサービス担当に指示が通知されます。このプロセスをカスタマイズして、作業手配要求を管理、転送またはエスカレートさせることができます。

**フィールド・サービス作業手配プロセス：** サービス要求データをインタフェース表に対して挿入または更新し、フィールド・サービス・エンジニアに対して、ディスパッチ情報の通知を送信します。このプロセスは、Oracle Mobile Field Service で使用されます。

## Oracle Training Administration

**OTA Workflow プロセス：** Self-Service Oracle Training Administration のワークフローおよび注文管理を介した Training Administration のワークフローで構成されます。

- Self-Service Oracle Training Administration
  - 既存のイベントに空きがあるかどうかを確認し、待機リストへの登録または配置を受講者に通知します。
  - イベントの直前に登録の取消しが発生した場合は、コストの転送が発生し、顧客に登録費用を請求します。
  - Oracle Training Administration が受注明細を自動作成するように設定されているときは、「転送元」または「転送先」の値が見つからない場合に、イベント所有者に通知します。
  - 登録申請の作成者に登録ステータスの変更を通知します。
  - 登録者および管理者に登録の取消しを通知します。
- 注文管理を介した Training Administration
  - 受注明細に対する請求を、受講者がコースを完了した後で生成します。
  - イベントまたは登録の取消しを受講者に通知します。
  - イベントの所有者に登録の取消しを通知し、待機リストの受講者を登録します。

- イベントの最大受講者数が増加したときは、イベントの所有者に通知します。
- 登録しているイベントを顧客が変更したときは、イベントの所有者に通知します。

## Oracle E-Business Suite への Oracle Workflow ビジネス・イベント・システムの実装

次に挙げる製品では、Oracle Workflow ビジネス・イベント・システムを利用してビジネス・プロセスが統合されます。各機能の詳細は、各製品のユーザーズ・ガイドまたは構成ガイドを参照してください。

### 関連項目：

B-16 ページ「[定義済のワークフロー、イベントおよびサブスクリプションに関するオラクル社のサポート・ポリシー](#)」

### Oracle Payables

**電子メール送金通知プログラム：** 支払バッチを確定するかクイック支払を作成すると、ビジネス・イベント・システムによってこのプログラムが開始され、送金通知電子メール・アドレスが定義されている各サプライヤに電子メールが自動的に送信されます。

### Oracle XML Gateway

Oracle XML Gateway は、Oracle Workflow ビジネス・イベント・システムを利用して、メッセージの作成または取込みを自動的に行うアプリケーション・ビジネス・イベントの発行およびサブスクライブを行います。XML Gateway 実行エンジンと直接対話しながらアウトバウンド・メッセージの生成やインバウンド・メッセージの取込みを行う場合は、事前定義されたワークフロー関数をワークフロー・プロセスで使用できます。実行エンジンが生成したアウトバウンド・メッセージは、ダウンストリーム・ワークフロー・アクティビティで処理できます。実行エンジンは、ワークフロー・プロセスから渡されたインバウンド・メッセージを取り込みます。

XML Gateway では、XML Gateway 標準項目タイプと XML Gateway エラー・プロセス項目タイプが提供されます。

**XML Gateway 標準項目タイプ：** XML Gateway 標準項目タイプには、文書配信の呼出しイベントが含まれます。このイベントは、既存のワークフロー・プロセスからビジネス・イベントを呼び出すときに使用します。これにより、既存のワークフロー・プロセスと Oracle XML Gateway をシームレスに統合して、アウトバウンド XML メッセージを作成できます。XML Gateway 標準項目タイプに含まれる関数は、XML 文書の取込み、XML 文書の生成、取引先 XML 文書の生成、文書の送信、XML の変換およびトランザクション・デリバリーの必要性の判別です。

企業間またはアプリケーション間の統合を行うときは、既成の XML メッセージに事前定義されたイベントおよびイベント・サブスクリプションが Oracle E-Business Suite から配信されるように構成します。

**XML Gateway (ECX) エラー・プロセス項目タイプ:** XML Gateway エラー・プロセス項目タイプに含まれるエラー・プロセスは、Oracle Workflow ビジネス・イベント・システムまたは Oracle XML Gateway によって検出されたエラーを管理します。エラー・プロセスには、デフォルト・エラー・プロセス、ECX エンジン通知プロセス、ECX メイン・エラー・プロセス、ECX メイン・インバウンド・エラー・プロセス、ECX メイン・アウトバウンド・エラー・プロセス、インバウンド・メッセージ用エラー・プロセスおよびアウトバウンド・メッセージ用エラー・プロセスがあります。

XML Gateway エラー・プロセス項目タイプでは、エラー・イベントの受信イベント・アクティビティと送信通知イベントの受信イベント・アクティビティがサポートされます。エラー・イベントの受信は、XML Gateway 実行エンジンがエラーを検出したときに XML Gateway によって使用されます。送信通知イベントの受信は、インバウンド・プロセスに関連するエラーを実行エンジンから通知しなければならないときに、XML Gateway によって使用されます。

Oracle Workflow のエラー・プロセスでは、XML Gateway のシステム管理者または取引先に対してエラー通知が有効になっており、ワークフローの再試行および再処理機能を使用できます。XML Gateway エラー・プロセス項目タイプによって提供される関数は、ECX 再処理インバウンド、ECX 再送信アウトバウンド・メッセージ、ECX In Error 詳細の取得、ECX Out Error 詳細の取得、システム管理者ロールの取得および取引先ロールの取得です。

詳細は、『Oracle XML Gateway User's Guide』を参照してください。

## Oracle9i プラットフォームへの Oracle Workflow の実装

次に挙げる製品では、Oracle Workflow を利用してビジネス・プロセスが定義および統合されます。各機能の詳細は、各製品のユーザーズ・ガイドまたは構成ガイドを参照してください。

### 関連項目:

B-16 ページ「定義済のワークフロー、イベントおよびサブスクリプションに関するオラクル社のサポート・ポリシー」

### Oracle Warehouse Builder

Oracle Warehouse Builder には Workflow 配布ウィザードが組み込まれています。このウィザードを使用すれば、マッピングの抽出、変換およびロードを項目タイプ内の関数として Oracle Workflow に配置できます。これらの関数は、Oracle Workflow Builder を使用してワークフロー・プロセスとして定義することができます。ワークフロー・プロセスを設計するときに、ジョブが適切な順序で実行されるように、ジョブの依存関係をマッピング間に指定できます。このワークフロー・プロセスは、Oracle Workflow から実行したり、Oracle Enterprise Manager を使用してスケジュールすることができます。

ジョブをワークフロー・プロセスとして定義することにより、一連のジョブのスケジュール全体を自動化することができます。プロセスの実行時は、プロセスに定義されている順序でジョブが実行されるように Oracle Workflow が管理します。例外が発生した場合、Oracle

Workflow はプロセスを終了します。この方法を使用すれば、ウェアハウスの負荷管理およびジョブの更新に必要な手動操作を最小限に抑えることができます。

詳細は、『Oracle Warehouse Builder ユーザーズ・ガイド』の「ロードおよび更新の管理」を参照してください。

## 定義済のワークフロー、イベントおよびサブスクリプションに関するオラクル社のサポート・ポリシー

Oracle Applications には Oracle Workflow が埋め込まれており、それぞれのモジュールで Oracle Workflow を使用し、ビジネス・プロセスを自動化して効率よく作業できます。Oracle Workflow Builder を使用すると、アプリケーション・コードを修正せずに、既存のビジネス・プロセスを簡単に変更できます。Oracle Workflow を使用すると、実際のニーズにあわせてビジネス・ルールが変更された場合にも、ワークフロー・プロセスを拡張できます。また、イベント・マネージャを使用すると、アプリケーション・コードを修正しないでイベント定義およびサブスクリプション定義を変更することができます。

Oracle Workflow を使用して定義済のワークフロー・プロセス、イベントまたはサブスクリプションをカスタマイズする前に、次の「カスタマイズのガイドライン」をよく読み、事前に標準設計、安全な設計および実際の開発について理解しておいてください。これらのガイドラインに従うと、カスタマイズで問題が発生した場合に、オラクル社カスタマ・サポート・センターに対して、支援に必要な重要な情報を提示できます。

### カスタマイズのガイドライン

1. 『Oracle Workflow ガイド』およびその他の製品固有のユーザーズ・ガイドに記載されているとおりに、すべての設定手順が完了していることを確認します。
2. シードされる未修正のワークフロー、イベントまたはサブスクリプションを、テスト・データベースでテストし、各自の環境に特有の設定およびデータを使用して、正常に稼働することを確認します。
3. ワークフロー、イベントまたはサブスクリプションの固有な問題については、製品固有のユーザーズ・ガイド、および MetaLink で利用できるドキュメントの改訂を参照してください。これらのドキュメント・ソースでは、特に「不可」について記載されています。オラクル社カスタマ・サポート・センターでは、「UNSUPPORTED」と明示されているオブジェクトに関する修正については、サポートしていません。
4. 手順ごとにカスタマイズを行い、各手順が終了した時点で、カスタマイズしたワークフローまたはサブスクリプションをテストします。
5. PL/SQL プロシージャを作成する場合は、『Oracle Workflow ガイド』に記載されている標準の PL/SQL API テンプレートを確認します。エラーの発生時には必ず例外を処理するようにして、エラーが発生したプロシージャに戻れるようにします。
6. カスタマイズしたワークフロー、イベントまたはサブスクリプションが、テスト・データベースで正常に実行できることを確認せずに、本稼働用のデータベースに実装しない

でください。テスト・データベースは、本稼働用の設定の特定部分のみが複製されているためです。

## カスタマイズに関する問題点の解決

ワークフロー、イベントまたはサブスクリプションのカスタマイズ中に問題が発生した場合は、次のようにします。

- サポート分析者に、修正したワークフローの定義ファイル、イベント定義またはサブスクリプション定義を提示し、可能な場合は、問題が発生した手順を識別します。
- サポート分析者に、修正していないワークフローまたはサブスクリプションを実行した結果を報告します。

## サポート対象外のカスタマイズ

次のタイプのカスタマイズは、サポート対象外となります。

1. 保護レベルが 100 未満のワークフロー・オブジェクトは修正できません。
2. 元の保護レベルが 100 未満の場合、そのワークフロー・オブジェクトの保護レベルは変更できません。
3. 100 未満のレベルで保護されているワークフロー・オブジェクトを修正するために、アクセス・レベルを、未定義 (100 未満) のレベルに変更することはできません。
4. 対象ワークフローの製品固有のユーザーズ・ガイドまたはドキュメントの改訂ノートに、明確に「UNSUPPORTED」と記載されているカスタマイズを行うことはできません。具体的には、「UNSUPPORTED」と記載されているプロセス、属性、関数アクティビティ、イベント・アクティビティ、通知、選択肢タイプ、メッセージ、イベントまたはサブスクリプションは修正できません。
5. 接頭辞 WF\_ または FND\_ を付加して、ワークフローの表を手動で修正することはできません (『Oracle Workflow ガイド』に記載されている場合、またはオラクル社サポート・サービスで必要な場合は除きます)。

## サポート対象のカスタマイズ

次のタイプのカスタマイズは、サポート対象となります。

1. ワークフロー、イベントまたはサブスクリプションの製品固有のユーザーズ・ガイド、またはドキュメントの改訂ノートで、必須とされているカスタマイズ。
2. 製品固有のユーザーズ・ガイドまたはドキュメントの改訂ノートに記載されているカスタマイズ例。ここで発生する問題点は、記載されているとおりにカスタマイズ例が実行される範囲までは、完全にサポート対象となります。記載とは異なる環境によって発生する問題は、カスタム開発の問題となり、さらに評価する必要があります。次の「3」を参照してください。
3. カスタマイズがサポート対象外か、必須か、またはサポート対象のカスタマイズ例かについて明確ではない場合は、サポートの範囲は次のようになります。まず、ユーザーが前述のカスタマイズのガイドラインに従って問題を切り分けます。評価の結果、切り分けられた問題点が **Oracle** 製品によるものであるとオラクル社カスタマ・サポート・センターがみなした場合は、オラクル社より解決策が提示されます。**Oracle** 製品による問題ではない、カスタム開発に対する問題解決は、ユーザーの責任において行ってください。

---

## Oracle Workflow のパフォーマンスの概念

付録 C では、Oracle Workflow の実行時パフォーマンスを向上させるために使用できる概念と技法について説明します。

## Oracle Workflow のパフォーマンスの概念

Oracle Workflow のパフォーマンスは、複数の要因に依存します。設計したワークフロー・プロセスは、同期プロセス、非同期プロセス、強制同期プロセス、項目属性、メッセージ属性、サブプロセスおよび遅延アクティビティを効果的に使用することにより、パフォーマンスを向上させることができます。また、パーティション化およびパージを使用して、大量のランタイム・データに関連するパフォーマンスの問題を解決することもできます。

### 関連項目：

『Oracle Applications Tuning Handbook』 Andy Tremayne, Steve Mayze  
共著 (Oracle Press, ISBN 0-07-212549-7)

『Oracle9i データベース概要』

## パフォーマンス改善のためのワークフロー・プロセスの設計

効果的なプロセス設計により、ワークフロー・プロセスのパフォーマンスを向上させることができます。

### 同期ワークフロー、非同期ワークフローおよび強制同期ワークフロー

ワークフロー・プロセスの設計時に、同期プロセスとして実行するか、非同期プロセスとして実行するかまたは強制同期プロセスとして実行するかを決定する必要があります。プロセスの設計は、ワークフロー・エンジンからプロセスを開始したコール元のアプリケーションに制御が戻るまでの時間に影響します。

- **同期：** 同期プロセスには、すぐに実行可能なアクティビティのみが含まれます。このため、プロセスは開始から終了まで中断しないで実行されます。ワークフロー・エンジンは、プロセスが完了するまで、コール元のアプリケーションに制御を返しません。同期プロセスの場合、項目属性に書き込まれたプロセスの結果またはデータベースに直接書き込まれたプロセスの結果をすぐにチェックできます。ただし、ユーザーはプロセスが完了するまで待機する必要があります。プロセスの実行に時間がかかる場合、アプリケーションがハングしているように見えます。この場合、プロセスを非同期プロセスに変更する必要があります。
- **非同期：** 非同期プロセスには、フローを中断するアクティビティが含まれます。このため、ワークフロー・エンジンは非同期プロセスの実行を待機します。非同期プロセスを要求するアクティビティには、遅延アクティビティ、応答を必要とする通知、ブロック・アクティビティ、待機アクティビティなどがあります。ワークフロー・エンジンがこれらのアクティビティを検出すると、無期限に待機するのではなく、監査表を適切に設定したうえでコール元のアプリケーションに制御を返します。非同期ワークフロー・プロセスは、再開されるまで未完了の状態になります。プロセスの再開は通常、通知システム、ビジネス・イベント・システムまたはバックグラウンド・エンジンによって行われます。非同期プロセスの場合、ユーザーはプロセスが完了するまで待機する必要がなく、アプリケーションの使用を続行できます。ただし、プロセスの結果は、プロセスが完了するまで確認できません。



- **強制同期：** 強制同期プロセスは、開始から終了まで1つのSQLセッションで完了し、データベース表に対して挿入または更新を行いません。そのため、強制同期プロセスの実行速度は、通常の同期プロセスよりもかなり速くなります。プロセスの結果は、完了時にすぐに確認できます。ただし、監査証跡は記録されません。強制同期プロセスは、監査証跡を記録する必要がなく、特定の結果をすばやく出力したいときに使用します。強制同期プロセスを作成するには、プロセスの項目キーを #SYNCH に設定し、プロセスを設計するときに一定の制限（通知アクティビティを含めないなど）に従う必要があります。

#### 関連項目：

8-3 ページ「ワークフロー・エンジンの概要」

8-14 ページ「同期プロセス、非同期プロセスおよび強制同期プロセス」

## 項目属性

項目タイプ属性はグローバル変数として機能し、ワークフロー・プロセス内のすべてのアクティビティから参照または更新できます。項目属性の数は、作業項目の開始時間に直接影響します。新しい作業項目が作成されると、ワークフロー・エンジンによってすべての項目属性のランタイム・コピーが作成されるためです。このため、項目属性の数は最小限に抑える必要があります。

項目属性は、次の目的で使用します。

- 作業項目に関する作業情報を格納します。
- メッセージのトークンを置換します。グループの繰り返しなどのために行数が異なるメッセージの場合は、行ごとに項目属性を作成しないで、「文書」タイプの項目属性およびメッセージ属性を使用して行を結合してください。
- 関数がデータベースから必要な値をすべて検索できるように、主キーの値を格納します。
- アクティビティ属性を動的に設定するための一時プレースホルダとして使用します。たとえば、通知の実行者が実行時までわからない場合は、項目属性を参照して、通知を実行する直前に必要な値をシードすることができます。

