Oracle® Workflow

API リファレンス

リリース 2.6.3

部品番号 : B12366-01

2004 年 3 月

ORACLE®

Oracle Workflow API リファレンス , リリース 2.6.3

部品番号 : B12366-01

原本名 : Oracle Workflow API Reference, Release 2.6.3

原本部品番号 : B10286-02

原本著者 : Siu Chang, Clara Jaeckel

原本協力者 : George Buzsaki, John Cordes, Mark Craig, Mark Fisher, Kevin Hudson, George Kellner, Angela Kung, David Lam, Jin Liu, Kenneth Ma, Steve Mayze, Santhana Natarajan, Tim Roveda, Robin Seiden, Sachin Sharma, Sheryl Sheh, Susan Stratton

制限付権利の説明

このプログラム（ソフトウェアおよびドキュメントを含む）には、オラクル社およびその関連会社に所有権のある情報が含まれています。このプログラムの使用または開示は、オラクル社およびその関連会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と工業所有権に関する法律により保護されています。

独立して作成された他のソフトウェアとの互換性を得るために必要な場合、もしくは法律によって規定される場合を除き、このプログラムのリバース・エンジニアリング、逆アセンブル、逆コンパイル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場合があります。オラクル社およびその関連会社は、このドキュメントに誤りが無いことの保証は致し兼ねます。これらのプログラムのライセンス契約で許諾されている場合を除き、プログラムを形式、手段（電子的または機械的）、目的に関係なく、複製または転用することはできません。

このプログラムが米国政府機関、もしくは米国政府機関に代わってこのプログラムをライセンスまたは使用する者に提供される場合は、次の注意が適用されます。

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation, and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このプログラムは、核、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への用途を目的としておりません。このプログラムをかかる目的で使用する際、上述のアプリケーションを安全に使用するために、適切な安全装置、バックアップ、冗長性（redundancy）、その他の対策を講じることは使用者の責任となります。万一かかるプログラムの使用に起因して損害が発生いたしましても、オラクル社およびその関連会社は一切責任を負いかねます。

Oracle は Oracle Corporation およびその関連会社の登録商標です。その他の名称は、Oracle Corporation または各社が所有する商標または登録商標です。

目次

3 ディレクトリ・サービス API

4 通知システム API

6 Workflow QUEUE API

7 文書管理 API

用語集

索引

はじめに

対象読者

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。

- ビジネスエリアの基礎と基本操作
- Oracle Workflow

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・システムの概念を基本的に理解し、Oracle Database、PL/SQL および Oracle Application Server のテクノロジに精通している読者を想定しています。

このマニュアルの構成

このマニュアルには、Oracle Workflow の API 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、Workflow Engine API について説明します。
- 第 3 章では、ディレクトリ・サービス API について説明します。
- 第 4 章では、通知システム API について説明します。
- 第 5 章では、ビジネス・イベント・システム API について説明します。
- 第 6 章では、Workflow QUEUE API について説明します。
- 第 7 章では、文書管理 API について説明します。

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・マニュアルやサポート・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 11*i* 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 に組み込まれている 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 11*i* 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ってい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。
- **11*i* 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。
- **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。

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 に組み込まれている 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 ユーザーズ・ガイド』

このマニュアルには、このリリースの Oracle Workflow で利用可能な Graphical User Interface（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、ユーザー・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 Workflow 管理者ガイド』
このマニュアルには、ワークフロー対応プロセスも含め、Oracle Applications 製品に必要な設定手順を完了する方法と、実行時のワークフロー・プロセスの進行状況を監視する方法について記載されています。

『Oracle Workflow 開発者ガイド』
このマニュアルには、新しいワークフロー・ビジネス・プロセスを定義する方法と、既存の Oracle Applications に組み込まれている Workflow プロセスをカスタマイズする方法が記載されています。また、ビジネス・イベントやイベント・サブスクリプションを定義する方法やカスタマイズする方法も記載されています。

『Oracle Workflow ユーザーズ・ガイド』
このマニュアルには、Oracle Applications のユーザーがワークフローの通知を表示して応答する方法と、ワークフロー・プロセスの進行状況を監視する方法について記載されています。

『Oracle General Ledger ユーザーズ・ガイド』
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 ユーザーズ・ガイド』
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) インプリメンテーション・ガイド』
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設定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 ユーザーズ・ガイド』
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 ユーザーズ・ガイド』
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 ユーザーズ・ガイド』
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 インプリメンテーション・ガイド』
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*i* 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 ユーザーズ・ガイド』
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 概要』
このマニュアルでは、Oracle Applications リリース 11*i* の概念、機能、テクノロジ・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）、言語セットとキャラクタ・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。

『Oracle Applications のインストール』
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 11*i* 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 および Oracle Database のテクノロジ・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。

『Oracle Applications のアップグレード』

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 11*i* 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 11*i* 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 11*i* に直接アップグレードすることはできません。

Oracle Applications の保守に関するドキュメント・セット

このドキュメント・セットを参考にして、AutoUpgrade、AutoPatch、AD 管理、AD コントローラ、AD 再リンク、ライセンス・マネージャなどの各種 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 システム管理者ガイド』

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 ユーザーズ・ガイド』

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 開発者ガイド』

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 6*i* 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 11*i* 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・オプ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。

『Oracle Applications における複数報告通貨』

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。

『Oracle Applications における複数組織』

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 フレックスフィールド・ガイド』

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 Applications フォーム・ベース製品のユーザー・インタフェース標準』

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 およびオープン・インタフェース・マニュアル』

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 11*i* のドキュメント CD-ROM から HTML 形式で利用できます。

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 Database、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

特に指示がないかぎり、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 Database 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が 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・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・サプライヤであり、世界第 2 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。

1

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。

Oracle Workflow の概要

Oracle Workflow は、ビジネス・プロセスに基づく統合をサポートする完全なワークフロー管理システムを備えています。Oracle Workflow のテクノロジを利用すると、ユーザー定義のビジネス・ルールに基づいてあらゆる種類の情報をルーティングしながら、ビジネス・プロセスをモデル化および自動化し、継続的に改善できます。

E-Business により、社内システムへの取引パートナおよび企業間取引の統合のみならず、企業内のアプリケーションの統合に対する需要が加速しています。Oracle Workflow を導入すると、従来のアプリケーションに基づくワークフローおよび E-Business の統合ワークフローをサポートしながら、企業内および企業間の両方でビジネス・プロセスを自動化して効率よく作業できます。Oracle Workflow は、アプリケーション間の内部プロセスとビジネス・プロセスの連携をもたらす、他に例を見ないワークフロー・ソリューションです。

情報のルーティング

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。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。Oracle Workflow では、企業内外のビジネス・プロセスの各意思決定者に支援情報を提供することができます。

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。

単に文書を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 Database の言語である Java および PL/SQL の機能を活用して、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 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るグラフィカル・ツールです。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 Database に埋め込まれたワークフロー・エンジンでは、実行時にプロセス定義を実装します。ワークフロー・エンジンは、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。通知システムでは、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。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 ページ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 に組み込まれている 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。