項目属性は、値を同期する必要がないため、静的値（データベースにない値）を参照します（ただし、主キーの値は変わりません）。表内のすべての列を項目属性として実装しないでください。

次の項目属性タイプを使用すると、必要な属性の数を減らすことができます。

- 文書： 属性値は添付文書です。複雑な構造をインラインで表示したり、通知に添付することができます。次の種類の文書を指定できます。
  - PL/SQL 文書： データベースのデータを文字列として表す文書で、PL/SQL プロシージャから生成されます。

- PL/SQL CLOB 文書： データベースのデータをキャラクタ・ラージ・オブジェクト（CLOB）として表す文書で、PL/SQL プロシージャから生成されます。
- ロール： 属性値はロールの内部名です。通知メッセージにロール・タイプのメッセージ属性が含まれる場合、属性は自動的にロールの表示名に設定されるため、ロールの内部名と表示名について別々の属性を保守する必要がなくなります。また、Web ブラウザから通知を表示する場合は、ロールの表示名がそのロールの電子メール・アドレスを示すハイパーテキスト・リンクになります。

複数の項目属性が作成される場合やワークフローの処理中に複数の項目属性値が設定される場合は、Workflow Engine API の `AddItemAttribute` および `SetItemAttribute` の配列 (`AddItemAttributeArray` と `SetItemAttributeArray`) を使用します。これらの API を使用すると、Workflow Engine API のコール数が大幅に減少し、バッチ処理中のパフォーマンスが大きく向上します。8-46 ページの「`AddItemAttributeArray`」および 8-53 ページの「`SetItemAttributeArray`」を参照してください。

#### 関連項目：

4-2 ページ「項目タイプ属性」

4-3 ページ「属性タイプ」

## メッセージ属性

パフォーマンスを向上させるには、メッセージ属性の数を最小限に抑える必要があります。グループの繰り返しなどのために行数が異なるメッセージの場合は、行ごとに項目属性およびメッセージ属性を作成 (`LINE_INFO1`、`LINE_INFO2` など) しないで、「文書」タイプの項目属性およびメッセージ属性を使用して行を結合してください。

#### 関連項目：

4-3 ページ「属性タイプ」

4-24 ページ「送信および応答メッセージ属性」

## サブプロセス

ワークフロー・プロセスを設計するときに、いくつかのアクティビティを 1 つのプロセス・アクティビティにまとめて、メイン・プロセス内のサブプロセスとして表すことができます。サブプロセスを適切に使用すると、ワークフロー・ダイアグラムがわかりやすくなり、ワークフローの監視と管理が容易になります。ただし、サブプロセスを使用すると、ワークフロー表に追加の DML 操作や追加の状態情報が格納されます。このため、機能面の利点がない場合は不必要にサブプロセスを使用しないでください。

たとえば、次の「プロセス 1」と「プロセス 2」は、両方とも「関数 1」という関数を実行するため、機能的には同じですが、ワークフロー表に格納される状態行の数が異なります。

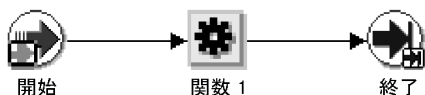
「プロセス 1」には、「開始」アクティビティ、「サブプロセス」アクティビティおよび「終了」アクティビティが含まれます。サブプロセスには、「開始」アクティビティ、「関数 1」

アクティビティおよび「終了」アクティビティが含まれます。このプロセスでは、7つの状態行がワークフロー表に格納されます。

プロセス 1

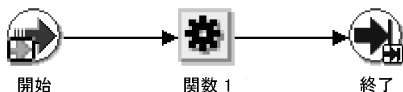


サブプロセス



「プロセス 2」には、「開始」アクティビティ、「関数 1」アクティビティおよび「終了」アクティビティが含まれます。このプロセスでは、4つの状態行がワークフロー表に格納されます。

プロセス 2



「プロセス 1」のような設計では、より多くの行がワークフロー表に格納されるため、「プロセス 2」の設計よりもワークフローのスループットが遅くなり、ワークフローのランタイム表をより頻繁にパージしなければならないようになります。

---

**注意：** このガイドラインは、サブプロセスの使用を禁止するものではありません。すべてのサブプロセスを解除すると、ワークフロー・ダイアグラムがわかりにくくなり、管理しにくくなることがあります。ここでは、サブプロセスを不必要に使用すると、パフォーマンスが低下する可能性があることを説明しています。

---

**関連項目：**

4-57 ページ「プロセス・アクティビティの作成」

## アクティビティの遅延

オンライン・ユーザーの応答時間を最も簡単かつ効果的に向上させるには、関数アクティビティを遅延します。Oracle Workflow では、このようなコストの高いアクティビティをバックグラウンド・タスクとして実行する補助エンジンを設定して、ワークフロー・エンジンの負荷とユーザーの応答時間を管理できます。このような場合、コストの高いアクティビティはワークフロー・エンジンによって遅延され、後でバックグラウンド・エンジンによって実行されます。

アクティビティが遅延すると、メインのワークフロー・エンジンは、次に使用可能なアクティビティに進むことができますが、これによりプロセスの別の並列する分岐が発生する可能性があります。実行可能なアクティビティがなくなると、ワークフロー・エンジンはコール元のアプリケーションにすぐに制御を返します。実行時間が短くなるため、ユーザーは処理が実行中であることを意識する必要がありません。

アクティビティを遅延するには、設計時にアクティビティのコストをデフォルトのしきい値コストより高く設定します。しきい値コストは PL/SQL パッケージ変数の 1 つで、デフォルト値は 50/100 秒です。遅延しないアクティビティのコストにはすべて、このしきい値より高い値を設定します。

ワークフロー・エンジンがしきい値より高いコストのアクティビティを実行時に検出すると、それらのスレッドを遅延してバックグラウンドに回します。バックグラウンド・エンジンは、そのプロセスを遅延として識別し、実行を続行します。

バックグラウンド・エンジンは、遅延アクティビティの他に、タイムアウトになったアクティビティと停止しているプロセスも処理します。バックグラウンド・エンジンは必要な数だけ実行できます。バックグラウンド・エンジンは、タイムアウトになったアクティビティをチェックし、遅延アクティビティを処理し、停止しているプロセスに対応するためにそれぞれ 1 つ以上必要です。少なくとも、タイムアウト / 遅延アクティビティ用に 1 つ、停止しているプロセス用に 1 つは設定する必要があります。

通常、停止しているプロセスをチェックするバックグラウンド・エンジンは、遅延アクティビティを処理するバックグラウンド・エンジンと別個に設定する必要があります。ただし、実行頻度は少なくともかまいません（通常は、1 日に 1 度以下）。システムの負荷が低いときにバックグラウンド・エンジンを実行して、停止しているプロセスをチェックします。

**関連項目：**

8-9 ページ「遅延処理」

2-43 ページ「バックグラウンドのワークフロー・エンジンの設定」

2-47 ページ「エンジンのしきい値の設定」

## パフォーマンス改善のためのランタイム・データの管理

ワークフロー・エンジンが強制同期プロセス以外のワークフローを実行すると、ステータス情報がランタイム表に格納されます。これらの表に格納されるデータの量は、実行されるワークフローの複雑さと数に応じて増加します。

大量のランタイム・データに関連するパフォーマンスの問題は、次の方法で解決できます。

- パーティション化
- パージ

### パフォーマンス改善のためのパーティション化

大きな表や索引を使用するときの主要な問題を解決するには、パーティションと呼ばれる管理しやすい小さな単位に表を分割（パーティション化）します。パーティション表にアクセスするために SQL 問合せと DML 文を変更する必要はありませんが、パーティションを定義すると、DDL 文は表または索引全体ではなく、個々のパーティションにアクセスして操作できるようになります。このように、パーティション化により、大きなデータベース・オブジェクトの管理を簡素化できます。また、パーティション化はアプリケーションに対して完全に透過的です。

スクリプトを実行して、ランタイム・ステータス・データを格納する特定のワークフロー表をパーティション化することもできます。パフォーマンス改善のため、この手順を実行することをお勧めします。スクリプトを実行する前に、パーティション化する表をバックアップし、スクリプトの実行に必要な空き領域と時間を確保していることを確認してください。

Oracle Applications embedded Workflow の場合、このスクリプトは `wfupartb.sql` と呼ばれ、`$FND_TOP` の `admin/sql` サブディレクトリにあります。Oracle Workflow のスタンドアロン版の場合、このスクリプトは `wfupart.sql` と呼ばれ、Oracle ホーム内の `wf/admin/sql` サブディレクトリにあります。2-12 ページの「ワークフロー表のパーティション化」を参照してください。

### パフォーマンス改善のためのパージ

Workflow PURGE API を使用して、完了した項目やプロセスに関する不要なランタイム・データや、使用されなくなった廃止アクティビティのバージョン情報を削除できます。このような廃止データをシステムから定期的に削除することで、パフォーマンスが向上します。Workflow PURGE API は、WF\_PURGE という PL/SQL パッケージに定義されています。

ランタイム・データをパージできるかどうかは、項目タイプの維持タイプによって決まります。維持タイプにより、項目タイプのインスタンスごとに維持される実行ステータス情報の期間を制御します。

- 項目タイプの「維持」を「永続」に設定すると、ランタイム・ステータス情報はプロシージャ `WF_PURGE.TotalPerm()` をコールして情報を意図的に削除するまで、無期限に保存されます。
- 項目タイプの「維持」を「一時」に設定した場合は、維持日数（「n」）も指定する必要があります。「一時」項目タイプの各インスタンスの実行ステータス情報は、その完了

日以後少なくとも *n* 日間は維持されます。*n* 日間維持された後は、WF\_PURGE API のいずれかを使用して項目タイプのランタイム・ステータス情報を削除できます。8-111 ページの「WF\_PURGE」を参照してください。

- 項目タイプの「維持」を「同期」に設定すると、その項目タイプのインスタンスは、項目キー #SYNCH が指定された状態で強制同期プロセスとして実行されます。強制同期プロセスは、単一 SQL セッション内で開始および終了し、データベース表に対して挿入または更新を行いません。強制同期プロセスではランタイム・ステータス情報が保持されないため、通常は、「同期」維持タイプのプロセスを削除する必要はありません。ただし、同期モードでプロセスを実行するときに、テストまたはデバッグを目的として一意の項目キーを使用した場合は、プロセス・インスタンスのランタイム・ステータス情報が保持されます。この情報を削除するには、項目タイプを「維持」から「一時」に変更してから、任意の WF\_PURGE API を実行します。実行したら、項目タイプを「一時」から「同期」に戻します。8-14 ページの「同期プロセス、非同期プロセスおよび強制同期プロセス」を参照してください。

また、特定の項目タイプのランタイム・データを削除する場合は、管理スクリプト `wfrmtype.sql` を使用します。このスクリプトを実行すると、削除する項目タイプを有効な値リストから選択するように求めるプロンプトが表示されます。次に、指定した項目タイプに関連したすべてのランタイム・データを削除するか、または指定した項目タイプの完了したアクティビティと項目のランタイム・データのみ削除するかが確認されます。16-15 ページの「wfrmtype.sql」を参照してください。

#### 関連項目：

8-111 ページ「WF\_PURGE」

4-4 ページ「維持タイプ」

---

# 『Oracle Workflow ガイド』の改訂ノート

ここでは、Oracle Workflow リリース 2.6.2 に関する重要な情報を記載します。『Oracle Workflow ガイド』に関する最新の改訂情報および追加情報を確認してください。

## 通知メーラーの起動

通知メーラーをインストールし、UNIX の Sendmail または Windows NT の MAPI 準拠のメール・アプリケーションで動作するように設定できます。ユーザーは、Windows NT で動作する MAPI 準拠の電子メール・ソフトウェア、または UNIX Sendmail でゲートウェイを提供できる電子メール・ソフトウェアを使用して、電子メール通知を受け取ります。

---

**注意：** 2000 年 6 月 7 日にリリースされた Microsoft Outlook E-mail Security Update は、Oracle Workflow MAPI メーラーで使用される MAPI Common Messaging Calls (CMC) インタフェースをサポートしていません（「OL2000: Developer Information About the Outlook E-mail Security Update」(<http://support.microsoft.com/support/kb/articles/Q262/7/01.ASP>) を参照）。その結果、この Microsoft Outlook E-mail Security Update 以降の更新が適用されている場合、Oracle Workflow MAPI メーラーの動作は、Microsoft Windows プラットフォームでは保証されていません。Oracle Workflow MAPI メーラーの動作は、Windows XP では保証されていません。

Windows NT/2000 で Workflow を使用する場合は、UNIX 版の Oracle Workflow Notification Mailer を UNIX にインストールして、Windows NT/2000 上で実行する Workflow Server のデータベースに接続する方法が保証されています。

---

通知メーラーを起動する前に、設定手順を実行して環境を準備し、メール・アプリケーションに通知メーラー専用のメール・アカウントを 1 つ以上設定する必要があります。

---

**注意：** また、Oracle9i リリース 2 で利用可能なスタンドアロン版の Oracle Workflow を使用している場合は、Oracle Enterprise Manager から利用可能なスタンドアロン版の Oracle Workflow Manager コンポーネントを使用して、通知メーラーのスレーブットを監視できます。詳細は、Oracle Workflow Manager のオンライン・ヘルプを参照してください。

---

### ► UNIX Sendmail バージョンの通知メーラーを使用するための設定

1. UNIX Sendmail のユーザーが外部宛先との間で電子メールを送受信できるように、UNIX Sendmail がオペレーティング・システム上で構成されていることを確認します。
2. Oracle Workflow の UNIX アカウントを作成します。この UNIX ユーザーは、Workflow メールを受信のみを行います。
3. Workflow UNIX ユーザーをオペレーティング・システムの DBA グループに追加し、アカウントの umask を 022 に設定します。Workflow ユーザーの umask を常に 022 にするには、次の行を .profile ファイルまたは .login ファイルに追加します。



umask 022

4. Workflow UNIX ユーザーがサーバーにログインできることと、Sendmail を介して外部宛先との間で電子メールを送受信できることを確認します。
5. 通知メーラーを実行する Workflow UNIX ユーザーの環境が、Oracle Workflow のインストール時に使用されたユーザーの環境と同じように設定されていることを確認します。スタンドアロン版の Oracle Workflow の場合は、Oracle Workflow のインストール時に使用されたアカウントと同じ環境ファイルを使用します。
6. Workflow UNIX ユーザーが Oracle データベース・サーバーにログインできることを確認します。SQL\*Plus を使用して Workflow データベース・ユーザーとしてデータベースにログインすると、このアクセスを確認できます。たとえば、次のようになります。

```
sqlplus <username>/<password>@<connect_string>
```

<username> は Workflow オブジェクトを所有するデータベース・ユーザーの名前に、<password> はそのユーザーのパスワードに、<connect\_string> はデータベースの接続文字列に置き換えます。

7. 通知メーラー・タグ・ファイルのファイル・プロパティをチェックして、Workflow UNIX ユーザーがこのファイルに必要な権限を保持していることを確認します。Oracle Workflow には、タグ・ファイルのサンプルとして wfmail.tag が用意されています。Oracle Workflow のスタンドアロン版を使用している場合、このファイルは Oracle Workflow Server のディレクトリ構造のうちサブディレクトリ \$ORACLE\_HOME/wf/res にあります。
8. Workflow UNIX ユーザーが、通知メーラー・プログラムと構成ファイルに必要な権限を保持していることを確認します。通知メーラー・プログラムのファイル名は wfmail.snd で、サーバー上の \$ORACLE\_HOME/bin サブディレクトリにあります。Oracle Workflow には、構成ファイルのサンプルとして wfmail.cfg が用意されています。Oracle Workflow のスタンドアロン版を使用している場合、このファイルは Oracle Workflow Server のディレクトリ構造のうちサブディレクトリ \$ORACLE\_HOME/wf/res にあります。
9. 通知メーラーの作業ディレクトリを指定します。たとえば、廃棄、処理済および未処理スプール・ファイルに使用するディレクトリとして、DISCARD、PROCESS および UNPROCESS 構成パラメータにこの作業ディレクトリへのパスを指定する必要があります。デフォルトでは、通知メーラーの作業ディレクトリは、通知メーラーを起動するコマンドが実行されるディレクトリです。特定のディレクトリを通知メーラーの作業ディレクトリとして一貫して使用することをお勧めします。Workflow UNIX ユーザーが、このディレクトリに対して読込み権限と書込み権限を持っていることを確認します。たとえば、Workflow UNIX ユーザーのホーム・ディレクトリの下に work というサブディレクトリを作成するには、この場所に対する権限が必要です。