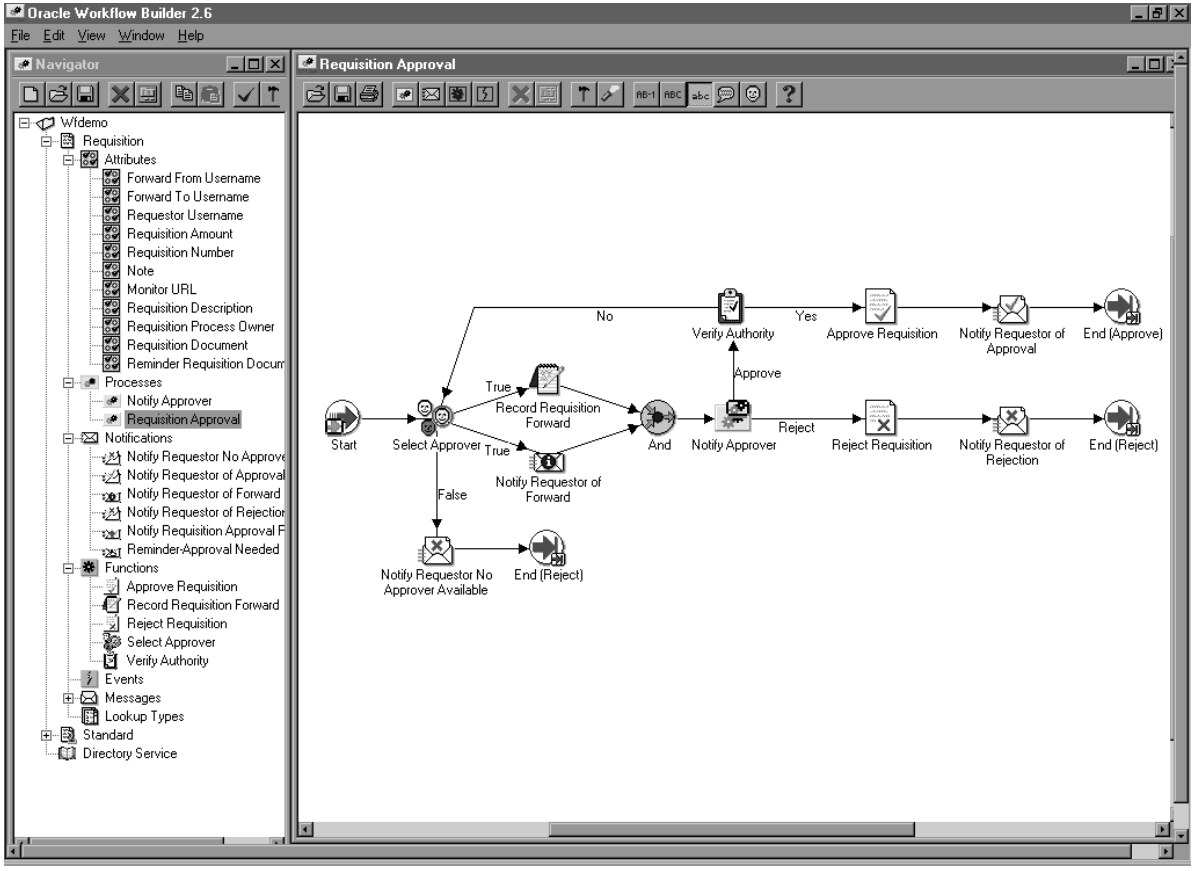
また、スタンドアロン版の Oracle Workflow を使用している場合は、Oracle Enterprise Manager から利用可能なスタンドアロン Oracle Workflow Manager コンポーネントを、Oracle Workflow の追加管理ツールとして使用できます。詳細は、Oracle Workflow Manager のオンライン・ヘルプを参照してください。

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。

2

Workflow Engine API

この章では、ワークフロー・エンジンの API について説明します。API は、ワークフロー・エンジン、ワークフロー・モニター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。

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 Database 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 Database 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。

- 2-20 ページ「CreateProcess」
- 2-28 ページ「StartProcess」
- 2-73 ページ「CompleteActivity」
- 2-76 ページ「CompleteActivityInternalName」
- 2-78 ページ「AssignActivity」
- 2-81 ページ「HandleError」
- 2-33 ページ「SuspendProcess」
- 2-35 ページ「ResumeProcess」
- 2-37 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。
- Deffered（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 Database では、自律型トランザクションをサポートしています。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、再実行可能である必要があります。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、『Oracle9*i* 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 ワークフロー・エンジンおよび通知の 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 WFEngineAPI クラスと WFNotificationAPI 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。2-18 ページの「Workflow Engine API」および 4-12 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。

『Oracle Workflow 開発者ガイド』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」と「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。2-89 ページの「Workflow 関数 API」および 2-97 ページの「Workflow 属性 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 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
```

Oracle Workflow Java API はスレッド内で安全に使用できますが、次のような制限事項があります。

- 各スレッドに独自の WFContext オブジェクトが必要です。したがって、スレッドを開始する前に WFContext をインスタンス化しないでください。コンテキストごとにエラー・スタックが管理されるので、コンテキストを共有することはできません。
- 異なるワークフローに同じ JDBC 接続を使用しないでください。同じ接続を使用すると、関連のないトランザクションのコミットやロールバックで問題が発生する可能性があります。

Oracle Workflow Java API には、同期されたコードはありませんが、共有されているリソースもありません。

Oracle Workflow Java API には接続プーリングもありません。Oracle Applications の場合、接続プーリングは AOL/J レベルで実装されます。JDBC 接続を取得した後、WFContext.setJDBCConnection() API を使用して接続を設定します。この方法を使用すれば、JDBC 接続を Oracle Workflow API の外部で管理できます。

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：** 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 に組み込まれている 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。

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、アクセスされ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「COMPLETE」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切な作業を即時に実行したり、何も処理しないですべてのトランジションに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 アドバンスト・キューと組み合せて、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：

『Oracle Workflow 開発者ガイド』の「アクティビティ・コスト」

『Oracle Workflow 管理者ガイド』の「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・プロセスを割り当てる方法と、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。『Oracle Workflow 開発者ガイド』の「ワークフロー・プロセスのエラー処理」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。『Oracle Workflow 開発者ガイド』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にど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で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かぎ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、最初に実行さ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。『Oracle Workflow 開発者ガイド』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、Oracle Database 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。2-48 ページの「AddItemAttributeArray」および 2-56 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で必要な標準 API と同じものを使用して記述されています。『Oracle Workflow 開発者ガイド』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 ページ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「ロールの拡張」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：

4-2 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。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。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。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、プロセスの項目キーを #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 などの重複した項目キー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 に渡す項目キーは、#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。『Oracle Workflow 管理者ガイド』の「`Wfstat.sql`」を参照してください。

注意： プロセス・インスタンスの項目キーには、シングル・バイト・キャラクタのみ使用できます。マルチバイトの値を含めることはできません。

関連項目：

『Oracle Workflow 管理者ガイド』の「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」

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。5-34 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。5-40 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・メッセージの相関 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。2-79 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。2-84 ページの「SetItemParent」を参照してください。

関連項目：

『Oracle Workflow 開発者ガイド』の「ビジネス・イベントの管理」

『Oracle Workflow 開発者ガイド』の「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。

関連項目：

『Oracle Workflow 開発者ガイド』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」

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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

itemtype 有効な項目タイプ。項目タイプは、Workflow Builder で定義します。

itemkey 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： プロセス・インスタンスの項目キーには、シングル・バイト・キャラクタのみ使用できます。マルチバイトの値を含めることはできません。

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。2-14 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。

process 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タ関数によって、実行する最上位レベルのプロセスが判別されます。セレクタ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。

user_key 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。

owner_role 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れば、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、WFTest.java プログラムからの引用です。

```
// create an item
if (WFEngineAPI.createProcess(ctx, iType, iKey, pr))
  System.out.println("Created Item");
  else
{
  System.out.println("createProcess failed");
  WFEngineAPI.showError(ctx);
}
```

SetItemUserKey

PL/SQL 構文

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

Java 構文

```
public static boolean setItemUserKey
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String userKey)
```

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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

itemtype または itemType
有効な項目タイプ。

itemkey または itemKey
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、プロセスが識別されます。2-20 ページの「CreateProcess」を参照してください。

userkey または userKey
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;
```

Java 構文

```
public static String getItemUserKey
    (WFContext wCtx,
     String itemType,
     String itemKey)
```

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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

itemtype または itemType
有効な項目タイプ。

itemkey または itemKey
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、プロセスが識別されます。2-20 ページの「CreateProcess」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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

itemtype 有効な項目タイプ。項目タイプは、Workflow Builder で定義します。

itemkey 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。

owner 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 (WFEngineAPI.setItemOwner(ctx, iType, iKey, owner))
   System.out.println("Set Item Owner: "+owner);
else
{
  System.out.println("Cannot set owner.");
  WFEngineAPI.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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

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。2-20 ページの「CreateProcess」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れば、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。2-14 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。

注意： プロセス・インスタンスの項目キーには、シングル・バイト・キャラクタのみ使用できます。マルチバイトの値を含めることはできません。

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 (WFEngineAPI.startProcess(ctx, iType, iKey))
  System.out.println("Process Started successfully");
  else
{
  System.out.println("launch failed");
  WFEngineAPI.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。このプロシージャは、CreateProcess と StartProcess を組み合せたラッパーです。

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

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： プロセス・インスタンスの項目キーには、シングル・バイト・キャラクタのみ使用できます。マルチバイトの値を含めることはできません。

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。2-14 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。

process
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タ関数によって、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 NULL です。

userkey
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 NULL 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。

owner
有効なロール。項目の所有者として指定します。所有者が NULL 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れば、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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

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。2-20 ページの「CreateProcess」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 (WFEngineAPI.suspendProcess(ctx, iType, iKey, null))
  System.out.println("Seems to suspend successfully");
else
{
  System.out.println("suspend failed");
  WFEngineAPI.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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

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。2-20 ページの「CreateProcess」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再開されます。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 (WFEngineAPI.resumeProcess(ctx, iType, iKey, null))
  System.out.println("Seems to resume successfully");
else
{
  System.out.println("resume failed");
  WFEngineAPI.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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