## ► MAPI に準拠した通知メーラーを使用するための設定

1. Workflow NT ユーザーが外部宛先との間で電子メールを送受信できるように、MAPI 準拠のメール・サーバーがインストールおよび構成されていることを確認します。

2. 通知メーラーを実行する Workflow NT ユーザーとして、Windows NT ユーザー・アカウントを指定します。このユーザーは、Workflow サーバーをインストールしたユーザー（oracle ユーザー、applmgr ユーザー）と一致していなくてもかまいません。この Workflow NT ユーザーが通知メーラーを実行するには、MAPI 準拠のメール・サーバーにログインする必要があります。このため、無制限にログインできるユーザー・アカウントを選択する必要があります。

---

**注意：** MAPI 準拠メール・サーバー・アプリケーションと同じマシンで通知メーラーを実行する場合は、メール・サーバーの管理者アカウントを Workflow NT ユーザー・アカウントとして使用しないで、Oracle Workflow の NT ユーザー・アカウントを別個に作成してください。

---

3. Oracle Universal Installer を使用して、Windows NT の PC に MAPI 準拠メール・アプリケーション用の通知メーラーをインストールします。MAPI 準拠の通知メーラーは、クライアント CD からインストールできます。通知メーラーをインストールして実行するマシンは、MAPI 準拠メール・サーバー・アプリケーションと異なるマシンでもかまいません。Workflow NT ユーザーとして Windows NT サーバー・マシンにログインし、Oracle Universal Installer を実行します。
4. MAPI 準拠メール・サーバー・アプリケーションで、Oracle Workflow 専用メール・アカウントを作成します。権限上の理由から、このメール・アカウントは Workflow NT ユーザーに関連付ける必要があります。このアカウントを使用して、Workflow メール以外のメールを受信しないでください。
5. 通知メーラーがインストールされている Windows NT サーバーに Workflow NT ユーザーとしてログインし、このマシン上でローカル・メール・プロファイルを作成して、メール・サーバー上の Oracle Workflow のメール・アカウントに接続します。メール・プロファイルを作成するには、Windows NT のコントロール・パネル（「スタート」>「設定」>「コントロール・パネル」）から「メール」アイコンを選択し、「メール」ダイアログ・ボックスに新しいプロファイルを追加します。
6. この Workflow NT ユーザーが Workflow メール・アカウントを使用して、Windows NT サーバー上の Windows Messaging クライアントにログインできることを確認します。MAPI 準拠メール・サーバーを介して外部宛先との間で電子メールを送受信できることを確認します。
7. Workflow NT ユーザーが Oracle データベース・サーバーにログインできることを確認します。このアクセスを確認するには、SQL\*Plus を使用して Workflow データベース・ユーザーとしてデータベースにログインします。たとえば、次のようにログインします。

```
sqlplus <username>/<password>@<connect_string>
```

<username> は Workflow オブジェクトを所有するデータベース・ユーザーの名前に、<password> はそのユーザーのパスワードに、<connect\_string> はデータベースの接続文字列に置き換えます。

8. 通知メーラー・タグ・ファイルのファイル・プロパティを確認して、Workflow NT ユーザーがこのファイルに対するすべての権限を持っていることを確認します。Oracle Workflow には、タグ・ファイルのサンプルとして wfmail.tag が用意されています。Oracle Workflow のスタンドアロン版を使用している場合、このファイルは Oracle Workflow Server のディレクトリ構造のうちサブディレクトリ res にあります。
9. Workflow NT ユーザーが、通知メーラー・プログラムと構成ファイルに対するすべての権限を持っていることを確認します。通知メーラー・プログラムのファイル名は wfmlr.exe で、<drive>:¥<ORACLE\_HOME>¥bin にあります。Oracle Workflow には、構成ファイルのサンプルとして wfmail.cfg が用意されています。Oracle Workflow のスタンドアロン版を使用している場合、このファイルは Oracle Workflow Server のディレクトリ構造のうちサブディレクトリ res にあります。また、このファイルは PC 上の <drive>:¥<ORACLE\_HOME>¥wf¥data サブディレクトリにもあります。
10. 通知メーラーの作業ディレクトリを指定します。たとえば、廃棄、処理済および未処理ファイルに使用するディレクトリとして、DISCARD、PROCESS および UNPROCESS 構成パラメータにこの作業ディレクトリへのパスを指定する必要があります。デフォルトでは、通知メーラーの作業ディレクトリは、通知メーラーを起動するコマンドが実行されるディレクトリです。特定のディレクトリを通知メーラーの作業ディレクトリとして一貫して使用することをお勧めします。たとえば、<drive>:¥<ORACLE\_HOME>¥wf ディレクトリの下に work というサブディレクトリを作成します。Workflow NT ユーザーが、選択したディレクトリに対するすべての権限を持っていることを確認します。

## ► UNIX Sendmail の通知メーラーの起動

1. 通知メーラーは、サーバー上の \$ORACLE\_HOME/bin サブディレクトリにあります。
  - 構成ファイルを使用して通知メーラーを実行するには、次のコマンド構文を使用します。
 

```
$<ORACLE_HOME>/bin/wfmail.snd -f <config_file>
```

<config\_file> は、通知メーラーの実行時に指定するパラメータを含む構成ファイルのフルパス名に置き換えます。
  - または、次のコマンドを入力して、通知メーラーのパラメータを構成ファイルではなくコマンドラインで引数として指定できます。
 

```
$<ORACLE_HOME>/bin/wfmail.snd <arg1> <arg2> ...
```
  - 構成ファイルを指定している場合でも、次のようにパラメータをコマンドラインの値で指定して、構成ファイル内で特定のパラメータ値を上書きできます。
 

```
$<ORACLE_HOME>/bin/wfmail.snd -f <config_file> <arg1> <arg2> ...
```

<arg1> <arg2> ... は、parameter=value 形式を使用して、必要な数のオプション・パラメータと値に置き換えます。

通知メーラーからの出力情報は、デフォルトではコンソールに送信されます。DEBUG 構成パラメータを Y に設定すると、追加のデバッグ情報を出力できます。また、LOG 構成パラメータにログ・ファイルを指定して、出力を記録することもできます。ログ・ファイルにはデバッグ情報が含まれますが、コンソールにダンプされる本文の内容は含まれません。

ほとんどの出力は、**stdout** を使用してコンソールに送信され、バッファに格納されます。ただし、一部のエラー・メッセージは **stderr** を使用して送信され、バッファには格納されません。つまり、**stderr** によってエラーが報告されてから、**stdout** によって対応する処理の実行結果がログ・ファイルに報告されることがあります。この場合、エラーを調査するには、出力がログ・ファイルに記録されるのを待たなければならないことがあります。

他の UNIX コマンドを追加して、通知メーラー・ジョブの実行方法を制御することもできます。

- バックグラウンドで実行している通知メーラーを UNIX アカウントからログアウトした後も実行し続けるには、通知メーラーの起動コマンドの前に UNIX の **nohup** コマンドを追加し、コマンド文字列の最後に「&」を付けます。たとえば、次のようになります。

```
nohup $<ORACLE_HOME>/bin/wfmail.snd -f <config_file> &
```

**nohup** コマンドを使用すると、ユーザーがログアウトした後もプロセスを引き続き実行できます。**nohup** コマンドからの出力は、現行の作業ディレクトリ内の **nohup.out** というファイルにデフォルトで書き込まれます。**nohup.out** ファイルには、**stdout** からの出力情報と **stderr** からのエラー・メッセージが両方含まれます。**nohup** コマンドの詳細は、UNIX の **man** ページを参照してください。

- 標準出力とエラー・メッセージを個別に記録するには、**stdout** と **stderr** を個別のログ・ファイルにリダイレクトします。たとえば、通知メーラーの **stdout** を **mailer.log** ファイルにリダイレクトし、**stderr** を **err.log** ファイルにリダイレクトするには、次のコマンドを使用します。

```
nohup $<ORACLE_HOME>/bin/wfmail.snd -f <config_file>  
1>mailer.log 2>err.log &
```

この場合、**err.log** ファイルのサイズを確認すれば、エラーが発生しているかどうかを確認できます。このファイルのサイズが 0 バイトの場合、エラーは報告されていません。

---

**注意：** **stderr** のログ・ファイルを指定しない場合は、通知メーラーの **stdout** をログ・ファイルにリダイレクトしないでください。リダイレクトした場合は、**stderr** を使用して報告されたエラー・メッセージが失われます。

---

## ▶ MAPI 準拠メール・アプリケーションの通知メーラーの起動

1. 通知メーラー・プログラムは、<drive>:¥<ORACLE\_HOME>¥bin にあります。MS-DOS プロンプト・ウィンドウに次のコマンドを入力し、通知メーラー・プログラムを起動します。

```
<drive>:¥<ORACLE_HOME>¥bin¥wfmlr.exe -f <config_file>
```

<config\_file> は、通知メーラーの実行時に指定するパラメータを含む構成ファイルのフルパス名に置き換えます。

---

**注意：** このプログラムを起動するには、「Oracle - <SID NAME>」プログラム・グループにある Oracle Workflow Notification Mailer のアイコンをダブルクリックする方法もありますが、最初にこのアイコンのプロパティを編集し、ターゲットとして前述のコマンドを挿入する必要があります。

---

2. または、通知メーラーのパラメータを構成ファイルではなくコマンドラインで引数として指定する場合は、次のコマンドを入力します。

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

構成ファイルを指定している場合でも、パラメータをコマンドラインの値で指定して、構成ファイル内で特定のパラメータ値を上書きできます。

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

<arg1> <arg2> ... は、parameter=value 形式を使用して、必要なオプション・パラメータと値に置き換えます。

## ▶ 通知メーラーの構成ファイルの作成

1. Oracle Workflow には、構成ファイルのサンプルとして wfmail.cfg が用意されています。Oracle Workflow のスタンドアロン版を使用している場合、このファイルは Oracle Workflow Server のディレクトリ構造のうちサブディレクトリ res にあります。また、このファイルは PC 上の <drive>:¥<ORACLE\_HOME>¥wf¥data サブディレクトリにもあります。

2. 構成ファイルの内容は、次の形式になっています。

#Description

PARAMETER1=<value1>

#Description

PARAMETER2=<value2>

...

# で始まる行は解析されないため、コメントとして使用できます。各パラメータ名を等号 (=) の左側に、各パラメータの値を右側に指定します。

3. パラメータは、次のとおりです。

- **CONNECT:** (必須) Oracle Workflow Server がインストールされているデータベース・アカウントに接続するための情報で、  
`username/password@connect_string` (または `alias`) 形式を使用します。
- **ACCOUNT:** (必須) プログラムが通知メッセージの送信に使用するメール・アカウントに接続するための情報です。MAPI 準拠メール・プログラムの場合、アカウント情報はメール・アカウントのプロファイル名とパスワードです。Sendmail の場合、アカウント情報は着信メッセージが格納されるメール・スプール・ファイルのフルパスで、`/var/mail/applmgr` のようになります。メール・スプール・ファイルのファイル名には通常、ユーザー・アカウントが使用されます。このファイルの格納場所は `/var/mail` または `/usr/mail` ディレクトリです。これは、通知メーラー起動元のアカунツ (この例では、`applmgr`) に対応していることに注意してください。

---

**注意：** 通知メーラーの Sendmail バージョンを起動する場合、環境変数 `PATH` に、Sendmail 実行ファイルのディレクトリのフルパスも指定する必要があります。

---

- **NODE:** (必須) ノードの識別子名。ノード名は送信通知 ID に含まれています。デフォルト名は `wf` です。  
  
複数のワークフロー・データベースを 1 つのサーバー上に配置している場合は、通知メーラー構成ファイルごとに一意のノード名を定義して、各インスタンスの通知メーラーで異なる `TMP` ファイルが使用されるようにする必要があります。たとえば、通知メーラーをノード名 `wf` でサーバーに配置している場合、次に設定する通知メーラーのインスタンスにはノード名 `wf2` を割り当てます。

---

**注意：** 通知メーラーはインスタンスごとに1つしか実行できません。このため、各通知メーラーでは、個別の電子メール・アカウントを使用して応答を処理する必要があります。

---

- **FROM:** 通知メッセージがユーザーに配信されるときに、メッセージ・ヘッダーの「送信元」フィールドに表示される値。デフォルトは Oracle Workflow です。

---

**注意：** FROM パラメータは、UNIX Sendmail バージョンの通知メーラーが「送信元」フィールドの値を設定する場合にのみ使用します。MAPI に準拠した通知メーラーでは、ACCOUNT パラメータで指定したメール・アカウントの表示値が使用されます。

---

- **SUMMARYONLY:** (必須) この通知メーラーでは、通知環境設定が SUMMARY に設定されているユーザー / ロールに割り当てられる通知のみを処理するのか、あるいは MAILTEXT、MAILATTH または MAILHTML に指定されたユーザー / ロールの通知のみを処理するのかを指定します。有効な値は Y または N で、デフォルト値は N です。通知環境設定を MAILTEXT、MAILATTH、MAILHTML または SUMMARY に指定したワークフロー・ユーザーまたはロールがある場合は、少なくとも2つの通知メーラーを設定し、一方では SUMMARYONLY=Y、他方では SUMMARYONLY=N を指定する必要があります。

SUMMARYONLY=Y に設定すると、データベースを検索して該当する通知要約を配信した後に、通知メーラー自体がシャットダウンされます。したがって、通知要約を配信する頻度で通知メーラーが実行されるように、スケジュールを設定する必要があります。要約には処理中のすべての通知が含まれるため、通知メーラーによる要約を1日に1度は実行してください。スタンドアロン環境で動作する Oracle Workflow の場合は、UNIX の cron ジョブなど、オペレーティング・システム・スクリプトを作成して、通知メーラーのスケジュールを設定する必要があります。

- **DIRECT\_RESPONSE:** 通知環境設定で MAILTEXT または MAILATTH が指定されているユーザーに、テンプレートによる応答を必要とするプレーン・テキスト通知を送信するには、N または n を指定します。Y または y を指定すると、これらの通知環境設定が指定されているユーザーに、直接応答を要求するプレーン・テキスト通知が送信されます。デフォルトは N です。

テンプレートによる応答方式の場合、ユーザーは応答プロンプトのテンプレートを使用して返信し、各プロンプトの後に応答値を引用符で囲んで入力する必要があります。直接応答方式の場合、ユーザーは応答値を返信の1行目として直接入力する必要があります。

---

---

**注意：** 作成した構成ファイルに `HTML_MAIL_TEMPLATE` パラメータが含まれている場合、通知メーラーは通知環境設定が `MAILATTH` のユーザーにメッセージを送信するときに、`DIRECT_RESPONSE` パラメータを無視します。かわりに、通知環境設定が `MAILATTH` のユーザーは、「Workflow Open Mail for Outlook Express」テンプレートに定義されたプレーン・テキスト・メッセージを受信します。応答は、テンプレートに基づいて行われます。「Workflow Open Mail for Outlook Express」テンプレートを使用する必要がない場合は、`HTML_MAIL_TEMPLATE` パラメータをコメントにして、`DIRECT_RESPONSE` パラメータを有効にする必要があります。

---

---

- **AUTOCLOSE\_FYI:** この通知メーラーで、FYI（参考）通知のように応答不要な通知を、電子メールで送信後に自動的に閉じるかどうかを指定します。有効な値は Y または N で、デフォルトは Y です。

値 N を指定すると、すべての FYI 通知はユーザーが手動で閉じるまで、通知ワークリスト内で開いたままになります。

- **ALLOW\_FORWARDED\_RESPONSE:** 他のロールから転送された電子メール通知に対する応答を許可するかどうかを指定します。有効な値は Y または N で、デフォルトは Y です。

値が N の場合、応答通知の「送信元」欄のメール・アドレスが、記録されている宛先ロール（またはそのロール内のユーザー）のメール・アドレスと一致するかどうかチェックされます。2つのメール・アドレスが一致する場合、その通知は有効なルーティング規則に従って転送された、または転送されなかったことを意味し、通知メーラーでは有効な応答として扱われます。2つのメール・アドレスが一致しない場合、その通知は単に電子メール転送コマンドを使用して転送されたことを意味し、通知メーラーではこの応答が処理されず、応答なしメールとして処理されます。

値が Y の場合、応答通知の「送信元」欄のメール・アドレスはチェックされず、応答は必ず処理されます。

---

---

**警告：** `ALLOW_FORWARDED_RESPONSE` を N に設定する場合は、制限事項があるため注意してください。たとえば、Oracle Workflow ディレクトリ・サービスに `USER/ROLE` の関係を持っていない配布リストのメール・エイリアスに通知が送信されたとします。配布リスト内のユーザーがこの通知に応答した場合、「送信元」欄のメール・アドレスは各ユーザーのメール・アドレスであり、配布リストのメール・エイリアスとは一致しないため、通知メーラーでは常に応答なしメールとして扱われます。