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。2-20 ページの「CreateProcess」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します。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 (!WFEngineAPI.abortProcess(ctx, iType, iKey, pr, null))
{
  System.out.println("Seemed to have problem aborting...");
  WFEngineAPI.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）

copy_itemtype 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。

copy_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。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。

new_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。項目タイプと新規項目キーにより、新規プロセスが識別されます。

注意： プロセス・インスタンスの項目キーには、シングル・バイト・キャラクタのみ使用できます。マルチバイトの値を含めることはできません。

---

same_version 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数アクティビティが「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： プロセス・インスタンスの項目キーには、シングル・バイト・キャラクタのみ使用できます。マルチバイトの値を含めることはできません。

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。

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。『Oracle Workflow 管理者ガイド』の「`wfbkgchk.sql`」を参照してください。

アプリケーション・コード内から Background() をコールしないでください。このプロシージャを直接コールする場合は、SQL*Plus からこのプロシージャを実行します。あるいは、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・エンジンのサンプル・ループ・スクリプトの 1 つを使用するか、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か、Oracle Enterprise Manager の Oracle Workflow Manager コンポーネントを使用してバックグラウンド・エンジンをスケジュールします。Oracle Applications に組み込まれている 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。Oracle Applications Manager の Workflow Manager コンポーネントを使用して、バックグラウンド・エンジン・コンカレント・プログラムを発行することもできます。『Oracle Workflow 管理者ガイド』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）

itemtype 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。

minthreshold	（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。
maxthreshold	（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。
process_deferred	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。
process_timeout	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。
process_stuck	スタック状態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 Database 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 Database 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
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String aName)

public static boolean addItemAttrText
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String aName,
     String aValue)

public static boolean addItemAttrNumber
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String aName,
     BigDecimal numberVal)

public static boolean addItemAttrDate
    (WFContext 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。

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。2-48 ページの「AddItemAttributeArray」を参照してください。

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

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。2-20 ページの「CreateProcess」を参照してください。

aname 項目タイプ属性の内部名。

text_value 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。

number_value または numberVal 項目タイプ属性のデフォルトの数値。PL/SQL プロシージャと addItemAttrNumber() Java メソッドの場合にのみ必須です。デフォルト値は NULL です。

date_value 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。

aValue 項目タイプ属性のデフォルト値。addItemAttrText() Java メソッドと addItemAttrDate() Java メソッド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 2-1

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）

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。2-20 ページの「CreateProcess」を参照してください。
aname	新しい項目タイプ属性の内部名の配列。
avalue	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)

public static boolean setItemAttrDate
     (WFContext wCtx,
      String itemType,
      String itemKey,
      String attributeName,
      java.util.Date attributeValue)
```

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。

Java での setItemAttrDate() の実装には、2 種類あります。Java String オブジェクトとして日付値を指定する実装と、Java Date オブジェクトとして日付値を指定する実装です。

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。2-56 ページの「SetItemAttributeArray」を参照してください。

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

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。2-20 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できます。2-14 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。

---

aname、name または attributeName　項目タイプ属性の内部名。

avalue、event または attributeValue　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
{
  WFEngineAPI.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：

5-8 ページ「イベント・メッセージ構造」

setItemAttrFormattedDate

Java 構文

```
public static boolean setItemAttrFormattedDate
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String attributeName,
     String attributeValue
     String dateFormat)
```

説明

プロセス内の日付型の項目タイプ属性の値に、書式付き文字列として指定された日付値を設定します。

引数（入力）

wCtx ワークフローのコンテキスト情報。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

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。2-20 ページの「CreateProcess」を参照してください。

attributeName 項目タイプ属性の内部名。

attributeValue 項目タイプ属性の日付値。

dateFormat 日付値の書式。Oracle Database でサポートされている日付の書式マスクを指定する必要があります。書式を指定しない場合、デフォルト値はデータベースの標準の日付書式になります。『Oracle Database グローバリゼーション・サポート・ガイド』の「日付書式」を参照してください。

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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

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。2-20 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できます。2-14 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。

aname　項目タイプ属性の内部名。

documentid　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。

DM:<*node_id*>:<*doc_id*>:<*version*>

<*node_id*> は、「文書管理ノード」Web ページで定義されている、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。

<*doc_id*> 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。

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 2-2

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）

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。2-20 ページの「CreateProcess」を参照してください。

aname 項目タイプ属性の内部名の配列。

avalue 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 WFTwoDArray 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。2-5 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,
     ignore_notfound in boolean default FALSE)
     return varchar2;

function GetItemAttrNumber
    (itemtype in varchar2,
     itemkey in varchar2,
     aname in varchar2,
     ignore_notfound in boolean default FALSE)
     return number;

function GetItemAttrDate
    (itemtype in varchar2,
     itemkey in varchar2,
     aname in varchar2,
     ignore_notfound in boolean default FALSE) 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。

GetItemAttrText()、GetItemAttrNumber() および GetItemAttrDate() に対し、ignore_notfound パラメータに TRUE を指定すると、指定した項目タイプ属性が存在しない場合、発生した例外は無視されます。この場合、関数は NULL 値を戻しますが、例外は発生しません。たとえば、ある項目タイプに新しい項目タイプ属性を追加したとき、この項目タイプの旧バージョンと新バージョンの両方をコードで処理する必要がある場合に、このパラメータを使用できます。

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。2-20 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。2-14 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。

---

aname 項目タイプ属性の内部名（GetItemAttrText()、GetItemAttrNumber() および GetItemAttrDate() の場合）。

name 項目タイプ属性の内部名（GetItemAttrEvent() の場合）。

ignore_notfound GetItemAttrText()、GetItemAttrNumber() および GetItemAttrDate() に対し、指定した項目タイプ属性が存在しない場合に例外を無視するかどうかを TRUE または FALSE で指定します。TRUE を指定すると、指定した項目タイプ属性が存在しない場合、関数は NULL 値を戻しますが、例外は発生しません。デフォルト値は FALSE です。

関連項目：

5-8 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,
     ignore_notfound in boolean default FALSE)
     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 ページで定義されている、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。

ignore_notfound パラメータに TRUE を指定すると、指定した項目タイプ属性が存在しない場合、発生した例外は無視されます。この場合、関数は NULL 値を戻しますが、例外は発生しません。たとえば、ある項目タイプに新しい項目タイプ属性を追加したとき、この項目タイプの旧バージョンと新バージョンの両方をコードで処理する必要がある場合に、このパラメータを使用できます。

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。2-20 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。2-14 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。

---

aname 項目タイプ属性の内部名。

ignore_notfound 指定した項目タイプ属性が存在しない場合に例外を無視するかどうかを TRUE または FALSE で指定します。TRUE を指定すると、指定した項目タイプ属性が存在しない場合、関数は NULL 値を戻しますが、例外は発生しません。デフォルト値は FALSE です。

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）

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。2-20 ページの「CreateProcess」を参照してください。

aname 項目タイプ属性の内部名。

getItemAttributes

Java 構文

```
public static WFTwoDArray 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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

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。2-20 ページの「CreateProcess」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、それを戻します。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）

itemtype 有効な項目タイプ。

aname 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、それを戻します。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戻しません。

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。2-20 ページの「CreateProcess」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。

GetActivityAttribute

PL/SQL 構文

```
function GetActivityAttrText
    (itemtype in varchar2,
     itemkey in varchar2,
     actid in number,
     aname in varchar2,
     ignore_notfound in boolean default FALSE)
     return varchar2;

function GetActivityAttrNumber
    (itemtype in varchar2,
     itemkey in varchar2,
     actid in number,
     aname in varchar2,
     ignore_notfound in boolean default FALSE)
     return number;

function GetActivityAttrDate
    (itemtype in varchar2,
     itemkey in varchar2,
     actid in number,
     aname in varchar2,
     ignore_notfound in boolean default FALSE) 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。

GetActivityAttrText()、GetActivityAttrNumber() および GetActivityAttrDate() に対し、ignore_notfound パラメータに TRUE を指定すると、指定したアクティビティ属性が存在しない場合、発生した例外は無視されます。この場合、関数は NULL 値を戻しますが、例

外は発生しません。たとえば、あるアクティビティに新しいアクティビティ属性を追加したとき、このアクティビティの旧バージョンと新バージョンの両方をコードで処理する必要がある場合に、このパラメータを使用できます。

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。2-20 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。2-14 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。

---

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）。

ignore_notfound GetActivityAttrText()、GetActivityAttrNumber() および GetActivityAttrDate() に対し、指定したアクティビティ属性が存在しない場合に例外を無視するかどうかを TRUE または FALSE で指定します。TRUE を指定すると、指定したアクティビティ属性が存在しない場合、関数は NULL 値を戻しますが、例外は発生しません。デフォルト値は FALSE です。

関連項目：

5-8 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）

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。2-20 ページの「CreateProcess」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。

getActivityAttributes

Java 構文