---

---



- **IDLE:** 送信するメッセージをチェックするまでの待機時間（秒）。この値には 0 以上の整数を指定する必要があります。デフォルトは 30 です。
- **LOG:** アクティビティを記録するログ・ファイルの名前。有効な値はファイル名です。このパラメータを指定できるのは、スタンドアロン版の通知メーラーのみです。コンカレント・プログラム版の通知メーラーの場合、アクティビティの出力はコンカレント・マネージャのログ・ファイルに送られます。
- **SHUTDOWN:** 通知メーラーにシャットダウンを指示するファイルの名前またはフルパス。これにより、プロセスを中断せずに通知メーラーを安全にシャットダウンできます。通知メーラーは、処理対象の通知を探す前に、常にシャットダウン・ファイルを探します。ファイルがあれば、通知メーラーはシャットダウンします。通知メーラーを再起動するには、シャットダウン・ファイルを削除する必要があります。デフォルトのファイル名は、`shutdown` です。

シャットダウン・ファイルのフルパスを指定するときは、通知メーラーに設定されている作業ディレクトリにあるファイルを指定することをお勧めします。パスが指定されていない場合、通知メーラーは現行の作業ディレクトリでシャットダウン・ファイルを探します。

Oracle Workflow のスタンドアロン版の場合、通知メーラーの現行の作業ディレクトリは、通知メーラーを起動するディレクトリです。
- **FAILCOMMAND:** 通知メーラーに重大なエラーが発生した場合に実行するコマンド。
- **DEBUG:** デバッグ情報をログに出力するかどうかを指定します。有効な値は Y または N で、デフォルトは N です。
- **TEST\_ADDRESS:** すべての送信電子メール通知をダイレクトするテスト用電子メール・アドレスを指定します。テスト・アドレスによって各宛先の電子メール・アドレスが上書きされるため、各宛先の電子メール・アドレスを変更せずにテスト通知にアクセスし、ワークフロー・プロセスをテストできます。
- **REPLYTO:** （必須）応答を処理する電子メール・アカウントが、発信通知を送信する電子メール・アカウントとは異なる場合に、デフォルトの返信先となる電子メール・アドレス。

---

**注意：** REPLYTO パラメータは、UNIX Sendmail バージョンの通知メーラーが返信先アドレスを設定する場合にのみ使用します。MAPI に準拠した通知メーラーでは、ACCOUNT パラメータで指定したメール・アカウントの表示値が使用されます。

---

- **HTMLAGENT:** HTML 通知応答を処理する HTML Web Agent を識別するベース URL。HTML 添付ファイル付きの電子メール通知をサポートするには、この URL が必須です。デフォルトの URL は、「グローバル・ワークフロー・プリファレン

ス」Web 画面で指定した Workflow Web Agent から取り出されますが、このパラメータに異なる値を入力すると、デフォルトを上書きできます。

- **DISCARD:** 通知メーラーが廃棄メッセージを格納するメール・ファイルのフルパス名。名前の先頭にハイフン (-) が付いている場合、このファイルは通知メーラーによって起動時に削除されます。デフォルトは `-discard` です。

---

**注意：** DISCARD 値には、メール・ファイルのフルパス名を指定する必要があります。

---

通知メーラーの初期設定フェーズが完了したら、作業ディレクトリの領域を節約するために、起動時に廃棄ファイルを削除することをお勧めします。通知メーラーが再起動して廃棄ファイルが削除される前に、廃棄ファイルをバック・アップすることをお勧めします。ファイルの内容を確認してメッセージの廃棄理由を判別することができます。

- **PROCESS:** 通知メーラーが処理済の通知メッセージを格納するメール・ファイルのフルパス名。名前の先頭にハイフン (-) が付いている場合、このファイルは通知メーラーによって起動時に削除されます。デフォルトは `processed` です。

---

**注意：** PROCESS 値には、メール・ファイルのフルパス名を指定する必要があります。

---

通知メーラーの初期設定フェーズが完了したら、作業ディレクトリの領域を節約するために、起動時に処理済ファイルを削除することをお勧めします。ただし、削除する前に、バックアップおよびアーカイブすることをお勧めします。このファイルによって、応答を受信したことを確認できるためです。

- **UNPROCESS:** 通知メーラーが未処理の通知メッセージを格納するメール・ファイルのフルパス名。名前の先頭にハイフン (-) が付いている場合、このファイルは通知メーラーによって起動時に削除されます。ただし、通知メーラーの通常の操作中は、未処理ファイルを削除しないでください。デフォルトは `unprocessed` です。

---

**注意：** UNPROCESS 値には、メール・ファイルのフルパス名を指定する必要があります。

---

- **TAGFILE:** タグ・ファイルのフルパス名。タグ・ファイルには、特殊なメッセージで見つかった文字列と、メッセージの応答にそのような文字列が含まれている場合に割り当てるステータスが表示されます。特殊なメッセージには、休暇デーモンや一括メール・リストで送信されるものなど、返されたり、配信できなかったり、自動返信されたメッセージが含まれます。返されたメッセージ、配信不能メッセー

ジまたは無効なメッセージの識別方法は、メール・システムによって異なります。そのため、タグ・ファイルを使用して、メール・システムによる宛先不明メッセージの識別方法と、そのメール・システムを通して送られてきたこのようなメッセージの通知メーラーによる処理方法を指定できます。通知メーラーによって使用されるオペレーティング・システム・アカウントには、タグ・ファイルへの読み込みアクセス権が必要です。

---

**注意：** タグ・ファイルによってチェックされるのは、通知 ID を持つメッセージ応答のみです。通知 ID を持たないメッセージ応答を受け取ると、通知メーラーはそのメッセージ応答を廃棄ファイルに移動し、応答なしメールを受け取ったという警告メッセージを送信者に送ります。

---

タグ・ファイルで使用される書式は、次のとおりです。

*Status "Matching string"*

*Status* 値は、ERROR、IGNORE または UNAVAIL で、*"Matching string"* は「送信元」行、「件名」行またはメッセージ本文で検索するテキストです。通知メーラーでは、このいずれかのステータス値が割り当てられたメッセージが次のように処理されます。

- IGNORE: メッセージを廃棄ファイルに移動し、オープン通知に対する有効な返信を待ち続けます。通知のステータスは OPEN のままであり、メール・ステータスは SENT のままです。
- ERROR: メッセージを廃棄ファイルに移動し、エラー処理が定義されていればそれを開始します。通知のステータスは OPEN のままですが、メールとアクティビティの各ステータスは ERROR に更新されます。原則的には、ワークフロー管理者が問題点を修正し、メール・ステータスを MAIL に更新して通知を再送信します。
- UNAVAIL (またはユーザーの定義した任意のタグ) : 通知のステータスが OPEN のままであるため、メッセージを廃棄ファイルに移動し、通知への返信を待ち続けます。ただし、メール・ステータスは UNAVAIL に更新されます。このステータスは情報表示のみのものです。この通知による追加処理はありません。

返信される応答値が、特定の選択肢（結果）タイプにある有効な値ではない場合、メッセージ応答に INVALID ステータスを割り当てることもできます。この場合、メッセージを廃棄ファイルに移動し、無効なメッセージを送信しますが、通知のステータスやメール・ステータスに変更されないため、有効な返信を待ち続けます。

---

---

**注意：** タグ・ファイルでは、返されたメッセージおよび自動返信メッセージを、通常の応答と一意に識別することが重要です。返されたメッセージおよび自動返信メッセージを識別しないと、通知メーラーはこれらが無効な応答と誤って判断し、無効メッセージを送って返信を待ち続けます。どちらの場合にも、通知メーラーが無効なメッセージを発信し、その無効なメッセージが返されるか自動返信されるという永久ループが発生してしまいます。

---

---

たとえば、件名またはメッセージ本文に文字列

"-- Unsent message follows --" を含むすべてのメッセージ応答にエラー・マークを設定する場合は、タグ・ファイルに次の行を含めます。

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

---

---

**注意：** メッセージ応答がタグ・ファイル内の複数の文字列と一致する場合は、一致する最初の文字列のステータスが付けられます。つまり、通知メーラーではタグ・ファイルに対して上から順に比較が実行されます。このような動作があるため、**ERROR** タグを先頭にして、次に **UNAVAIL** タグ、次に **IGNORE** タグという優先度順に文字列を並べる必要があります。

---

---

Oracle Workflow には、タグ・ファイルのサンプルとして `wfmail.tag` が用意されています。Oracle Workflow のスタンドアロン版を使用している場合、このファイルは Oracle Workflow Server のディレクトリ構造のうちサブディレクトリ `res` にあります。

- **RESET\_FAILED:** この通知メーラーが起動したときに、**FAILED** ステータスの通知メールをすべて **MAIL** ステータスにリセットするかどうかを指定します。有効な値は **Y** または **N** で、デフォルトは **N** です。

この値が **Y** の場合、通知メーラーが起動したときに、**FAILED** ステータスの通知メールをすべて **MAIL** ステータスにリセットし、通常のメールとして処理しようとします。

- **RESET\_NLS:** 通知メーラーがメッセージを作成する前に、受信者の通知環境設定に従って通知メッセージの **NLS** コードセットを変換するかどうかを指定します。有効な値は **Y** または **N** で、デフォルトは **N** です。

この値が **Y** の場合、通知メーラーは受信者の **Workflow** ユーザー設定項目の言語および地域に基づいて、**WF\_LANGUAGES** 表のコードセットにメッセージを変換します。ユーザー設定項目に地域が指定されていない場合、ユーザー設定項目の言語に指定されている最初のエントリに関連付けられたコードセットを使用します。ユーザー設定項目に言語および地域が指定されていない場合は、**WF\_LANGUAGES** 表内で言語 **AMERICAN** および地域 **AMERICA** に割り当てられたコードセットを使用します。

- **HTML\_MAIL\_TEMPLATE:** 通知環境設定が MAILHTML または MAILATTH のユーザーに対して、応答を必要とする電子メール通知のテンプレートとして「Workflow Open Mail for Outlook Express」メッセージを使用するには、このパラメータに OPEN\_MAIL\_OUTLOOK を指定します。「ワークフロー・オープン・メール (テンプレート)」および「ワークフロー・オープン・メール (ダイレクト)」テンプレートに応答リンクが含まれるときに、応答リンクを処理できない電子メール・アプリケーション (Microsoft Outlook Express などの電子メール・クライアント) を使用している場合は、このテンプレートを使用して対応できます。これらの電子メール・クライアントを使用している場合は、このメッセージ・テンプレートを選択すれば、「通知の詳細」Web 画面にリンクを追加して、ユーザーに対して通知への応答を要求できます。

通常の「ワークフロー・オープン・メール」および「ワークフロー・オープン・メール (ダイレクト)」テンプレートを使用する場合は、構成ファイルでこのパラメータをコメントにすれば、これらのテンプレートがデフォルトで使用されます。

---

**注意:** 作成した構成ファイルに HTML\_MAIL\_TEMPLATE パラメータが含まれている場合、通知メーラーは通知環境設定が MAILATTH のユーザーにメッセージを送信するときに、DIRECT\_RESPONSE パラメータを無視します。かわりに、通知環境設定が MAILATTH のユーザーは、「Workflow Open Mail for Outlook Express」テンプレートに定義されたプレーン・テキスト・メッセージを受信します。応答は、テンプレートに基づいて行われます。「Workflow Open Mail for Outlook Express」テンプレートを使用する必要がない場合は、HTML\_MAIL\_TEMPLATE パラメータをコメントにして、DIRECT\_RESPONSE パラメータを有効にする必要があります。

---

---

**注意:** HTML\_MAIL\_TEMPLATE パラメータは、通知環境設定が MAILTEXT のユーザーには適用されません。

---

- **SEND\_ACCESS\_KEY:** アクセス・キーを含む「通知の詳細リンク」添付ファイルを、HTML 電子メール通知および HTML 添付ファイル付きプレーン・テキスト電子メール通知とともに送信するには、このパラメータに Y を指定します。アクセス・キーを含めると、現在ログインしているかどうかにかかわらず、ユーザーは「通知の詳細リンク」をクリックして、「通知の詳細」Web 画面に直接アクセスできます。ユーザーがログインしていない場合は、アクセス・キーを含む添付ファイルとともに送信された通知でなければ、通知にアクセスすることはできません。「通知の詳細リンク」からアクセス・キーを除外するには、N を指定します。ユーザーがアクセス・キーを含まないリンクをクリックしたときに、まだログインしていない場合は、「通知の詳細」Web 画面にアクセスする前にログインを求めるプロンプトが表示されます。デフォルトは Y です。

- **SENDMAIL\_ARG:** Sendmail バージョンの通知メーラーを使用している場合に、Sendmail の実行時に指定しなければならない引数。Sendmail 引数のデフォルト値は、次のとおりです。

```
%s -t -F "%s" < %s
```

Sendmail コマンドラインの最初の %s は sendmail コマンド、2 番目の %s は FROM パラメータの値、3 番目の %s は一時ファイルの名前に置き換えられます。これらの引数の置換は必須で、この順序で行う必要があります。ただし、これらの引数の置換を正しい順序で行っていれば、コマンドラインの他の引数は SENDMAIL\_ARG を使用してカスタマイズしてもかまいません。

- **INLINE\_ATTACHMENT:** 通知メッセージ（URL 文書、PL/SQL 文書または PL/SQL CLOB 文書を添付した「通知の詳細リンク」、「HTML メッセージ本文」、「通知参照」など）の添付ファイルの Content-Disposition MIME ヘッダーをインラインに設定する場合は、Y を指定します。これらの添付ファイルの Content-Disposition MIME ヘッダーを添付ファイルに設定する場合は、N を指定します。たとえば、電子メール・アプリケーションが「通知の詳細リンク」などの HTML コンテンツをインラインで表示できない場合は、N を指定してこのリンクを添付ファイルとして表示します。デフォルトは N です。

### ➤ 通知メーラーの常駐シェル・スクリプトの実行

1. Oracle Workflow のスタンドアロン版を実行している場合は、障害のためにシャットダウンした場合に、その通知メーラーを再起動する常駐シェル・スクリプトを設定する必要があります。Oracle Workflow には、UNIX Sendmail の通知メーラーを再起動するためのサンプル・シェル・スクリプトが用意されています。このスクリプトは wfmail.csh という名前で、サーバー上の Oracle ホームの bin サブディレクトリにあります。

---

**注意：** Windows NT の通知メーラーを再起動する場合も、同じ技法を使用します。

---

2. オペレーティング・システムのスクリプト・プロンプトから、次のコマンドを入力してシェル・スクリプトを実行します。

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

<config\_file> は、通知メーラーの実行時に指定するパラメータを含む構成ファイルのフルパス名に置き換えます。シェル・スクリプトにより、すべてのコマンドライン引数が通知メーラーの実行ファイルに直接渡されます。

## 応答処理

通知メーラーにより、応答処理に使用する 3 つのファイルが応答メール・アカウント内に作成されます。この 3 つのファイルには、廃棄メッセージ、未処理のメッセージおよび処理済のメッセージが保持されます。

通知メーラーは次の処理を行って、応答メッセージをチェックします。

- 応答用メール・アカウントにログインします。
- メッセージをチェックします。メッセージがあればそれを読み込み、通知 ID とノード識別子をチェックします。
- メッセージが通知でない場合は、それを廃棄ファイルに移動します。
- メッセージが現行のノードに対する通知であれば、それを未処理ファイルに移動します。
- メッセージが異なるノード宛の通知の場合は、後で正しいノードの通知メーラーが読み込めるようにメッセージを移動しません。

次に、通知メーラーは未処理ファイルを開き、各応答を処理します。メッセージごとに次の処理を行います。

- 通知 ID を取り出します。
- 指定されたタグ・ファイルがある場合はそれを参照し、そのメッセージがバウンスされたのかどうかをチェックします。メッセージがバウンスされている場合は、タグ・ファイルの指定に応じて、それを再ルーティングするか、または通知のステータスを更新してそれ以降の処理を中止します。
- Oracle Workflow データベース内で、この通知をチェックします。
  - 通知が存在しない場合は、それを廃棄ファイルに移動します。
  - 通知があっても、完了または取消しされている場合は、廃棄ファイルに移動します。
  - 通知が存在していて処理中の場合は、データベースにあるメッセージの応答属性の定義を使用して応答値を検証します。応答が無効な場合は、宛先のロールに「ワークフロー無効のメール」メッセージを送信します。応答が有効な場合には、応答関数をコールして通知の応答を完了し、変更をデータベースに保存します。
- 完了した通知メッセージを処理済ファイルに移動し、未処理ファイルを閉じます。