```
public static WFTwoDArray getActivityAttributes
    (WFContext wCtx,
     String itemType,
     String itemKey,
     BigDecimal actID)
```

説明

指定されたアクティビティに対するすべてのアクティビティ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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

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。2-20 ページの「CreateProcess」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。

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。2-73 ページの「CompleteActivity」を参照してください。

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。

activity 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、*<parent_process_internal_name>:<label_name>* のように指定します。

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 in varchar2);
```

Java 構文

```
public static boolean completeActivity
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String activity,
     String result)
```

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、Workflow Builder 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() を使用する方法と、CreateProcess() および StartProcess() を使用する方法には、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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

itemtype または itemType
有効な項目タイプ。

itemkey または 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。

activity
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。

result
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。2-37 ページの「AbortProcess」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。2-73 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）

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。

activity 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、*<parent_process_internal_name>:<activity_internal_ name>* のように指定します。

result 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。2-37 ページの「AbortProcess」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）

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。

activity 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、*<parent_process_internal_name>:<label_name>* のように指定します。

performer 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。2-84 ページの「SetItemParent」を参照してください。

引数（入力）

itemtype 有効な項目タイプ。

itemkey 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。

注意： プロセス・インスタンスの項目キーには、シングル・バイト・キャラクタのみ使用できます。マルチバイトの値を含めることはできません。

process_name 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。

event_message 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);
```

Java 構文

```
public static boolean handleError
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String activity,
     String command,
     String result)
```

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の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。

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て、すでに遷移した後続のすべてのアクティビティも消去します。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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。
item_type または itemType	有効な項目タイプ。
item_key または 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。
activity	エラーが発生したか、または元に戻すアクティビティ・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、*<parent_process_internal_name>:<label_name>* のように指定します。
command	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。 **SKIP**: 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。 **RETRY**: 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。
result	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);
```

Java 構文

```
public static boolean setItemParent
    (WFContext wCtx,
     String itemType,
     String itemKey,
     String parentItemType,
     String parentItemKey,
     String parentContext)
```

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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

itemtype または itemType
有効な項目タイプ。

itemkey または 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、子プロセスが識別されます。

注意： プロセス・インスタンスの項目キーには、シングル・バイト・キャラクタのみ使用できます。マルチバイトの値を含めることはできません。

---

parent_itemtype または parentItemType 親プロセスの有効な項目タイプ。

parent_itemkey または parentItemKey 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。

注意： プロセス・インスタンスの項目キーには、シングル・バイト・キャラクタのみ使用できます。マルチバイトの値を含めることはできません。

---

parent_context または parentContext 親プロセスに複数の「フロー待ち」アクティビティが含まれている場合、このパラメータには、このディテール・プロセスに対応する「フロー待ち」アクティビティ・ノードのアクティビティ・ラベル名を指定します。親プロセスに含まれている「フロー待ち」アクティビティが 1 つのみの場合、このパラメータは NULL のままでかまいません。

ItemStatus

PL/SQL 構文

```
procedure ItemStatus
    (itemtype in varchar2,
     itemkey in varchar2,
     status out varchar2,
     result out varchar2);
```

Java 構文

```
public static WFTwoDArray 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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

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、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 = WFEngineAPI.itemStatus(ctx, iType, iKey);
System.out.print("Status and result for " + iType + "/" + iKey + " = ");
displayDataSource(ctx, dataSource);
```

getProcessStatus

Java 構文

```
public static WFTwoDArray 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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

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。2-20 ページの「CreateProcess」を参照してください。

process 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 関数 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。

関連項目：

『Oracle Workflow 開発者ガイド』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」

『Oracle Workflow 開発者ガイド』の「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。2-5 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）

pWCtx ワークフローのコンテキスト情報。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

iType 有効な項目タイプ。

iKey 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。2-20 ページの「CreateProcess」を参照してください。

actid 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。2-95 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。2-90 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）

pWCtx 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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

aName 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）

aName 項目属性の内部名。

setItemAttrValue

Java 構文

```
public void setItemAttrValue
    (WFContext pWCtx,
     WFAttribute 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）

pWCtx ワークフローのコンテキスト情報。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

pAttr 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。『Oracle Workflow 開発者ガイド』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。

引数（入力）

pWCtx ワークフローのコンテキスト情報。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

Workflow 属性 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。

関連項目：

『Oracle Workflow 開発者ガイド』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」

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 2-3

定数変数宣言	定数値
`public static final String TEXT`	"TEXT"
`public static final String NUMBER`	"NUMBER"

表 2-3（続き）

定数変数宣言	定数値
`public static final String DATE`	"DATE"
`public static final String LOOKUP`	"LOOKUP"
`public static final String FORM`	"FORM"
`public static final String URL`	"URL"
`public static final String DOCUMENT`	"DOCUMENT"
`public static final String ROLE`	"ROLE"
`public static final String EVENT`	"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 2-4

定数変数宣言	定数値
`public static final String CONSTANT`	"CONSTANT"
`public static final String ITEMATTR`	"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）

pName	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。
pType	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。
pValue	属性値。2 番目のメソッドの場合にのみ必須です。
pValueType	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。2-95 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。『Oracle Workflow 開発者ガイド』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。『Oracle Workflow 開発者ガイド』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。『Oracle Workflow 開発者ガイド』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）

pValue　　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。『Oracle Workflow 開発者ガイド』の「ワークフロー・プロセスのエラー処理」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。

- 2-109 ページ「CLEAR」
- 2-110 ページ「GET_ERROR」
- 2-111 ページ「TOKEN」
- 2-112 ページ「RAISE」
- 2-116 ページ「CONTEXT」
- 2-118 ページ「TRANSLATE」

関連項目：

『Oracle Workflow 開発者ガイド』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」

CLEAR

構文

```
procedure CLEAR;
```

説明

エラー・バッファを消去します。

関連項目：

2-110 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：

2-109 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：

2-112 ページ「RAISE」

2-116 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させます。

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。

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 に組み込まれている Workflow の場合は、サーバー上の $FND_TOP の import/<*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：

2-111 ページ「TOKEN」

2-116 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。

注意： ロードする言語を決定するために、ワークフロー・リソース・ジェネレータは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い場合、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 環境変数に設定されます。

NLS_LANG を設定するには、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、変更内容を保存します。

Oracle Applications に組み込まれている 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、「要求の発行」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「要求の発行」フォームから実行するには、「要求の発行」フォームに移動します。

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 システム管理者ガイド』の「コンカレント・プログラムおよびコンカレント要求の概要」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 ユーザーズ・ガイド』の「要求の発行」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、「送信」を選択して要求を発行します。
7. 「要求の発行」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したり、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）

pkg_name プロシージャのパッケージ名

proc_name プロシージャ名または関数名

arg1 最初の IN 引数

argn 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), funmode);
    raise;
end MyFunction;
```

関連項目：

2-111 ページ「TOKEN」

2-112 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）

tkn_name 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「永久」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。『Oracle Workflow 開発者ガイド』の「維持タイプ」を参照してください。

> **注意：** 将来の終了日に WF_PURGE API を実行することはできません。将来の終了日を指定すると、「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。将来の終了日を指定すると、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。

- 2-121 ページ「Items」
- 2-122 ページ「Activities」
- 2-123 ページ「Notifications」
- 2-124 ページ「Total」
- 2-125 ページ「TotalPERM」
- 2-126 ページ「Directory」

注意： Oracle Workflow の以前のバージョンの AdHocDirectory() API は、Directory() API に変更になりました。下位互換性を保つために Oracle Workflow の現行バージョンでも AdHocDirectory() API は認識されますが、将来的にも、該当する場合は Directory() API のみを使用するようにしてください。

注意： Oracle Applications に組み込まれている 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。2-127 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： 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、ワークフローのパージ・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。

関連項目：

『Oracle Workflow 開発者ガイド』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」

『Oracle Workflow 管理者ガイド』の「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）

itemtype 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。

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。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。

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。

docommit 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・サイズ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、指定された終了日以前の END_DATE が定義されているものを削除します。Total() は、ワークフローのローカル表で期限切れとなり使用されなくなったユーザーとロールも、Directory() をコールして削除します。2-126 ページの「Directory」を参照してください。

Total() では、すべてのアクティビティと期限切れのユーザーおよびロールの情報が削除されるため、Items() に比べパフォーマンスが犠牲になります。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。2-121 ページの「Items」を参照してください。

引数（入力）

itemtype 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。

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。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。

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。

docommit 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・サイズ数を減らしてパフォーマンスを改善するために、Total() 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」（永久）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。2-124 ページの「Total」を参照してください。

引数（入力）

itemtype 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。

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。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。

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。

docommit 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・サイズ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。

Directory

構文

```
procedure Directory
    (end_date in date default sysdate);
```

説明

WF_LOCAL_ROLES 表および WF_LOCAL_USER_ROLES 表で、指定された終了日以前の失効日が定義されており、どの通知でも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。

失効日を過ぎたユーザーおよびロールは、WF_USERS、WF_ROLES および WF_USER_ROLES のシードされたビューには表示されませんが、Directory() によって削除されるまでワークフローのローカル表には残っています。パフォーマンスを改善するために、期限切れのユーザーとロールを定期的に削除してください。

引数（入力）

end_date 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 に組み込まれている 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「要求の発行」フォームを使用します。

注意： Oracle Applications に組み込まれている 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「要求の発行」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。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 Oracle Workflow Purge Obsolete Data で、短縮名は FNDWFPR です。『Oracle Applications システム管理者ガイド』の「コンカレント・プログラムおよびコンカレント要求の概要」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 ユーザーズ・ガイド』の「要求の発行」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」（永久）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、「送信」を選択して要求を発行します。

Workflow Monitor API

次の API をコールして、アクセス・キーを取得したり、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。Workflow Monitor API は、WF_MONITOR という PL/SQL パッケージに定義されています。

注意： Oracle Workflow の以前のバージョンの 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す。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）

x_item_type 有効な項目タイプ。

x_item_key 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。

x_admin_mode 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。

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。『Oracle Workflow 管理者ガイド』の「グローバル・ユーザー設定の設定」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）

x_agent
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。
`http://<server.com:portID>/<PLSQL_agent_path>`

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。『Oracle Workflow 管理者ガイド』の「グローバル・ユーザー設定の設定」を参照してください。

x_item_type
有効な項目タイプ。

x_item_key
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。

x_admin_mode
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：

2-118 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。『Oracle Workflow 管理者ガイド』の「グローバル・ユーザー設定の設定」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）

x_agent
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。
http://*<server.com:portID>*/*<PLSQL_agent_path>*

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。『Oracle Workflow 管理者ガイド』の「グローバル・ユーザー設定の設定」を参照してください。

x_item_type
有効な項目タイプ。

x_item_key	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。
x_admin_mode	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：

2-118 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。『Oracle Workflow 管理者ガイド』の「グローバル・ユーザー設定の設定」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）

x_agent
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。
`http://<server.com:portID>/<PLSQL_agent_path>`

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。『Oracle Workflow 管理者ガイド』の「グローバル・ユーザー設定の設定」を参照してください。

x_item_type
有効な項目タイプ。

x_item_key
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。

x_admin_mode
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。

x_options
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：

2-118 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 に組み込まれている 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。

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。次の表でビューの列を説明します。

表 2-5

ビュー名	Null?	型
ROWID		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)

表 2-5（続き）

ビュー名	Null?	型
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)

注意： プロセス・インスタンスの項目キーには、シングル・バイト・キャラクタのみ使用できます。マルチバイトの値を含めることはできません。

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。『Oracle Workflow 開発者ガイド』の「投票アクティビティ」を参照してください。

次の表でビューの列を説明します。

表 2-6

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。

次の表でビューの列を説明します。

表 2-7

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。

次の表でビューの列を説明します。

表 2-8

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

注意： プロセス・インスタンスの項目キーには、シングル・バイト・キャラクタのみ使用できます。マルチバイトの値を含めることはできません。

3

ディレクトリ・サービス API

この章では、Oracle Workflow ディレクトリ・サービスの API について説明します。API は、ディレクトリ・サービスへのアクセスに使用できる PL/SQL 関数とプロシージャで構成されています。

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。

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_ROLES 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：

『Oracle Workflow 開発者ガイド』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」

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。

注意： ロールには、個々のユーザーのみをメンバーとして加えることができます。ロールに別のロールを含めることはできません。

引数（入力）

role 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）

user 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、MAILHTM2、SUMMARY）
- 言語
- 地域

注意： Oracle Applications では、元のシステムとして FND_USR または PER を持つ Oracle Applications ユーザーのロールについて、GetRoleInfo() プロシージャは、デフォルトで、その Oracle Applications ユーザーの「ICX: 言語」プロファイル・オプションと「ICX: テリトリ」プロファイル・オプションから言語と地域の値を取得します。

ただし、WF_PREFERENCE リソース・トークンが定義され FND に設定されている場合、GetRoleInfo() プロシージャは、Oracle Workflow 環境設定表から言語と地域の値を取得します。この表は、Oracle Workflow の「グローバル・ワークフロー設定」Web ページまたは「ユーザー設定」Web ページを使用して設定できます。

WF_PREFERENCE リソース・トークンは、スタンドアロン版の Oracle Workflow では使用されません。

引数（入力）

role 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、MAILHTM2、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

注意： Oracle Applications では、元のシステムとして FND_USR または PER を持つ Oracle Applications ユーザーのロールについて、GetRoleInfo2() プロシージャは、デフォルトで、その Oracle Applications ユーザーの「ICX: 言語」プロファイル・オプションと「ICX: テリトリ」プロファイル・オプションから言語と地域の値を取得します。

ただし、WF_PREFERENCE リソース・トークンが定義され FND に設定されている場合、GetRoleInfo2() プロシージャは、Oracle Workflow 環境設定表から言語と地域の値を取得します。この表は、Oracle Workflow の「グローバル・ワークフロー設定」Web ページまたは「ユーザー設定」Web ページを使用して設定できます。

WF_PREFERENCE リソース・トークンは、スタンドアロン版の Oracle Workflow では使用されません。

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（メンバー）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）

user 有効なユーザー名。

role 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 を戻します。

引数（入力）

username　　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）

p_orig_system　元のリポジトリ表を識別するコード。

p_orig_system_id　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）

p_orig_system 元のリポジトリ表を識別するコード。

p_orig_system_id 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）

p_role_name　　ロールの内部名。

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_ROLES 表に値を作成し、ユーザー・フラグを Y に設定することで、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： スタンドアロン版の Oracle Workflow に 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_ROLES 表に新しいユーザーを作成しないでください。

引数（入力）

name ユーザーの内部名。内部名の長さは 320 文字以下にする必要があります。内部名には大文字のみを使用することをお薦めします。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。

display_name 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。

language 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。

territory 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。

description ユーザーの説明（オプション）。

notification_preference 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、MAILHTM2、QUERY または SUMMARY です。NULL の場合は、MAILHTML に設定されます。

email_address ユーザーの電子メール・アドレス（オプション）。

fax ユーザーの FAX 番号（オプション）。

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。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。

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。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：

『Oracle Workflow 管理者ガイド』の「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、ユーザー・フラグを N に設定することで、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）

role_name ロールの内部名。内部名の長さは 320 文字以下にする必要があります。内部名には大文字のみを使用することをお薦めします。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。

role_display_name 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。

language 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。

territory 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。

role_description ロールの説明（オプション）。

notification_preference	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、MAILHTM2、QUERY または SUMMARY です。NULL の場合は、MAILHTML に設定されます。
role_users	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。
email_address	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）。
fax	ロールの FAX 番号（オプション）。
status	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。
expiration_date	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：

『Oracle Workflow 管理者ガイド』の「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。

注意： ロールには、個々のユーザーのみをメンバーとして加えることができます。ロールに別のロールを含めることはできません。

引数（入力）

role_name アドホック・ロールの内部名。

role_users 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る必要があります。

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）

role_name アドホック・ロールの内部名。

role_users 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。

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_ROLES 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 ユーザーの内部名。

status 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）

role_name ロールの内部名。

status 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。

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。

失効日の過ぎたユーザーやロールは、WF_USERS、WF_ROLES および WF_USER_ROLES の各ビューには表示されませんが、AdHocDirectory() によって削除されるまでワークフローのローカル表には残っています。パフォーマンスを改善するために、期限切れのユーザーとロールを定期的に削除してください。2-126 ページの「Directory」を参照してください。

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_ROLES 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。

失効日の過ぎたユーザーやロールは、WF_USERS、WF_ROLES および WF_USER_ROLES の各ビューには表示されませんが、AdHocDirectory() によって削除されるまでワークフローのローカル表には残っています。パフォーマンスを改善するために、期限切れのユーザーとロールを定期的に削除してください。2-126 ページの「Directory」を参照してください。

引数（入力）

role_name	アドホック・ロールの内部名。
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。

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_ROLES 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 新しい通知環境設定を表します。有効値は MAILTEXT、MAILHTML、MAILATTH、MAILHTM2、QUERY または SUMMARY です。NULL の場合、通知環境設定は更新されません。

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）

role_name 更新するアドホック・ロールの内部名。

display_name 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。

notification_preference 新しい通知環境設定を表します。有効値は MAILTEXT、MAILHTML、MAILATTH、MAILHTM2、QUERY または SUMMARY です。NULL の場合、通知環境設定は更新されません。

language 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。

territory 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。

email_address 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。

fax 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。

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。

関連項目：

『Oracle Workflow 管理者ガイド』の「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.global.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・ワークフロー設定」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：

『Oracle Workflow 管理者ガイド』の「Workflow ディレクトリ・サービスと Oracle Internet Directory の同期」

『Oracle Workflow 管理者ガイド』の「グローバル・ユーザー設定の設定」

『Oracle Workflow 開発者ガイド』の「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.global.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・ワークフロー設定」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 を戻します。

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：

『Oracle Workflow 管理者ガイド』の「Workflow ディレクトリ・サービスと Oracle Internet Directory の同期」

『Oracle Workflow 管理者ガイド』の「グローバル・ユーザー設定の設定」

『Oracle Workflow 開発者ガイド』の「User Entry Has Changed イベント」

3-27 ページ「Synch_changes」

3-29 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 を繰り返し実行します。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）

l_day	Synch_changes() API の実行間隔の日数。デフォルト値はゼロです。
l_hour	Synch_changes() API の実行間隔の時間数。デフォルト値はゼロです。
l_minute	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：

3-27 ページ「Synch_changes」

『Oracle Workflow 管理者ガイド』の「Workflow ディレクトリ・サービスと Oracle Internet Directory の同期」

Workflow ローカル同期 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、アプリケーションの表に保存されているユーザーおよびロールの情報と、ワークフローのローカル表内の情報とを同期させます。これらの API は、WF_LOCAL_SYNCH という PL/SQL パッケージに定義されています。

- 3-31 ページ「Propagate_User」
- 3-35 ページ「Propagate_Role」
- 3-39 ページ「Propagate_User_Role」

関連項目：

『Oracle Workflow 管理者ガイド』の「Oracle Workflow のディレクトリ・サービスの設定」

Propagate_User

構文