構成ファイルで指定された DISCARD パラメータと PROCESS パラメータが「-」で始まっている場合、通知メーラーは破棄フォルダと処理済ファイルの中身を削除し、メール・アカウントとデータベース・アカウントからログアウトします。





---

# 用語集

## アクセス・レベル (Access Level)

0 ～ 1000 の数値。各ワークフロー・ユーザーは固有のアクセス・レベルで操作します。アクセス・レベルにより、そのユーザーが特定のワークフロー・データを変更できるかどうか定義されます。変更できるのは、自分のアクセス・レベル以上のレベルで保護されているデータのみです。

## アクティビティ (Activity)

ビジネス・プロセス中に実行される 1 単位の作業。

## アクティビティ属性 (Activity Attribute)

関数アクティビティの動作を制御するために、その関数アクティビティの外部で定義されているパラメータ。アクティビティ属性を定義するには、「アクティビティ」ウィンドウでアクティビティの「属性」プロパティ画面を表示します。アクティビティ属性に値を割り当てるには、プロセス・ウィンドウで、そのアクティビティ・ノードの「属性値」プロパティ画面を表示します。

## イベント (Event)

システム内の他のオブジェクトまたは外部エージェントに関連付けられた、インターネットまたはイントラネット内の状態変化です。

## イベント・アクティビティ (Event Activity)

ワークフロー・プロセスに組み込めるように、アクティビティとしてモデル化されたビジネス・イベント。

## イベント・キー (Event Key)

イベントのインスタンスを一意に識別する文字列。イベント名、イベント・キーおよびイベント・データは、イベント内で発生したすべてのアクティビティと対話します。

### イベント・サブスクリプション (Event Subscription)

特定のイベントと特定のシステムの関連を指定し、およびイベントのトリガーが発生したときに実行する処理を指定すること。サブスクリプション処理には、カスタム・コードのコール、ワークフロー・プロセスへのイベント・メッセージの送信、またはエージェントへのイベント・メッセージの送信を指定することができます。

### イベント・データ (Event Data)

イベントを説明する一連の追加詳細。イベント・データは XML 文書として作成できます。イベント名、イベント・キーおよびイベント・データは、イベント内で発生したすべてのアクティビティと対話します。

### イベント・メッセージ (Event Message)

データ型 WF\_EVENT\_T によって定義された、ビジネス・イベントを伝達するための標準ワークフロー構造。イベント・メッセージは、イベント・データ以外に、イベント名、イベント・キー、アドレッシング属性およびエラー情報を含む、いくつかのヘッダー・プロパティで構成されます。

### エージェント (Agent)

システム内の通信の名前付きポイント。

### 外部 Java 関数 (External Java Functions)

Oracle データベース・サーバーの外部で、Java 関数アクティビティ・エージェントによって実行される Java プログラム。

### 外部関数 (External Functions)

Oracle データベース・サーバーの外部で実行されるプログラム。

### 関数 (Function)

ビジネス・ルールの定義、アプリケーション内で自動化されているタスクの実行、またはアプリケーション情報の取出しができる PL/SQL ストアド・プロシージャ。ストアド・プロシージャは、標準引数を受け入れて完了結果を戻します。

### 関数アクティビティ (Function Activity)

PL/SQL ストアド・プロシージャで定義され、自動化されている 1 単位の作業。

### 結果コード (Result Code)

結果タイプにより定義される、結果値の内部名。

### 結果タイプ (Result Type)

アクティビティの結果値の候補を含む選択肢タイプの名前。

## 結果値 (Result Value)

完了したアクティビティから戻される値。

## 項目 (Item)

ワークフロー・プロセスにより管理される特定のプロセス、ドキュメントまたはトランザクション。たとえば、「購買承認申請」プロセスのワークフローで管理される項目は、Oracle Internet Commerce の「Web 購買依頼」ページで作成される特定の購買申請です。

## 項目属性 (Item Attribute)

「[項目タイプ属性 \(Item Type Attribute\)](#)」を参照。

## 項目タイプ (Item Type)

同じ項目属性セットを共有する特定カテゴリの全項目のグループ。たとえば、「発注依頼」は、Oracle Internet Commerce の「Web 購買依頼」ページで作成された購買申請をすべてグループ化するための項目タイプです。項目タイプは、プロセスを上位レベルでグループ化する手段としても使用されます。

## 項目タイプ属性 (Item Type Attribute)

特定の項目タイプに関連付けられている機能。「項目属性」と同義。項目タイプ属性は、その項目を保存するアプリケーションで値を検索して設定できる変数として定義されます。項目タイプ属性とその値は、プロセスのすべてのアクティビティに使用可能です。

## コスト (Cost)

アクティビティの完了までに必要な処理量をワークフロー・エンジンに通知するために、関数アクティビティまたは通知アクティビティに割当て可能な相対値。大きいコストを割り当てるほど、アクティビティは複雑になり、完了までの所要時間が長くなります。ワークフロー・エンジンは、コストのしきい値で動作するように設定できます。ワークフロー・エンジンのコストのしきい値を超えるアクティビティは、「DEFERRED」に設定され、処理されません。バックグラウンド・エンジンは、延期されたアクティビティをチェックして処理するように設定できます。

## サブスクリプション (Subscription)

「[イベント・サブスクリプション \(Event Subscription\)](#)」を参照。

## システム (System)

ホスト・マシンやデータベース・インスタンスなどの論理的に孤立したソフトウェア環境。

## 実行者 (Performer)

手動によるアクティビティ（通知）を実行するように割り当てられているユーザーまたはロール。プロセスに含まれている通知アクティビティは、実行者に割り当てる必要があります。

### **選択肢コード (Lookup Code)**

選択肢タイプに定義されている値の内部名。

### **選択肢タイプ (Lookup Type)**

事前定義済の値リスト。選択肢タイプのそれぞれの値には、内部名と表示名が付いています。

### **属性 (Attribute)**

「アクティビティ属性 (Activity Attribute)」、「項目タイプ属性 (Item Type Attribute)」または「メッセージ属性 (Message Attribute)」を参照。

### **タイムアウト (Timeout)**

ワークフロー・エンジンがエラー・プロセスまたは代替アクティビティ（定義されている場合）に進む前に、通知アクティビティを実行する必要がある期間。

### **通知 (Notification)**

ユーザーに配信されるメッセージのインスタンス。

### **「通知」 Web 画面 (Notification Web Page)**

任意の Web ブラウザで表示し、ワークフロー通知の問合せと応答に使用できる Web 画面。

### **通知アクティビティ (Notification Activity)**

ユーザーによる操作を必要とする 1 単位の作業。通知アクティビティは、作業の完了に必要な情報を含むメッセージをユーザーに送信します。

### **通知メーラー (Notification Mailer)**

ユーザーに対してメール・アプリケーションを介して電子メール通知を送信し、電子メールによる応答を処理するコンカレント・プログラム。

### **ディレクトリ・サービス (Directory Services)**

Oracle Workflow のユーザーおよびロールと、サイトのディレクトリ・リポジトリとのマッピング。

### **トランジション (Transition)**

あるアクティビティの完了とプロセス内の別のアクティビティのアクティブ化を定義する関連。プロセス・ダイアグラムでは、2 つのアクティビティを結ぶ矢印がトランジションを表します。

### **ノード (Node)**

「プロセス」ウィンドウに表示されるプロセス・ダイアグラム内のアクティビティのインスタンス。

## バックグラウンド・エンジン (Background Engines)

延期されたアクティビティやタイムアウトになったアクティビティを処理する補助的なワークフロー・エンジン。

## ビジネス・イベント (Business Event)

「[イベント \(Event\)](#)」を参照。

## プロセス (Process)

ビジネス目標を達成するために実行する必要があるアクティビティのセット。

## プロセス・アクティビティ (Process Activity)

他のプロセスで参照できるように、アクティビティとしてモデル化されているプロセス。

## プロセス定義 (Process Definition)

Oracle Workflow Builder に定義されているワークフロー・プロセス。

## 保護レベル (Protection Level)

データ変更が禁止されているユーザーを表す 0 ～ 1000 の数値。ワークフロー・データを定義するときに、誰でも変更できることを示すカスタマイズ可能 (1000) に設定するか、そのデータを定義中のユーザーのアクセス・レベルと同じ保護レベルを割り当てることができます。後者の場合、データを変更できるのは、そのデータの保護レベル以下のアクセス・レベルで操作するユーザーのみです。

## メッセージ (Message)

通知アクティビティにより送信される情報。メッセージは、通知アクティビティに関連付ける前に定義する必要があります。メッセージには、件名、優先度、本文が含まれ、1 つ以上のメッセージ属性も含まれている場合があります。

## メッセージ属性 (Message Attribute)

メッセージが通知で送信されるときに、情報を提供したり応答プロンプトを表示するために、特定のメッセージに対して定義する変数。事前定義済の項目タイプ属性をメッセージ属性として使用できます。「送信」ソースとして定義されたメッセージ属性は、メッセージの送信時にランタイム値で置き換えられます。「応答」ソースとして定義されたメッセージ属性では、メッセージの送信時に応答を求めるプロンプトが表示されます。

## ロール (Role)

共通の職責または職階別にグループ化された 1 人以上のユーザー。

## ワークフロー・エンジン (Workflow Engine)

ワークフロー・プロセス定義を実装する Oracle Workflow コンポーネント。ワークフロー・エンジンにより、項目のすべてのアクティビティのステータスが管理され、関数が自動的に実行されて通知が送信され、完了したアクティビティの履歴が保存され、エラー条件が検出

されて、エラー・プロセスが開始されます。ワークフロー・エンジンはサーバーの PL/SQL に実装され、エンジン API のコール時にアクティブ化されます。

### **ワークフロー定義ローダー (Workflow Definitions Loader)**

フラット・ファイルとデータベース間でワークフロー定義をアップロードおよびダウンロードするためのコンカレント・プログラム。

## 記号

---

#FROM\_ROLE 属性, 4-24  
#HIDE\_REASSIGN 属性, 4-23  
&#NID, 4-30

## A

---

AbortProcess(), 8-39  
AccessCheck(), 8-236  
ACCOUNT パラメータ, 2-56  
ACTID, 7-4, 7-13  
Activities(), 8-117  
AddAttr(), 8-222  
AddCorrelation(), 8-300  
AddItemAttr(), 8-46  
addItemAttrDate(), 8-46  
AddItemAttrDateArray(), 8-49  
addItemAttrNumber(), 8-46  
AddItemAttrNumberArray(), 8-49  
addItemAttrText(), 8-46  
AddItemAttrTextArray(), 8-49  
AddParameterToList, 8-261  
AddParameterToList(), 8-282  
AddParameterToListPos(), 8-283  
Address, 8-260  
AddUsersToAdHocRole(), 8-141  
AdHocDirectory(), 8-121  
Advanced Queuing, 13-2  
Advanced Queuing の統合, 8-167  
ALLOW\_FORWARDED\_RESPONSE パラメータ, 2-58  
「And」アクティビティ, 6-2  
Any イベント, 14-13  
API, 8-3  
AQ メッセージのペイロード, 8-168

AssignActivity(), 8-76  
AUTOCLOSE\_FYI パラメータ, 2-58

## B

---

Background(), 8-44  
BeginActivity(), 8-69  
「Buyer Workbench」  
    Web 画面, 15-66

## C

---

Cancel(), 8-214  
CancelGroup(), 8-215  
CLEAR(), 8-105  
ClearMsgStack(), 8-185  
Close(), 8-221  
compareTo(), 8-102  
CompleteActivity(), 8-71  
CompleteActivityInternalName(), 8-74  
CONNECT パラメータ, 2-56  
Content, 8-260  
CONTEXT(), 8-111  
CreateAdHocRole()", 8-137, 8-139  
CreateAdHocUser(), 8-136  
CreateForkProcess(), 8-41  
CreateMsg(), 8-186  
CreateProcess(), 8-22

## D

---

DBA Studio, 2-90  
DEBUG パラメータ, 2-59  
DEFAULT\_ERROR, 6-25  
DEFAULT\_EVENT\_ERROR, 6-32

Default\_Rule(), 8-287  
DeferredQueue 関数, 8-182  
DequeueEventDetail(), 8-175  
DequeueException(), 8-181  
DequeueOutbound(), 8-172  
DIRECT\_RESPONSE パラメータ, 2-57  
DISCARD パラメータ, 2-60

## E

---

Enqueue(), 8-276  
EnqueueInbound(), 8-170  
Error(), 8-290  
Error\_Rule(), 8-294  
Event(), 8-77  
execute(), 8-91

## F

---

FAILCOMMAND パラメータ, 2-59  
FND\_FNDWFIAS, 11-9  
FND\_FNDWFNOT, 10-14  
FNDWFLST, 8-279  
    コンカレント・プログラム, 16-5  
FNDWFPR, 8-122  
    コンカレント・プログラム, 16-5  
Forward(), 8-199, 8-210  
FORWARD モード, 8-14  
FROM\_ROLE 属性, 4-24  
FROM パラメータ, 2-57  
FUNCMODE, 7-5

## G

---

Generate()  
    WF\_AGENTS\_PKG, 8-318  
    WF\_EVENT\_FUNCTIONS\_PKG, 8-302  
    WF\_EVENT\_GROUPS\_PKG, 8-312  
    WF\_EVENT\_SUBSCRIPTIONS\_PKG, 8-321  
    WF\_EVENTS\_PKG, 8-309  
    WF\_SYSTEMS\_PKG, 8-315  
get\_launch\_attach\_url(), 8-193  
get\_launch\_document\_url(), 8-192  
get\_open\_dm\_select\_window(), 8-194, 8-195  
get\_pref(), 8-152  
GetAccessKey(), 8-154  
getActivityAttr(), 8-87

GetActivityAttrClob(), 8-68  
GetActivityAttrDate(), 8-66  
GetActivityAttrEvent(), 8-66  
GetActivityAttrInfo(), 8-65  
GetActivityAttrNumber(), 8-66  
GetActivityAttrText(), 8-66  
GetActivityLabel(), 8-27  
GetAdvancedEnvelopeURL(), 8-159  
GetAttrDate(), 8-229  
GetAttrDoc(), 8-231  
GetAttrInfo(), 8-225  
GetAttrNumber(), 8-229  
GetAttrText(), 8-229  
GetBody(), 8-233  
getCorrelationID, 8-252  
GetDiagramURL(), 8-155  
GetEnvelopeURL(), 8-157  
GET\_ERROR(), 8-106  
getErrorMessage, 8-254  
getErrorStack, 8-254  
getErrorSubscription, 8-254  
getEventData, 8-253  
getEventKey, 8-253  
getEventName, 8-253  
getFormat(), 8-99  
getFromAgent, 8-253  
GetInfo(), 8-226  
getItemAttr(), 8-89  
GetItemAttrClob(), 8-62  
GetItemAttrDate(), 8-59  
GetItemAttrDocument(), 8-61  
GetItemAttrEvent(), 8-59  
getItemAttributes(), 8-63  
GetItemAttrInfo(), 8-64  
GetItemAttrNumber(), 8-59  
GetItemAttrText(), 8-59  
getItemTypes(), 8-58  
GetItemUserKey(), 8-26  
GetMessageHandle(), 8-180  
getName  
    WF\_AGENT\_T, 8-243  
    WF\_PARAMETER\_T, 8-245  
    WFAttribute, 8-96  
getNotificationAttributes(), 8-239  
getNotifications(), 8-238  
getParameterList, 8-252  
getPriority, 8-251



getProcessStatus(), 8-83  
getReceiveDate, 8-252  
GetRoleDisplayName(), 8-134  
GetRoleInfo(), 8-128  
GetRoleInfo2(), 8-129  
GetRoleName(), 8-133  
GetRoleUsers(), 8-126  
getSendDate, 8-252  
GetShortBody(), 8-234  
GetShortText(), 8-228  
GetSubject(), 8-232  
getSystem, 8-243  
GetText(), 8-227  
getToAgent, 8-254  
getType(), 8-98  
GetUserName(), 8-132  
GetUserRoles(), 8-127  
getValue  
    WF\_PARAMETER\_T, 8-245  
    WFAttribute, 8-97  
GetValueForParameter, 8-261  
GetValueForParameter(), 8-284  
GetValueForParameterPos(), 8-285  
getValueType(), 8-100

## H

---

HandleError(), 8-79  
HIDE\_REASSIGN 属性, 4-23  
HTML\_MAIL\_TEMPLATE パラメータ, 2-63  
HTMLAGENT パラメータ, 2-60