```
procedure Propagate_User
    (p_orig_system in varchar2,
     p_orig_system_id in number,
     p_attributes in wf_parameter_list_t,
     p_start_date in date default null,
     p_expiration_date in date default null);
```

説明

アプリケーション表に保存されているユーザーの情報と WF_LOCAL_ROLES 表とを同期させ、このレコードが個々のユーザーであることを示すためにユーザー・フラグを Y に設定します。元のシステムおよび元のシステム ID を指定することにより、ユーザーが識別されます。ユーザーの情報が保存されているパーティションの ID は、元のシステムに応じて自動的に設定されます。

> **注意：** Oracle Applications では、Propagate_User() を使用して、FND_USER 表に保存されている Oracle Applications ユーザー、Oracle Trading Community Architecture (TCA) 個人パーティおよび TCA 担当者（関連パーティ）のみを同期させるようにしてください。Oracle Applications の他のモジュールで情報を同期させる場合は、Propagate_Role() を使用します。

WF_LOCAL_ROLES 表に保存するユーザー情報は、WF_PARAMETER_LIST_T の形式で指定する必要があります。WF_EVENT.AddParameterToList() API を使用して、このリストに属性を追加できます。次の表は、このリストに含める必要のある属性の一覧です。これらは、WF_LOCAL_ROLES の必須の列に挿入します。これらの属性には、標準の LDAP 属性名を使用してください。

表 3-1

データベースの列	属性名
NAME	[USER_NAME]
DISPLAY_NAME	[DisplayName]
DESCRIPTION	[description]
NOTIFICATION_ PREFERENCE	[orclWorkFlowNotificationPref]
LANGUAGE	[preferredLanguage]

表 3-1（続き）

データベースの列	属性名
TERRITORY	[orclNLSTerritory]
EMAIL_ADDRESS	[mail]
FAX	[FacsimileTelephoneNumber]
STATUS	[orclIsEnabled]
EXPIRATION_DATE	[ExpirationDate]
ORIG_SYSTEM	[orclWFOrigSystem]
ORIG_SYSTEM_ID	[orclWFOrigSystemID]

標準の動作モードでは、これらの属性のいずれか（USER_NAME は除く）が属性リストに渡されない場合または NULL に設定されている場合、WF_LOCAL_ROLES でその属性に対応するフィールドの既存の値は変更されません。たとえば、電子メール・アドレスが渡されない場合は、ユーザーの既存の電子メール・アドレスが保持されます。ただし、USER_NAME 属性は常に渡す必要があります。Propagate_User() プロシージャは、WHERE 条件にこの値を使用するため、USER_NAME が指定されていないと失敗します。また、ユーザー・レコードがまだ存在していない場合は、使用する既存の値がないため、前述の属性をすべて渡す必要があります。

コードの信頼性を高めるために、Propagate_User() をコールするときは、必ず前述の属性をすべて渡してください。このようにすると、渡す属性を動的に決定するときに発生するエラーを回避できます。

> **注意：** 標準の動作モードで最初にユーザー・レコードが作成されるとき、属性リストに表示名が指定されていない場合は、WF_LOCAL_ROLES 内のユーザー・レコードの値を *<orig_system>:<orig_system_ID>* という形式で組み合せた値がデフォルト値として設定されます。また、通知環境設定が指定されていない場合、ユーザー・レコードの通知環境設定はデフォルトで MAILHTML に設定され、ステータスが指定されていない場合、ユーザー・レコードのステータスはデフォルトで ACTIVE に設定されます。

WFSYNCH_OVERWRITE という特殊な属性を追加し、値を TRUE に指定することで、Propagate_User() を上書きモードでコールすることもできます。上書きモードでは、次のいずれかの属性の値が渡されない場合または NULL に設定されている場合、WF_LOCAL_ROLES でその属性に対応するフィールドの値は NULL に設定され、既存の値は削除されます。

- description

- preferredLanguage
- orclNLSTerritory
- mail
- FacsimileTelephoneNumber
- ExpirationDate

したがって、上書きモードを使用する場合は、NULL に設定する属性を除き、すべての属性に値を渡す必要があります。また、USER_NAME 属性は常に指定する必要があります。

注意： WF_LOCAL_ROLES 表の DISPLAY_NAME、NOTIFICATION_PREFERENCE、STATUS、ORIG_SYSTEM および ORIG_SYSTEM_ID の各列は、NULL に設定することはできません。したがって、これらの列では、対応する属性に値を渡さないと、上書きモードの場合でも既存の値が保持されます。

WF_LOCAL_ROLES 表の NAME 列も NULL に設定することはできませんが、USER_NAME 属性はこの API に必須なので省略できません。

元のシステム、元のシステム ID、失効日などの一部の値は、Propagate_User() API のパラメータとして渡すことも、属性リスト・パラメータで属性として渡すこともできます。LDAP 統合を管理する Entity Manager には、Propagate_User() から属性リストだけが送られ、プロシージャの他のパラメータは渡されないため、これらの値は属性リストにも渡されます。

- 元のシステムおよび元のシステム ID の値では、属性リストで属性として指定された値よりも、プロシージャにパラメータとして渡される値が優先されます。
- 同様に、失効日の値では、属性リストで属性として指定された値よりも、プロシージャにパラメータとして渡される値が優先されます。ただし、`p_expiration_date` パラメータが NULL の場合は、ExpirationDate 属性の値が指定されていればそれが使用されます。ExpirationDate 属性の値は次の形式で指定する必要があります。

```
to_char(<your date variable>, WF_ENGINE.Date_Format)
```

Oracle Workflow には他にも 2 つの特殊属性が用意されており、これらを使用してユーザー情報の変更方法を指定できます。

- DELETE: この属性は、ユーザーがワークフローに参加できないようにする場合に使用します。この属性を追加して `TRUE` の値を指定すると、WF_LOCAL_ROLES でユーザーの失効日として sysdate が設定され、ステータスは `INACTIVE` に設定されます。

注意： ただし、p_expiration_date パラメータにも値を指定した場合は、その値が DELETE 属性よりも優先されます。また、p_expiration_date パラメータが NULL でも、ExpirationDate 属性を指定した場合は、この属性が DELETE 属性よりも優先されます。このような場合、指定した失効日まで、ユーザーは有効でアクティブなままになります。

- UpdateOnly: この属性は、パフォーマンス改善のため、WF_LOCAL_ROLES にすでにレコードが存在するユーザーの情報を変更する場合に使用します。この属性を追加して TRUE の値を指定すると、Propagate_User() API はレコードの挿入を行わず、レコードを直接更新しようとします。

  そのユーザーのレコードが存在しないために更新が失敗する場合、プロシージャはレコードを挿入します。ただし、最初の更新が失敗するとパフォーマンスが低下するため、WF_LOCAL_ROLES にユーザー・レコードが確実に存在している場合にのみ UpdateOnly 属性を使用してください。

引数（入力）

p_orig_system	ユーザー情報の提供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。
p_orig_system_id	このリポジトリ・システムのユーザーを識別する主キー。
p_attributes	ユーザーに関する情報が含まれている、属性の名前と値のペアのリスト。
p_start_date	ディレクトリ・サービスにおいて、ユーザーが有効になる日付。
p_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。

関連項目：

5-50 ページ「AddParameterToList」

Propagate_Role

構文