## I

---

IDLE パラメータ, 2-59  
InboundQueue 関数, 8-183  
Initialize, 8-251  
init.ora パラメータ, 13-58  
IsPerformer(), 8-130  
ITEMKEY, 7-4, 7-13  
Items(), 8-116  
ItemStatus(), 8-82  
ITEMTYPE, 7-4, 7-13

## J

---

Java API, 8-6

    関数アクティビティ, 7-8  
Java Runtime Environment, 2-4  
JavaScript  
    Web ブラウザでのサポート, 2-4  
Java インタフェース, 8-6  
Java 関数アクティビティ・エージェント, 2-81  
    停止, 2-90, 16-12  
    起動, 2-81  
Java モニター・ツール, 11-2

## L

---

LaunchProcess(), 8-33  
LDAP, 2-29  
LDAP API, 2-32, 8-148  
Listen(), 8-277  
loadActivityAttributes(), 8-86  
loadItemAttributes(), 8-85  
Log(), 8-289  
Login Server, 2-30  
LOG パラメータ, 2-59

## M

---

MAPI 準拠のメール・アプリケーション, 2-52  
MIME のサポート, 2-46  
mod\_osso, 2-31

## N

---

NewAgent(), 8-274  
NLS コードセット, 2-62  
NLS サポート  
    Oracle Workflow Builder, 2-36  
    Web セッション, 2-37  
    電子メール通知, 2-37  
NODE パラメータ, 2-56  
「NOOP」アクティビティ, 6-6  
Notifications(), 8-118

## O

---

OMBAQ\_TEXT\_MSG, 8-264  
OpenNotificationsExist(), 8-220  
Oracle Advanced Queuing, 13-2  
Oracle Advanced Queuing の統合, 8-167  
Oracle Applications Manager, 1-4

- Oracle DBA Studio, 2-90
- Oracle HTTP Server, 2-31
  - ワークフロー・サーバーの要件, 2-4
- Oracle Internet Directory, 2-29
- Oracle Message Broker, 2-94
- Oracle Net Services, 2-2
- Oracle Workflow Builder
  - 概要, 3-2
  - コマンドラインからの起動, 3-18
  - 保存モード, 3-16, 4-16
  - 要件, 2-2
  - ローダー機能, 3-16
- Oracle Workflow Manager, 1-4
- Oracle Workflow の Web エージェント, 2-16
- Oracle Workflow のバージョン, 2-8
- Oracle Workflow のビュー, 8-161
- Oracle Workflow ホーム・ページ, 9-2
- Oracle9i Application Server
  - ワークフロー・サーバーの要件, 2-4
- Oracle9iAS Single Sign-On, 2-30
- 「Or」アクティビティ, 6-2
- OutboundQueue 関数, 8-184

## P

---

- Parameters(), 8-297
- Ping の確認イベント, 14-10
- PL/SQL
  - 文書, 7-15
- PL/SQL API
  - 関数アクティビティ, 7-3
  - 「PL/SQL CLOB」文書, 7-15
  - 「PL/SQL」文書, 7-15
  - イベント・サブスクリプションのルール関数, 7-24
  - イベント・データ・ジェネレート関数, 7-19
  - キュー・ハンドラ, 7-21
  - セレクト関数またはコールバック関数, 7-12
- PL/SQL CLOB
  - 文書, 7-15, 7-17
- PL/SQL ストアド・プロシージャ
  - 作成, 15-15
  - スクリプト, 15-15
  - ネーミング規則, 15-15
- PL/SQL 文書, 4-5
- ProcessInboundQueue(), 8-179
- PROCESS パラメータ, 2-60
- PURGE

- Workflow PURGE API, 8-114
- PurgeEvent(), 8-177
- PurgeItemType(), 8-178

## R

---

- RAISE(), 8-108
- Raise(), 8-268
- Receive()
  - WF\_AGENTS\_PKG, 8-319
  - WF\_EVENT\_FUNCTIONS\_PKG, 8-304
  - WF\_EVENT\_GROUPS\_PKG, 8-313
  - WF\_EVENT\_SUBSCRIPTIONS\_PKG, 8-322
  - WF\_EVENTS\_PKG, 8-310
  - WF\_SYSTEMS\_PKG, 8-316
- RemoveUsersFromAdHocRole, 8-147
- REPLYTO パラメータ, 2-59
- RESET\_FAILED パラメータ, 2-62
- RESET-NLS パラメータ, 2-62
- Respond(), 8-198, 8-216
- Responder, 8-216
- Responder(), 8-218
- RESPOND モード, 8-14
- RESULT, 7-5, 7-14
- ResumeProcess(), 8-37
- RETRY\_ONLY, 6-29

## S

---

- Schedule\_changes(), 8-151
- Send(), 8-197, 8-204, 8-272
- SEND\_ACCESS\_KEY パラメータ, 2-63
- SendGroup(), 8-197, 8-208
- set\_document\_id\_html(), 8-196
- SetAdHocRoleAttr(), 8-146
- SetAdHocRoleExpiration(), 8-143
- SetAdHocRoleStatus(), 8-136
- SetAdHocUserAttr(), 8-144
- SetAdHocUserExpiration(), 8-142
- SetAdHocUserStatus(), 8-135
- SetAttrDate(), 8-223
- SetAttrNumber(), 8-223
- SetAttrText(), 8-223
- setCorrelationID, 8-256
- SetDispatchMode(), 8-281
- SetErrorInfo(), 8-280
- setErrorMessage, 8-259

setErrorStack, 8-259  
setErrorSubscription, 8-258  
setEventData, 8-257  
setEventKey, 8-257  
setEventName, 8-257  
setFromAgent, 8-258  
SetItemAttrDate(), 8-51  
SetItemAttrDateArray(), 8-56  
SetItemAttrDocument(), 8-54  
SetItemAttrEvent(), 8-51  
SetItemAttrNumber(), 8-51  
SetItemAttrNumberArray(), 8-56  
SetItemAttrText(), 8-51  
SetItemAttrTextArray(), 8-56  
setItemAttrValue(), 8-90  
SetItemOwner(), 8-28  
SetItemParent API, 6-11  
SetItemParent(), 8-81  
SetItemUserKey(), 8-25  
SetMsgAttr(), 8-188  
SetMsgResult(), 8-189  
setName  
    WF\_AGENT\_T, 8-244  
    WF\_PARAMETER\_T, 8-246  
setParameterList, 8-256  
SetParametersIntoParameterList(), 8-295  
setPriority, 8-255  
setReceiveDate, 8-255  
setSendDate, 8-255  
setSystem, 8-244  
setToAgent, 8-258  
setValue, 8-246  
SHUTDOWN パラメータ, 2-59  
StartForkProcess(), 8-43  
StartProcess(), 8-30  
StartProcess 関数  
    サンプル購買申請プロセス, 15-23  
SubscriptionParameters(), 8-299  
Success(), 8-292  
SUMMARYONLY パラメータ, 2-57  
SuspendProcess(), 8-35  
Synch\_all(), 8-150  
Synch\_changes(), 8-149

## T

---

TAGFILE パラメータ, 2-61

TCP/IP ドライバ, 2-2  
Test(), 8-275  
TEST\_ADDRESS パラメータ, 2-59  
TestContext(), 8-235  
TOKEN(), 8-107  
toString(), 8-101  
Total(), 8-119  
TotalPERM(), 8-120  
Transfer(), 8-199, 8-212  
TRANSLATE(), 8-113

## U

---

Unexpected イベント, 14-15  
UNIX Sendmail, 2-52  
UNPROCESS パラメータ, 2-60  
URL  
    Oracle Workflow ホーム・ページ, 9-2  
    イベント・システム・デモンストレーションの Web  
        画面, 15-66, 15-68  
    イベント・データ, 8-53  
    「購買申請デモンストレーション」 Web 画面, 15-10  
    「製品アンケート」 Web 画面, 15-35  
    「通知の検索」 Web 画面, 10-14  
    「プロセスの検索」 Web 画面, 11-8  
    ワークフロー・モニター, 11-8  
    「ワークリスト」 Web 画面, 10-13  
URL 属性  
    フレーム・ターゲット, 4-35  
URL-タイプ属性, 4-3  
「URL」メッセージ属性  
    添付と埋込み, 4-36  
User Entry Has Changed イベント, 14-18  
UserActive(), 8-131

## V

---

value(), 8-95  
VoteCount(), 8-219

## W

---

Warning(), 8-291  
Web 通知  
    要件, 2-4  
Web ホーム・ページ, 9-2  
WF\_ACCESS\_LEVEL, 2-96, 2-99

WF\_AGENT\_T, 8-243  
WF\_AGENTS\_PKG.Generate, 8-318  
WF\_AGENTS\_PKG.Receive, 8-319  
WF\_AGENTS ドキュメント・タイプ定義, 8-317  
WF\_DEFERRED エージェント, 13-27  
WF\_DEFERRED キュー, 2-91  
WF\_ENGINE.BACKROUND, 2-42  
WF\_ERROR\_QH, 13-26  
WF\_ERROR エージェント, 13-27  
WF\_ERROR キュー, 2-91  
WF\_EVENT\_FUNCTIONS\_PKG.Generate(), 8-302  
WF\_EVENT\_FUNCTIONS\_PKG.Receive(), 8-304  
WF\_EVENT\_GROUPS\_PKG.Receive, 8-313  
WF\_EVENT\_GROUPS ドキュメント・タイプ定義,  
8-311  
WF\_EVENT\_OMB\_QH, 13-26  
    設定, 2-94  
    属性のマッピング, 8-264  
WF\_EVENT\_QH, 13-26  
WF\_EVENT\_SUBSCRIPTIONS\_PKG.Generate, 8-321  
WF\_EVENT\_SUBSCRIPTIONS\_PKG.Receive, 8-322  
WF\_EVENT\_SUBSCRIPTIONS ドキュメント・タイプ  
定義, 8-320  
WF\_EVENT\_T, 8-248  
    OMBAQ\_TEXT\_MSG への属性のマッピング, 8-264  
WF\_EVENTS\_PKG.Generate, 8-309  
WF\_EVENTS\_PKG.Receive, 8-310  
WF\_EVENTS ドキュメント・タイプ定義, 8-308  
WF\_IN エージェント, 13-27  
WF\_IN キュー, 2-91  
WF\_ITEM\_ACTIVITY\_STATUSES\_V, 8-161  
WF\_ITEMS\_V, 8-165  
WF\_LANGUAGES ビュー, 2-36  
WF\_LOCAL \* 表, 2-20  
WF\_NOTIFICATION() メッセージ関数, 4-24  
WF\_NOTIFICATION\_ATTR\_RESP\_V, 8-163  
WF\_OUT エージェント, 13-27  
WF\_OUT キュー, 2-91  
WF\_PARAMETER\_LIST\_T, 8-247  
WF\_PARAMETER\_T, 8-245  
wf\_payload\_t, 8-168  
WF\_PURGE, 8-114  
WF\_REQDEMO.SelectApprover, 15-26  
WF\_REQDEMO.StartProcess, 15-8  
WF\_REQDEMO.VerifyAuthority, 15-18, 15-28  
WF\_RESOURCES  
    環境変数, 2-40

WF\_ROLES  
    ビュー, 2-24  
WF\_RUNNABLE\_PROCESSES\_V, 8-164  
WF\_SYSTEMS\_PKG.Generate, 8-315  
WF\_SYSTEMS\_PKG.Receive, 8-316  
WF\_SYSTEMS ドキュメント・タイプ定義, 8-314  
WF\_USER\_ROLES  
    ビュー, 2-25  
WF\_USERS  
    ビュー, 2-21  
Wfagtlst.sql, 16-6  
WFAttribute(), 8-94  
WFAttribute クラス, 8-92  
wfbkgchk.sql, 16-7  
wfbkg.sql, 16-7  
wfhacta.sql, 16-8  
wfhact.sql, 16-8  
wfhcita.sql, 16-8  
wfhcitt.sql, 16-9  
wfhcluc.sql, 16-9  
wfhclut.sql, 16-9  
wfhcmsga.sql, 16-10  
wfhcmsg.sql, 16-10  
wfdirchk.sql, 16-10  
wfdirhrv.sql, 2-26  
wfdirouv.sql, 2-27  
wfevquec.sql, 2-93  
wfevqued.sql, 2-93  
Wfevtreq.sql, 16-11  
WFFunctionAPI クラス, 8-84  
wfjvlsnr.bat, 2-82  
wfjvlsnr.csh, 2-82  
Wfjvstop.sql, 16-12  
WFLOAD, 2-103  
wflload, 2-102  
wfmail.cfg, 2-55  
Wfmqupd.sql, 16-12  
WFNLADD.sql, 16-6  
WFNLENA.sql, 16-12  
wfntfsh.sql, 16-13  
wfprot.sql, 16-13  
wfqclean.sql, 16-13  
Wfquhndob.pls, 2-94  
Wfquhndos.pl, 2-94  
wrfefchk.sql, 16-14  
wrfresgen, 8-108  
wfretry.sql, 16-14

wfrmall.sql, 16-14  
wfrmita.sql, 16-15  
wfrmitms.sql, 16-15  
wfrmitt.sql, 16-15  
wfrmtime.sql, 16-16, C-8  
WFRUND.SQL, 15-8  
wfrun.sql, 16-16  
wfstat.sql, 16-16  
wfstatus.sql, 16-16  
wfstdchk.sql, 16-16  
Wftypes.sql, 8-242  
wfverchk.sql, 16-17  
wfver.sql, 16-17  
wfverupd.sql, 16-17  
wfxload, 2-108, 2-111  
wfxload.bat, 2-108, 2-111  
WorkCount(), 8-237  
Workflow CORE API, 8-104  
Workflow Engine API, 8-3  
Workflow LDAP API, 2-32, 8-148  
「Workflow Open Mail for Outlook Express」メッセージ・テンプレート, 2-69  
Workflow PURGE API, 8-114  
Workflow QUEUE API, 8-167  
「Workflow URL Attachment」メッセージ・テンプレート, 2-73  
Workflow Web Agent の識別, 2-16  
Workflow XML Loader, 2-106  
Workflow\_Protocol(), 8-293  
Workflow Builder のメニュー, A-2  
Workflow ディレクトリ・サービス API, 8-124  
Workflow の Web 画面  
    テンプレートの変更, 2-79  
Workflow のビュー, 8-161  
Workflow 設定項目 API, 8-152  
Workflow 通知 API, 8-202  
WriteMsg(), 8-187  
WriteToClob(), 8-240

## X

---

「XML タグ値の取得」アクティビティ, 6-17  
「XML タグ値の比較」アクティビティ, 6-18  
「XML タグ値の比較 (数値)」アクティビティ, 6-18  
「XML タグ値の比較 (テキスト)」アクティビティ, 6-18  
「XML タグ値の比較 (日付)」アクティビティ, 6-18

「XML 変換」アクティビティ, 6-19

## あ

---

アイコン, 2-80  
アクション  
    サブスクリプション, 13-39  
「アクセス」プロパティ画面, 4-16  
アクセス保護, 2-95  
    カスタマイズの保持, 4-17  
アクセス・レベル, 2-96  
    インジケータ, 4-16  
    デフォルト, 2-99  
アクティビティ, 3-10, 4-40  
    アイコン, 2-80  
    イベント, 4-40, 4-42  
    エラー・プロセス, 6-23  
    外部 Java 関数, 2-81  
    関数, 4-40, 4-42  
    購買申請プロセス, 15-15  
    コスト, 4-44  
    異なるデータ・ストアからのアクセス, 5-7, 6-2  
    コンカレント・マネージャ, 6-20  
    作成, 4-45  
    サプライヤ  
        アドバンスト出荷通知プロセス, 15-97  
        オーダー詳細の取得プロセス, 15-91  
        クレジット・チェック・プロセス, 15-94  
        在庫チェック・プロセス, 15-95  
        最上位のオーダー・プロセス, 15-87  
        サプライヤの請求書の送信プロセス, 15-98  
システム  
    エラー, 4-40  
「詳細 Ping」プロセス, 13-89  
「承認者に通知」サブプロセス, 15-21  
処理コスト, 8-10  
ステータス, 8-3  
タイムアウト, 5-10  
遅延, 4-44  
通知, 4-40, 4-41  
バイヤー  
    アドバンスト出荷通知プロセス, 15-82  
    最上位の発注プロセス, 15-74  
    サプライヤの請求の受信プロセス, 15-84  
    サプライヤの発注承認の受信プロセス, 15-80  
    サプライヤへの発注送信プロセス, 15-78  
プロセス, 4-40, 4-44

- 分岐の結合, 5-4
- 「マスター Ping」プロセス, 13-87
- ワークフロー・イベント・プロトコル・プロセス, 14-24
- アクティビティ属性
  - 値の設定, 5-12
- アクティビティ・ノード
  - 購買申請プロセス, 15-15
  - サプライヤ
    - アドバンスト出荷通知プロセス, 15-97
    - オーダー詳細の取得プロセス, 15-91
    - クレジット・チェック・プロセス, 15-94
    - 在庫チェック・プロセス, 15-95
    - 最上位のオーダー・プロセス, 15-87
    - サプライヤの請求書の送信プロセス, 15-98
  - 「詳細 Ping」プロセス, 13-89
- バイヤー
  - アドバンスト出荷通知プロセス, 15-82
  - 最上位の発注プロセス, 15-74
  - サプライヤの請求の受信プロセス, 15-84
  - サプライヤの発注承認の受信プロセス, 15-80
  - サプライヤへの発注送信プロセス, 15-78
- 「マスター Ping」プロセス, 13-87
- ワークフロー・イベント・プロトコル・プロセス, 14-24
- アクティビティの結合, 5-4
- アクティビティ・ノード
  - 「承認者に通知」サブプロセス, 15-21
- 値リスト
  - Web インタフェース, 10-24, 13-23
- アドバンスト出荷通知イベント, 15-104
- アド・ホックのユーザーおよびロール, 5-25
- API, 8-124
- 「アンケート - シングル・プロセス」
  - アクティビティ, 15-40
  - 概要, 15-39
- 「アンケート - マスター / ディテール・プロセス」
  - アクティビティ, 15-43
  - 概要, 15-42

## い

---

- 維持, 4-4, C-7
- イベント, 13-4
  - エージェントへの送信, 13-42
  - 検索, 13-15
  - 更新, 13-17

- 削除, 13-17
- 事前定義済, 14-2
- 定義, 13-6
- 呼出し, 13-4, 13-71
- ワークフロー・プロセスへの送信, 13-40
- 「イベント」
  - Buyer Workbench
    - Web 画面, 15-66
  - オーダーの追跡
    - Web 画面, 15-68
- 「イベント」 Web 画面, 13-6, 13-8, 13-17, 13-48
- イベント・アクティビティ
  - ワークフロー・エンジン, 8-18
- 「イベント」アクティビティ, 4-42
- イベント・アクティビティの詳細, 5-13
- イベント関数の API, 8-296
- イベント・グループ, 13-4
  - 定義, 13-8
- イベント・グループ更新イベント, 14-4
- イベント・グループ削除イベント, 14-4
- イベント・グループ作成イベント, 14-4
- イベント更新イベント, 14-3
- イベント削除イベント, 14-3
- イベント作成イベント, 14-3
- イベント・サブスクリプション, 13-37
  - ルール関数, 7-24
- 「イベント・サブスクリプション」 Web 画面, 13-54
- イベント・システム・デモンストレーション
  - 開始, 15-65
  - 概要, 15-62
  - 設定, 15-65
  - データ・モデル, 15-64
- イベント・システム・デモンストレーション・プロセス
  - インストール, 15-64
- イベント・システムの同期イベント, 14-8
- 「イベント・システム・ローカル・キュー」 Web 画面, 13-80
- イベント・タイプ属性, 4-3
- イベント・データ, 7-19, 13-5, 13-38
- イベント・データの URL, 8-53
- イベントの API, 8-267
- イベント・ノード, 5-13
- 「イベントの検索」 Web 画面, 13-15, 13-53
- イベントの呼出し, 13-4, 13-71
- 「イベントの呼出し」 Web 画面, 13-71
- 「イベント・プロパティの取得」アクティビティ, 6-14

「イベント・プロパティの設定」 アクティビティ, 6-14  
「イベント・プロパティの比較」 アクティビティ, 6-15  
イベント・マネージャ, 13-3  
イベント・メッセージ  
    エンキュー, 16-11  
イベント・ルール API, 8-286  
イベント・メッセージ  
    データ型, 8-248

## う

---

「ウィンドウ」 メニュー, A-6

## え

---

エージェント, 13-24  
    ping, 13-84  
    キュー, 13-26  
    キュー・ハンドラ, 13-26  
    検索, 13-34  
    更新, 13-36  
    削除, 13-36  
    定義, 13-30  
    データ型, 8-243  
    伝播のスケジュール, 13-66  
    プロトコル, 13-24  
    方向, 13-24  
    リスナーのスケジュール, 13-61  
「エージェント」 Web 画面, 13-30, 13-36  
エージェント更新イベント, 14-6  
エージェント削除イベント, 14-6  
エージェント作成イベント, 14-6  
エージェントの Ping, 13-84  
エージェントの Ping イベント, 14-10  
「エージェントの検索」 Web 画面, 13-34  
エラー・アクティビティ, 6-23  
エラーが発生したアクティビティ  
    再試行, 16-14  
エラー処理  
    イベント・サブスクリプション, 13-46  
    プロセス・アクティビティ, 8-79  
    ワークフロー・プロセス, 8-11  
エラー・チェック・プロセス, 15-53  
    アクティビティ, 15-56  
    概要, 15-55  
エラー・プロセス, 6-23  
エンキュー

    キュー・ハンドラ, 7-22  
    エンジンのしきい値, 2-44

## お

---

応答処理, 8-198  
    通知メーラーによる, 2-64  
応答属性, 2-66, 2-68, 2-71, 2-75  
応答方式  
    直接応答とテンプレートによる応答, 2-58

## か

---

「開始」 アクティビティ, 5-4, 6-8  
「外部 Java」 関数アクティビティ, 8-6, 8-84  
外部 Java 関数アクティビティ, 2-81  
外部文書の統合, 4-5  
カスタマイズの保持  
    アクティビティ, 4-17  
カスタマイズ・レベル, 2-98  
    アクティビティ, 4-7, 4-19, 4-30  
カスタム・ロゴ  
    Web 画面, 2-79  
「割当」 アクティビティ, 6-13  
環境変数  
    WF\_ACCESS\_LEVEL, 2-96, 2-99  
    WF\_RESOURCES, 2-40  
監視  
    作業項目, 1-4  
    ワークフロー・モニター, 11-2  
関数, 3-11  
関数アクティビティ, 4-42  
    標準 Java API, 7-8  
    標準 PL/SQL API, 7-3  
関数アクティビティ属性, 4-8  
関数アクティビティの例  
    「承認権限の検証」, 15-28  
    「承認者の選択」, 15-26  
管理者権限, 2-15

## き

---

キュー  
    エージェントへの割当て, 13-26  
    確認, 13-79  
    設定, 2-91  
    チェック, 13-60

休暇中の転送, 10-26  
キュー・ハンドラ, 7-21  
WF\_EVENT\_OMB\_QH, 2-94  
キュー表, 2-91  
キュー・ハンドラ, 13-26  
強制同期プロセス, 8-16, C-2

## く

---

グローバル変数, 4-2  
グローバル・ワークフロー・プリファレンス  
Web 画面, 2-13

## け

---

「継続フロー」アクティビティ, 6-11

## こ

---

購買申請  
データ・モデル, 15-6  
「購買申請デモンストレーション」  
Web 画面, 15-8  
購買申請承認が必要なことを通知, 15-31  
購買申請プロセス, 15-5  
インストール, 15-6  
開始, 15-8  
概要, 15-14  
関数アクティビティの例, 15-26  
項目属性  
外部文書の統合, 4-5  
項目タイプ, 3-10, 4-2  
維持タイプ, 4-4, C-7  
「イベント・システムのデモ」, 15-71  
「購買申請」, 15-13  
コールバック関数, 4-5  
「コンカレント・マネージャ機能」, 6-20  
コンテキストの再設定, 7-12  
作成, 4-6  
「システム  
エラー」, 6-23  
セクタ関数, 4-5, 7-12  
保存, 3-13  
ロード, 3-13  
「ワークフロー・エージェントの Ping/ 確認」,  
13-85  
「ワークフロー送信プロトコル」, 14-21

「システム  
メーラー」, 2-65  
「標準」, 6-2  
項目タイプ属性, 4-2, 4-7, 4-8, 8-14  
「イベント・システムのデモ」, 15-71  
「購買申請」, 15-13  
配列, 8-14  
パフォーマンス, C-3  
「ワークフロー送信プロトコル」, 14-21  
項目タイプのロード, 3-13  
コールバック関数, 7-12  
command, 7-13  
項目タイプ, 4-5  
コマンド, 7-13  
コストのしきい値, 4-44  
コンカレント・プログラム  
FNDWFLST, 16-5  
FNDWFPR, 16-5  
通知メーラー, 2-45  
ワークフロー・エージェント・リスナー, 8-279  
ワークフロー定義ローダー, 2-103  
ワークフロー・バックグラウンド・プロセス, 2-42  
通知メーラー, 2-53  
「コンカレント・プログラムの実行」アクティビティ,  
6-20  
「コンカレント・プログラムの発行」アクティビティ,  
6-21  
「コンカレント・プログラム待ち」アクティビティ,  
6-22  
「コンカレント・マネージャ機能」項目タイプ, 6-20  
コンカレント・マネージャのアクティビティ, 6-20  
コンカレント・プログラム  
ワークフローの不要ランタイム・データのパージ,  
8-122  
ワークフロー・リソース・ジェネレータ, 8-108

## さ

---

再開封時, 8-12  
再試行エラー, 6-29  
「再割当て」Web 画面, 10-22  
作業項目, 3-10  
削除  
アウトバウンド通知メッセージ・キュー, 16-12  
項目タイプ属性, 16-15  
項目タイプのデータ, 16-15  
項目タイプのランタイム・データ, 16-16, C-8



- すべてのワークフロー・データ, 16-14
- ワークフローのステータス情報, 16-15
- ワークフロー・プロセス
  - 作成および開始, 16-16
- サブスクリプション, 13-37
  - 検索, 13-53
  - 削除, 13-54
  - 事前定義済, 14-2
  - 遅延, 13-43
  - 定義, 13-48
- サブスクリプション更新イベント, 14-7
- サブスクリプション削除イベント, 14-7
- サブスクリプション作成イベント, 14-7
- サブプロセス
  - タイムアウト, 5-10
  - パフォーマンス, C-4
- サブライヤ
  - アドバンスト出荷通知プロセス
    - 概要, 15-96
  - オーダー詳細の取得プロセス
    - 概要, 15-90
  - クレジット・チェック・プロセス
    - 概要, 15-93
  - 在庫チェック・プロセス
    - 概要, 15-95
  - 最上位のオーダー・プロセス
    - 概要, 15-86
  - サブライヤの請求書の送信プロセス
    - 概要, 15-98
- サンプル・プロセス
  - イベント・システム・デモンストレーション,  
15-62
  - 購買申請, 15-5
- サンプル・ワークフロー・プロセス, 15-2

## し

---

- シード・イベント・グループ, 14-8
- ジェネレート関数, 13-5
- システム, 13-18
  - 検索, 13-21
  - 更新, 13-22
  - サインアップ, 13-73, 13-75
  - 削除, 13-22
  - 定義, 13-19
  - 同期, 13-77
  - マスター / コピー, 13-79

- ローカル, 13-18
- 「システム  
エラー」項目タイプ, 6-23
- 「メーラー」項目タイプ, 2-65
- 「システム」Web 画面, 13-19, 13-22
- システム更新イベント, 14-5
- システム削除イベント, 14-5
- システム作成イベント, 14-5
- システム識別子, 13-74
- 「システム識別子」Web 画面, 13-74
- 「システムの検索」Web 画面, 13-21
- 「システムのサインアップ」Web 画面, 13-75
- システムのサインアップ・イベント, 14-12
- システムの統合, 13-2
- 事前定義済イベント, 14-2
- 「実行時間の比較」アクティビティ, 6-3
- 実装手順, 2-6
- 自動応答, 10-26
- 自動通知ハンドラ, 10-26
- 自動ルーティング, 10-26
- 自動レプリケーション
  - イベント・マネージャのオブジェクト, 13-78
- シャットダウン・ファイル, 2-59
- 「終了」アクティビティ, 5-4
- 受信日
  - イベント・メッセージ, 8-277
- 「終了」アクティビティ, 6-8
- 「詳細 Ping」プロセス
  - 概要, 13-88
- 「詳細アンケート・プロセス」
  - アクティビティ, 15-46
- 「詳細アンケート」プロセス
  - 概要, 15-45
- 「承認権限の検証」関数アクティビティ, 15-28
- 「承認者に通知」
  - 通知アクティビティの例, 15-30
- 「承認者に通知」サブプロセス
  - 概要, 15-20
- 「承認者の選択」関数アクティビティ, 15-26
- ショートカット, 5-24
- シングル・サイン・オン, 2-29, 2-30

## す

---

- 数値 - タイプ属性, 4-3
- 「数値の比較」アクティビティ, 6-3
- ステータス・レポート

エンド・ユーザー, 16-16  
    開発者, 16-16  
「スレッドの遅延」アクティビティ, 6-5

## せ

---

請求書イベント, 15-105  
「製品アンケート」  
    Web 画面, 15-35  
「製品アンケート」項目タイプ, 15-38  
製品アンケート・プロセス, 15-33  
    インストール, 15-34  
    開始, 15-35  
セーブポイント, 7-3, 7-8, 8-4  
「セレクトアップのチェック」Web 画面, 13-57, 13-62,  
    13-67  
セレクト関数, 4-4, 7-12  
選択肢コード  
    コピー, 4-21  
選択肢タイプ, 3-10, 4-18  
    コピー, 4-21  
    作成, 4-18  
選択肢 - タイプ属性, 4-3

## そ

---

送信日  
    イベント・メッセージ, 8-273  
ソース・タイプ, 13-37  
属性  
    タイプ, 4-3, 4-33  
    トークンの置換, 4-39  
    「標準」, 4-40, 6-2  
属性タイプ  
    URL, 4-33  
    イベント, 4-35  
    数値, 4-33  
    選択肢, 4-33  
    テキスト, 4-33  
    日付, 4-33  
    フォーム, 4-34  
    文書, 4-35  
    ロール, 4-35  
属性 - タイプ属性, 4-3  
その他の項目タイプ, 3-5  
ソフトウェア要件, 2-2

## た

---

ダイアグラムの矢印, 5-2  
「待機」アクティビティ, 6-4  
タイムアウト, 5-10  
    動的, 5-11  
タイムアウト・トランジション, 5-2, 5-3  
タイムアウトになったプロセス, 2-40, C-6  
タグ・ファイル, 2-61  
多言語サポート, 16-6, 16-12  
単一コンシューマ・キュー, 13-42

## ち

---

チェック  
    アクティビティのバージョン, 16-17  
    外部キー / 主キーの参照, 16-14  
    ディレクトリ・サービスのデータ・モデル, 16-10  
    バックグラウンド・エンジン, 16-7  
    ワークフローのデータ・モデル, 16-16  
遅延アクティビティ, 2-40, 4-44  
    パフォーマンス, C-6  
遅延処理  
    イベント・サブスクリプション, 13-43  
    ワークフロー・プロセス, 2-40, 8-9, C-6  
直接応答の電子メール, 10-3

## つ

---

通知, 10-2  
    「From Role」の設定, 4-24  
    HTML 形式の電子メール, 10-9  
    応答者の識別, 8-216  
    「再割当て」ボタンの非表示, 4-23  
    ステータス, 16-13  
    タイムアウト, 8-200  
    直接応答を使用したプレーン・テキストの電子メール, 10-6  
    「通知」Web 画面経由, 10-13  
    「通知」Web 画面での再割当て, 10-22  
    「通知」Web 画面を使用した応答, 10-22  
    ディレクトリ・サービスへの依存, 10-2  
    電子メール経由, 10-2, 2-45  
    電子メールによる再割当て, 10-13  
    転送, 8-199  
    添付ファイル付きのプレーン・テキスト電子メール, 10-12

- テンプレートによる応答を使用したブレーン・テキストの電子メール, 10-4
- ロード・バランス, 6-8
- 通知 AP, 8-197
- 通知 API, 8-202
- 通知 ID, 10-3
- 通知 ID トークン, 4-30
- 「通知」Web 画面, 1-4
  - 通知の再割当て, 10-22
- 「通知」アクティビティ, 6-8
- 通知アクティビティ, 4-41
  - カスタム関数との結合, 8-14
  - 「購買申請承認が必要なことを通知」, 15-31
  - 作成, 4-45
- 通知環境設定, 2-19, 2-46, 9-8
- 通知関数, 8-14
- 通知後関数, 4-41, 8-14
- 通知システム, 2-45, 8-197
- 通知
  - 譲渡, 8-199
- 通知テンプレート
  - 電子メール通知用, 2-65
- 通知のアクセス・キー, 10-3
- 「通知の検索」Web 画面, 10-15
- 通知の再割当て
  - 「再割当て」ボタンの非表示, 4-23
  - 「通知」Web 画面, 10-22
  - 電子メール経由, 10-13
- 「通知の詳細」Web 画面, 10-19
- 通知の表示
  - Web ブラウザ, 10-13
  - 「通知」Web 画面, 10-13
  - 電子メール, 10-2
  - 電子メールによる要約, 10-25
- 通知の要約
  - 電子メール経由, 10-25
- 通知への応答
  - HTML 形式の電子メール, 10-9
  - 直接応答を使用したブレーン・テキストの電子メール, 10-6
  - 「通知」Web 画面経由, 10-13
  - 添付ファイル付きのブレーン・テキスト電子メール, 10-12
  - テンプレートによる応答を使用したブレーン・テキストの電子メール, 10-4
- 通知方法, 10-2
- 通知メーラー