```
procedure Propagate_Role
    (p_orig_system in varchar2,
     p_orig_system_id in number,
     p_attributes in wf_parameter_list_t,
     p_start_date in date default null,
     p_expiration_date in date default null);
```

説明

アプリケーション表に保存されているロールの情報と WF_LOCAL_ROLES 表とを同期させ、このロールのユーザー・フラグを N に設定します。元のシステムおよび元のシステム ID を指定することにより、ロールが識別されます。ロールの情報が保存されているパーティションの ID は、元のシステムに応じて自動的に設定されます。

WF_LOCAL_ROLES 表に保存するロール情報は、WF_PARAMETER_LIST_T の形式で指定する必要があります。WF_EVENT.AddParameterToList() API を使用して、このリストに属性を追加できます。次の表は、このリストに含める必要のある属性の一覧です。これらは、WF_LOCAL_ROLES の必須の列に挿入します。これらの属性には、標準の LDAP 属性名を使用してください。

表 3-2

データベースの列	属性名
NAME	[USER_NAME]
DISPLAY_NAME	[DisplayName]
DESCRIPTION	[description]
NOTIFICATION_ PREFERENCE	[orclWorkFlowNotificationPref]
LANGUAGE	[preferredLanguage]
TERRITORY	[orclNLSTerritory]
EMAIL_ADDRESS	[mail]
FAX	[FacsimileTelephoneNumber]
STATUS	[orclIsEnabled]
EXPIRATION_DATE	[ExpirationDate]
ORIG_SYSTEM	[orclWFOrigSystem]
ORIG_SYSTEM_ID	[orclWFOrigSystemID]

標準の動作モードでは、これらの属性のいずれか（USER_NAME は除く）が属性リストに渡されない場合または NULL に設定されている場合、WF_LOCAL_ROLES でその属性に対応するフィールドの既存の値は変更されません。たとえば、電子メール・アドレスが渡されない場合は、ロールの既存の電子メール・アドレスが保持されます。ただし、USER_NAME 属性は常に渡す必要があります。Propagate_Role() プロシージャは、WHERE 条件にこの値を使用するため、USER_NAME が指定されていないと失敗します。また、ユーザー・レコードがまだ存在していない場合は、使用する既存の値がないため、前述の属性をすべて渡す必要があります。

コードの信頼性を高めるために、Propagate_Role() をコールするときは、必ず前述の属性をすべて渡してください。このようにすると、渡す属性を動的に決定するときに発生するエラーを回避できます。

注意： 標準の動作モードで最初にロール・レコードが作成されるとき、属性リストに表示名が指定されていない場合は、WF_LOCAL_ROLES 内のロール・レコードの値を *<orig_system>*:*<orig_system_ID>* という形式で組み合せた値がデフォルト値として設定されます。また、通知環境設定が指定されていない場合、ロール・レコードの通知環境設定はデフォルトで MAILHTML に設定され、ステータスが指定されていない場合、ロール・レコードのステータスはデフォルトで ACTIVE に設定されます。

WFSYNCH_OVERWRITE という特殊な属性を追加し、値を TRUE に指定することで、Propagate_Role() を上書きモードでコールすることもできます。上書きモードでは、次のいずれかの属性の値が渡されない場合または NULL に設定されている場合、WF_LOCAL_ROLES でその属性に対応するフィールドの値は NULL に設定され、既存の値は削除されます。

- description
- preferredLanguage
- orclNLSTerritory
- mail
- FacsimileTelephoneNumber
- ExpirationDate

したがって、上書きモードを使用する場合は、NULL に設定する属性を除き、すべての属性に値を渡す必要があります。また、USER_NAME 属性は常に指定する必要があります。

注意： WF_LOCAL_ROLES 表の DISPLAY_NAME、NOTIFICATION_PREFERENCE、STATUS、ORIG_SYSTEM および ORIG_SYSTEM_ID の各列は、NULL に設定することはできません。したがって、これらの列では、対応する属性に値を渡さないと、上書きモードの場合でも既存の値が保持されます。

WF_LOCAL_ROLES 表の NAME 列も NULL に設定することはできませんが、USER_NAME 属性はこの API に必須なので省略できません。

WFSYNCH_OVERWRITE という特殊な属性を追加し、値を TRUE に指定することで、Propagate_Role() を上書きモードでコールすることもできます。上書きモードでは、いずれかの属性の値が渡されない場合または NULL に設定されている場合、WF_LOCAL_ROLES でその属性に対応するフィールドの値は NULL に設定され、既存の値は削除されます。したがって、上書きモードを使用する場合は、NULL に設定する属性を除き、すべての属性に値を渡す必要があります。また、WF_LOCAL_ROLES 表の NAME、DISPLAY_NAME、NOTIFICATION_PREFERENCE、STATUS、ORIG_SYSTEM および ORIG_SYSTEM_ID の各列は、NULL に設定することはできません。したがって、上書きモードを使用する場合は、これらの列に対応する属性には常に値を渡す必要があります。そうしないと、プロシージャは失敗し、更新は行われません。

元のシステム、元のシステム ID、失効日などの一部の値は、Propagate_Role() API のパラメータとして渡すことも、属性リスト・パラメータで属性として渡すこともできます。LDAP 統合を管理する Entity Manager には、Propagate_Role() から属性リストだけが送られ、プロシージャの他のパラメータは渡されないため、これらの値は属性リストにも渡されます。

- 元のシステムおよび元のシステム ID の値では、属性リストで属性として指定された値よりも、プロシージャにパラメータとして渡される値が優先されます。
- 同様に、失効日の値では、属性リストで属性として指定された値よりも、プロシージャにパラメータとして渡される値が優先されます。ただし、p_expiration_date パラメータが NULL の場合は、ExpirationDate 属性の値が指定されていればそれが使用されます。ExpirationDate 属性の値は次の形式で指定する必要があります。

```
to_char(<your date variable>, WF_ENGINE.Date_Format)
```

Oracle Workflow には他にも 2 つの特殊属性が用意されており、これらを使用してロール情報の変更方法を指定できます。

- DELETE: この属性は、ロールがワークフローに参加できないようにする場合に使用します。この属性を追加して TRUE の値を指定すると、WF_LOCAL_ROLES でロールの失効日として sysdate が設定され、ステータスは INACTIVE に設定されます。

注意： ただし、p_expiration_date パラメータにも値を指定した場合は、その値が DELETE 属性よりも優先されます。また、p_expiration_date パラメータが NULL でも、ExpirationDate 属性を指定した場合は、この属性が DELETE 属性よりも優先されます。このような場合、指定した失効日まで、ロールは有効でアクティブなままになります。

- UpdateOnly: この属性は、パフォーマンス改善のため、WF_LOCAL_ROLES にすでにレコードが存在するロールの情報を変更する場合に使用します。この属性を追加して TRUE の値を指定すると、Propagate_Role() API はレコードの挿入を行わず、レコードを直接更新しようとします。

  そのロールのレコードが存在しないために更新が失敗する場合、プロシージャはレコードを挿入します。ただし、最初の更新が失敗するとパフォーマンスが低下するため、WF_LOCAL_ROLES にロール・レコードが確実に存在している場合にのみ UpdateOnly 属性を使用してください。

注意： Oracle Applications では、元のシステムが PER_ROLE である Oracle Human Resources 担当者ロールを Propagate_Role() を使用して伝播するとき、この担当者が Oracle Applications ユーザーにリンクされている場合は、担当者レコードだけでなく、WF_LOCAL_ROLES 内の元のシステムが PER である対応するユーザー・レコードも更新されます。

引数（入力）

p_orig_system	ロール情報の提供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。
p_orig_system_id	このリポジトリ・システムのロールを識別する主キー。
p_attributes	ロールに関する情報が含まれている、属性の名前と値のペアのリスト。
p_start_date	ディレクトリ・サービスにおいて、ロールが有効になる日付。
p_expiration_date	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。

関連項目：

5-50 ページ「AddParameterToList」

Propagate_User_Role

構文