- MAPI 準拠アプリケーションとして起動, 2-54
- MIME のサポート, 2-46
- UNIX Sendmail の通知メーラーを起動, 2-53
- 応答処理, 2-64
- 概要, 2-45
- 起動, 2-53
- 構成ファイル, 2-55
- 再起動するためのスクリプト, 2-64
- シャットダウン, 2-45
- 通知環境設定, 2-46
- 必須のフォルダ, 2-60
- 通知履歴, 4-24
- 通知ワークリスト, 10-18
- ツールバー
  - Oracle Workflow Builder, A-8

## て

---

- 「定期警告」
  - 項目タイプ, 15-54
- 停止しているプロセス, 2-41
- 定数
  - WFAttribute クラス, 8-92
- ディスパッチ・モード, 13-45
- ディテール・プロセス, 6-11
- ディレクトリ・サービス, 2-20
  - Builder からの表示, 5-27
  - Oracle HR との統合, 2-26
  - データ・モデルのチェック, 2-26, 16-10
  - 同期, 2-29
  - ナビゲータ・ツリー, 3-5
  - ネイティブ Oracle ユーザーとの統合, 2-26
  - ローカル・ワークフロー・ユーザーとの統合, 2-28
- ディレクトリ・リポジトリ, 2-20
- ディレクトリ・サービス
  - API, 8-124
  - 同期, 8-148
- データ型
  - WF\_AGENT\_T, 8-243
  - WF\_EVENT\_T, 8-248
  - WF\_PARAMETER\_LIST\_T, 8-247
  - WF\_PARAMETER\_T, 8-245
  - ビジネス・イベント・システム, 8-242
  - 例, 8-262
- データベース・リンク
  - 作成, 2-90
  - チェック, 13-60

データ型  
    wf\_payload\_t, 8-168  
テキスト - タイプ属性, 4-3  
「テキストの比較」 アクティビティ, 6-3  
デキュー  
    キュー・ハンドラ, 7-22  
テスト・ハーネス, 12-2  
デフォルト・イベント・エラー・プロセス, 6-32  
デフォルト・エラー・プロセス, 6-25  
デフォルト・トランジション, 5-2  
デモンストレーション  
    ディレクトリ・サービス, 15-7  
電子メール通知, 1-4, 2-45  
    HTML 添付ファイル, 2-4, 10-2  
    直接応答インスタクションの例, 10-7  
    テンプレート, 2-45, 10-3  
    メール・テンプレートの変更, 2-65  
    要件, 2-4  
    要約, 10-25  
伝播  
    アウトバウンド・エージェント, 13-66  
    更新, 13-70  
    削除, 13-70  
    スケジュール, 13-67  
    設定, 13-56  
テンプレートによる応答の電子メール, 10-3

## と

---

同期  
    Oracle Internet Directory との, 2-29, 2-32, 8-148  
同期プロセス, 8-16, C-2  
動的タイムアウト, 5-11  
動的優先度, 5-11  
投票アクティビティ  
    処理, 8-200  
トークンの置換  
    属性, 4-39  
ドキュメント・タイプ定義  
    WF\_AGENTS, 8-317  
    WF\_EVENT\_GROUPS, 8-311  
    WF\_EVENT\_SUBSCRIPTIONS, 8-320  
    WF\_EVENTS, 8-308  
    WF\_SYSTEMS, 8-314  
    ビジネス・イベント・システム, 8-306  
ドキュメント・レビュー・プロセス, 15-48  
    アクティビティ, 15-51

概要, 15-50  
トランジション, 5-2  
    作成, 5-18  
    タイムアウト, 5-2  
デフォルト, 5-2  
編集, 5-18

## な

---

内部名  
    アクティビティ属性の更新, 16-8  
    アクティビティの更新, 16-8  
    項目属性の更新, 16-8  
    項目タイプの更新, 16-9  
    選択肢コードの更新, 16-9  
    選択肢タイプの更新, 16-9  
    メッセージ属性の更新, 16-10  
    メッセージの更新, 16-10  
「内容の添付」 チェックボックス, 4-36  
ナビゲータ・ツリー  
    オブジェクトの検索, 3-7  
ナビゲータのツールバー, A-8

## に

---

任意トランジション, 5-2

## ね

---

ネーミング規則  
    PL/SQL ストアド・プロシージャ, 15-15

## の

---

ノード  
    開始と終了, 5-8  
    プロセスへの追加, 5-6  
ノードのアクティビティ  
    動的優先度, 5-11

## は

---

ページ  
    パフォーマンス, C-7  
    ランタイム・データ, 16-5  
バージョン, 8-13, 16-17  
バージョン管理, 3-8

バージョンの互換性, 2-8  
ハードウェア要件, 2-2  
「はい / いいえ投票」アクティビティ, 6-10  
バイヤー

    アドバンスト出荷通知プロセス

        概要, 15-82

    最上位の発注プロセス

        概要, 15-73

    サプライヤの請求の受信プロセス

        概要, 15-84

    サプライヤの発注承認の受信プロセス

        概要, 15-79

    サプライヤへの発注送信プロセス

        概要, 15-77

バックグラウンド・エンジン

    概要, 2-40

    起動, 2-42

    スクリプト, 16-7

    発行, 2-42

発注イベント, 15-100

発注承認イベント, 15-102

パフォーマンス

    概念, C-2

    項目属性, C-3

    サブプロセス, C-4

    遅延アクティビティ, C-6

    同期ワークフローおよび非同期ワークフロー, C-2

    ページ, C-7

    メッセージ属性, C-4

    ワークフロー表のパーティション化, C-7

パラメータ

    データ型, 8-245

パラメータ・リスト

    データ型, 8-247

## ひ

---

「比較」アクティビティ, 6-3

ビジネス・イベント, 13-4

    ワークフロー・プロセス, 8-18

ビジネス・イベント・システム, 1-3

    Ping/ 確認の例, 13-84

        概要, 8-241

    事前定義済イベント, 14-2

        設定, 2-90

    セットアップのチェック, 13-56

    ビジネス・イベントの管理, 13-2

ビジネス・イベント・システムのレプリケーションの  
    API, 8-306

日付 - タイプ属性, 4-3

「日付の比較」アクティビティ, 6-3

非同期プロセス, 8-16, C-2

ビュー

    Oracle Workflow, 8-161

「表示」メニュー, A-4

標準 API

    PL/SQL CLOB 文書, 7-15, 7-17

    PL/SQL 文書, 7-15

    イベント・サブスクリプションのルール関, 7-24

    イベント・データ・ジェネレート関数, 7-19

    関数アクティビティ, 7-3, 7-8

    キュー・ハンドラ, 7-21

    セレクト関数 / コールバック関数, 7-12

標準アクティビティ, 6-2

標準エラー・プロセス, 6-23

標準項目タイプ, 6-2

## ふ

---

「ファイル」メニュー, A-2

フェーズ番号, 13-38, 13-45

フォーム - タイプ属性, 4-3

フォント

    設定, 5-22

    変更, 5-22

複数コンシューマ・キュー, 13-42

フレーム・ターゲット

    URL 属性, 4-35

「フロー待ち」アクティビティ, 6-11

プロセス

    アクティビティ・トランジション, 5-2

    印刷, 5-21

    開始, 5-4

    概要, 5-20

    クリップボードへのコピー, 5-21

    検証, 5-21

    作成, 3-8

    定義

        変更, 3-12

    編集, 3-12

    ループ, 7-6, 8-11

「プロセス」ウィンドウ, 5-2

「プロセス」ウィンドウのツールバー, A-9

プロセスウィンドウ

- 編集, 5-2
- プロセス・ダイアグラム
  - 作成, 5-2, 5-6
  - ノードの追加, 5-6
- 「プロセスの開始」アクティビティ, 6-5
- プロセスのリセット, 8-79
- プロセスのロールバック, 8-79
- プロセス・アクティビティ, 4-44
- 「ブロック」アクティビティ, 6-5
- プロトコル, 13-24
- 文書, 4-5
- 「文書管理」
  - 項目タイプ, 15-49
- 文書管理 API, 8-191
- 文書管理の統合, 4-3, 4-5
- 文書 - タイプ属性, 4-3
- 文書の統合, 4-3, 4-35, 7-15
- 「文書」メッセージ属性
  - 添付と埋込み, 4-36

## へ

---

- ペイロード
  - Oracle Advanced Queuing メッセージ用, 8-168
- 「ヘルプ」メニュー, A-7
- 変換, 2-36
- 「編集」メニュー, A-3

## ほ

---

- ホーム・ページ, 9-2
- 保護レベル, 2-97
  - 再設定, 16-13
- 保護レベル・ロック, 2-95

## ま

---

- 「マスター Ping」プロセス
  - 概要, 13-86
- マスター / コピー・システム, 13-79
- 「マスター / ディテール連携」アクティビティ, 6-11
  - 使用方法に関する注意, 6-13
- マスター / ディテールを連携するアクティビティ,
  - 6-11
- マスター・プロセス, 6-11

## み

---

- 未来日付のイベント, 13-44

## め

---

- メッセージ, 3-10
  - 件名, 4-28, 15-31
  - コピー, 4-39
  - 作成, 4-27
  - デフォルト優先度の上書き, 5-11
  - 表示, 15-32
  - 本文, 4-29, 15-31
- 「メッセージ」ウィンドウ, 4-22
- メッセージ関数
  - WF\_NOTIFICATION(), 4-24
- メッセージ属性, 4-22, 4-23, 4-32, 15-31
  - #FROM\_ROLE, 4-24
  - #HIDE\_REASSIGN, 4-23
- 「Workflow Open Mail for Outlook Express」メッセージ, 2-71
- 「Workflow URL Attachment」メッセージ, 2-73
- 「応答」, 4-23, 4-33, 4-37
- 書式化された表, 4-24
- 「送信」, 4-23, 4-33
- ソース, 4-23, 4-33
- パフォーマンス, C-4
- 「ワークフロー・オープン・メール」メッセージ,
  - 2-66
- 「ワークフロー・オープン参考メール」メッセージ,
  - 2-72
- 「ワークフロー・クローズ・メール」メッセージ,
  - 2-76
- 「ワークフロー警告メール」メッセージ, 2-78
- 「ワークフロー取消のメール」メッセージ, 2-74
- 「ワークフロー無効メール」メッセージ, 2-75
- 「ワークフロー要約メール」メッセージ, 2-77

- メッセージ伝播
  - 設定, 13-56
- メッセージ・テンプレート
  - 電子メール通知用, 2-65
- メニュー
  - Oracle Workflow Builder, A-2

## も

---

- 「モニター URL」アクティビティ, 6-13

## や

---

矢印, 5-2

## ゆ

---

有効

日付, 3-8

有効日付, 3-15, 3-17, 8-13

ユーザー

アド・ホック, 5-25

ユーザー設定項目

Web 画面, 9-6

言語と地域, 2-19, 9-8

通知環境設定, 2-19, 9-8

文書管理ホーム, 2-19, 9-8

ユーザー定義データ型

ビジネス・イベント・システム, 8-242

ユーザー定義の警告処理プロセス

アクティビティ, 15-61

概要, 15-60

ユーザー設定項目, 2-12

優先する通知方法, 10-2

## よ

---

要件

ハードウェアとソフトウェア, 2-2

## ら

---

ランタイム・データ, C-7

## り

---

リスナー

インバウンド・エージェント, 13-61

更新, 13-66

削除, 13-66

実行, 16-6

スケジュール, 13-62

## る

---

ルーティング

自動, 10-26

ルーティング・ルール

上書き, 10-31

更新, 10-32

削除, 10-32

リストの表示, 10-26

ロール, 10-27

ループ, 7-6, 8-11

「ループ・カウンタ」アクティビティ, 6-6

ループ・リセット, 5-3

ルール関数, 7-24

イベント・サブスクリプション, 13-39

## れ

---

レプリケーション API

ビジネス・イベント・システム, 8-306

## ろ

---

ローカル・システム, 13-18

ローダー・プログラム, 2-101

ロード・バランス, 6-8

ロール, 5-25

Builder からの表示, 5-27

Workflow Builder へのロード, 5-25

アド・ホック, 5-25

管理者, 2-15

「ロール」

タブ, 5-25

プロパティ画面, 5-27

「ロール解決」アクティビティ, 6-8

ロール - タイプ属性, 4-3

ロールバック

プロセス, 8-79

## わ

---

ワークフロー・イベント・プロトコル・プロセス

概要, 14-22

ワークフロー・エージェントの Ping/ 確認, 13-84

「ワークフロー・エージェントの Ping/ 確認」

項目タイプ, 13-85

項目タイプ属性, 13-85

ワークフロー・エージェント・リスナー, 16-5

「ワークフロー・エージェント・リスナー」コンカレント・プログラム, 8-279

ワークフロー・エンジン, 1-3

CANCEL モード, 8-12

CORE API, 8-104  
 Java API, 8-6, 8-20  
 PL/SQL API, 8-20  
 RUN モード, 8-12  
   アクティビティ開始のコール, 8-3  
   アクティビティ完了後のコール, 8-9  
   エラー処理, 8-11  
   コストのしきい値, 4-44  
   しきい値コスト, 2-44, 8-10  
   遅延アクティビティ, 8-9  
   マスター/ディテール・プロセス, 8-81  
   ループ, 8-11  
 ワークフロー・オープン・メール (ダイレクト),  
   2-67, 2-69  
 ワークフロー・オープン・メール」メッセージ・テン  
   プレート, 2-66  
 「ワークフロー・オープン・メール」メッセージ・テン  
   プレート, 2-72  
 ワークフロー管理者, 2-15  
 ワークフロー・キュー  
   削除, 16-13  
 「ワークフロー・クローズ・メール」メッセージ・テン  
   プレート, 2-76  
 「ワークフロー警告メール」メッセージ・テンプレー  
   ト, 2-78  
 ワークフロー・サーバー  
   要件, 2-3  
 ワークフロー送信プロトコル  
   サンプル・ワークフロー・プロセス, 14-20  
 「ワークフロー送信プロトコル」  
   項目タイプ, 14-21  
 ワークフロー送信プロトコル・イベント, 14-27  
 ワークフロー送信プロトコル受信確認イベント, 14-28  
 ワークフロー定義  
   ソース・コントロール, 3-13  
   テスト, 12-2  
   転送, 2-101  
   ロード, 1-3  
 ワークフロー定義のアップグレード, 8-13  
 ワークフロー定義ローダー, 1-3, 2-101, 2-102  
   コンカレント・プログラム, 2-103  
 ワークフロー・デザイナー、「Oracle Workflow Builder」  
   を参照, 1-3  
 「ワークフロー・デモンストレーション」ホーム・ペー  
   ジ, 15-2  
 「ワークフロー取消のメール」メッセージ・テンプレー  
   ト, 2-74  
 「ワークフローの不要ランタイム・データのパージ」コ  
   ンカレント・プログラム, 8-122  
 ワークフロー表のパーティション化, 2-11, C-7  
 ワークフロー・プロセス  
   作成および開始, 16-16  
   監視, 11-2  
   サンプル, 15-2  
 ワークフロー・プロセスの開始, 15-8, 15-35, 15-65  
 「ワークフロー無効メール」メッセージ・テンプレー  
   ト, 2-75  
 ワークフロー・モニター, 11-2  
   管理ボタン, 11-7  
   詳細タブ・ウィンドウ, 11-5  
   プロセス・ダイアグラム・ウィンドウ, 11-4  
   プロセス・タイトル, 11-3  
   設定, 2-79  
 ワークフロー・ユーザー, 2-20  
 「ワークフロー要約メール」メッセージ・テンプレー  
   ト, 2-77  
 ワークフロー・リソース・ジェネレータ  
   コンカレント・プログラム, 8-109  
 ワークフロー・ロール, 2-20  
 ワークフロー・エンジン  
   CORE API, 8-114  
   ディレクトリ・サービス, 8-124  
 ワークフロー・モニター API, 8-153  
 ワークフロー・リソース・ジェネレータ, 8-108  
 「ワークリスト」Web 画面, 10-18