```
procedure Propagate_User_Role
    (p_user_orig_system in varchar2,
     p_user_orig_system_id in number,
     p_role_orig_system in varchar2,
     p_role_orig_system_id in number,
     p_start_date in date default null,
     p_expiration_date in date default null);
```

説明

アプリケーション表に保存されているユーザーとロールの関連付けの情報と、WF_LOCAL_USER_ROLES 表とを同期させます。

引数（入力）

p_user_orig_system ユーザー情報の提供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。

p_user_orig_system_id このリポジトリ・システムのユーザーを識別する主キー。

p_role_orig_system ロール情報の提供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。

p_role_orig_system_id このリポジトリ・システムのロールを識別する主キー。

p_start_date ディレクトリ・サービスにおいて、このユーザーとこのロールの関連付けが有効になる日付。

p_expiration_date ディレクトリ・サービスにおいて、このユーザーとこのロールの関連付けが有効でなくなる日付。

Workflow Preferences API

次の API をコールし、ユーザー設定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の値を取得します。

引数（入力）

p_user_name ユーザーの内部名。グローバル設定の値を取得するには、ユーザーに WF_DEFAULT と指定します。

p_preference_name 値を取得するユーザー設定の名前。次の設定名を取得できます。

LANGUAGE

TERRITORY

MAILTYPE

DMHOME

DATEFORMAT

4

通知システム API

この章では、Oracle Workflow 通知システムの API について説明します。API は、通知システムへのアクセスに使用できる 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 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 Workflow Builder 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 ページを使用して、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を検出すると、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がチェックされます。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（RESPOND 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。4-5 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。

- oracle.apps.wf.notification.send イベントを呼び出します。このイベントの処理時に、通知の実行者ロールが、MAILTEXT、MAILHTML、MAILHTM2、MAILATTH または SUMMARY の通知環境設定を持つ場合、通知メーラーは電子メール版の通知を作成し、実行者に送信します。『Oracle Workflow 管理者ガイド』の「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 ページ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クローズします。4-4 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。4-4 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。4-5 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。4-5 ページの「タイムアウト通知の処理」を参照してください。

> **注意：** Oracle Applications に組み込まれている Workflow で利用可能な Oracle Applications Framework 対応の「ワークリスト」ページを使用している場合は、「FND: 通知再割当モード」プロファイル・オプションを使用して、ユーザーが通知の転送（委任とも呼ばれます）または通知の譲渡、あるいはその両方によって通知の再割当てを行うことができるかどうかを判断できます。『Oracle Workflow 管理者ガイド』の「「FND: 通知再割当モード」プロファイル・オプションの設定」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 ページ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。2-12 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 ページは、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。2-12 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 ページは、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 ページ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。2-12 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じ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。『Oracle Workflow 開発者ガイド』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 ページを使用すると、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、通知を表す XML 文書に必要な構造を示しています。通知システムはこの構造を使用して、通知メーラーにメッセージを伝達します。次の表に、DTD の各要素のレベル、タグ名および説明を示します。

表 4-1

レベル	タグ	説明
1	<NOTIFICATIONGROUP maxcount="">	<NOTIFICATIONGROUP> タグは XML 構造の開始タグです。maxcount 属性は、指定できる <NOTIFICATION> タグの最大数を定義します。<NOTIFICATIONGROUP> タグに含まれる <NOTIFICATION> タグの数は、この数に達しなくてもかまいませんが、この数を超えることはできません。
2	<NOTIFICATION nid="" language="" territory="" codeset="" priority="" accesskey="" node="" item_type="" message_name="">	<NOTIFICATION> 要素は、1 つのメッセージ・エンティティを定義します。<NOTIFICATION> の構造は <NOTIFICATIONGROUP> 内に複数存在しますが、その最大数は maxcount で指定した値に制限されます。各 <NOTIFICATION> 要素は、通知システムによって送信される 1 つの通知を表し、一意の nid 属性（通知 ID）によって識別されます。ユーザーからの通知応答など、外部ソースから受信したメッセージの場合、通知 ID は 0 になります。 priority 属性は、他のメッセージと比較した、そのメッセージの相対的な優先度を示します。1 ～ 33 は高い優先度、34 ～ 66 は通常の優先度、67 ～ 99 は低い優先度を表します。 language と territory の値は、通知の受信者の言語設定と地域設定を表します。codeset 属性は、WF_LANGUAGES 表でその言語に関連付けられているコードセットを示します。codeset 属性の値は、Oracle データベースのコードセット表記に従う必要があります。この通知を送信する通知メーラーに対して Reset NLS 構成パラメータが選択されている場合、最終的な電子メールは、Oracle データベースのコードセットに相当する IANA（Internet Assigned Numbers Authority）コードセットにエンコードされます。 accesskey 属性と node 属性には、インバウンド応答メッセージの情報が格納されます。これらの属性は nid 属性とともに、応答を検証するために使用されます。 item_type 属性は、この通知を所有している Oracle Workflow 項目タイプの内部名を示します。message_name 属性は、その項目タイプ内で、この通知を表す内部名を示します。これら 2 つの属性は参考のために指定されるもので、Java ベースの通知メーラーで使用されることはありません。
3	<HEADER>	<The HEADER> 要素は、メッセージのエンベロープ情報を定義します。ここには、受信者の詳細、メッセージの送信元およびメッセージの件名が含まれます。

表 4-1（続き）

レベル	タグ	説明
4	<RECIPIENTLIST>	<RECIPIENTLIST> タグを使用すると、複数の受信者にメッセージを送信できます。リストの先頭の受信者は、主受信者として扱われます。他の受信者は、メッセージのコピーを受け取ります。主受信者の言語設定と書式設定に従って、リスト内のすべての受信者が同じ電子メールを受け取ります。
5	<RECIPIENT name="" type="">	<RECIPIENT> タグは、メッセージの受信者を定義します。<RECIPIENT> の構造は <RECIPIENTLIST> 内に複数存在します。各 <RECIPIENT> は、受信者ロールの内部名を表す name 属性によって識別されます。 type 属性は、受信者のコピー・タイプを示します。この属性の有効な値は、to、cc および bcc です。type 属性が指定されていない場合、この受信者のコピー・タイプは to とみなされます。
6	<NAME> </NAME>	<NAME> タグは、受信者の表示名を定義します。
6	<ADDRESS> </ADDRESS>	<ADDRESS> タグは、受信者の電子メール・アドレスを定義します。
5	</RECIPIENT>	このタグは、<RECIPIENT> 要素の終わりを示します。
4	</RECIPIENTLIST>	このタグは、<RECIPIENTLIST> 要素の終わりを示します。
4	<FROM>	<FROM> タグは、メッセージの送信者を示します。アウトバウンド通知の場合、#FROM_ROLE メッセージ属性を使用して「From Role」を設定できます。また、「From Role」は、再割当てされた通知であれば再割当てを実行したロールに設定され、詳細情報を要求する通知であれば要求元ロールに設定され、詳細情報の要求に応答する通知であれば応答側ロールに設定されます。 インバウンド通知の場合、この情報は、着信する電子メール・メッセージの「送信元」アドレスによって決定されます。
5	<NAME> </NAME>	<NAME> タグは、送信者の表示名を定義します。
5	<ADDRESS> </ADDRESS>	<ADDRESS> タグは、送信者の電子メール・アドレスを定義します。
4	</FROM>	このタグは、<FROM> 要素の終わりを示します。
5	<SUBJECT> </SUBJECT>	<SUBJECT> 要素は、通知の件名を保持します。
3	</HEADER>	このタグは、<HEADER> 要素の終わりを示します。

表 4-1（続き）

レベル	タグ	説明
3	<CONTENT content-type="">	<CONTENT> 要素は、通知メッセージの内容を保持します。<CONTENT> 要素には、1 つ以上の <BODYPART> 要素が含まれます。content-type 属性は、<CONTENT> 要素内の内容に対して有効な MIME タイプ定義を保持します。content-type 属性の有効な値は、multipart/mixed、text/plain、text/html などです。<CONTENT> タグ内の最初の <BODYPART> 要素は、メッセージの主要な内容として扱われ、multipart/* メッセージ構造内で最初のコンポーネントになります。
4	<BODYPART content-type="">	<BODYPART> タグは、最終的なメッセージの MIME コンポーネントを表します。この要素には、1 つの <MESSAGE> タグが含まれます。1 つ以上の <RESOURCE> タグが含まれることもあります。<RESOURCE> タグを実装する場合は、<RESOURCE> 要素と <MESSAGE> 要素の関係を説明するために、<BODYPART> タグに content-type 属性を定義する必要があります。この content-type 属性の有効な値は、`multipart/related` のみです。 最初の <BODYPART> 要素は、メッセージの主要な内容として扱われます。この内容は、text/plain または text/html になります。他の <BODYPART> 要素には、通知メッセージ定義と受信者の通知環境設定に応じて、必要な添付ファイルが含まれます。添付ファイルとしては、HTML 形式の通知、通知の詳細リンク、「内容の添付」がオンになっているメッセージ属性などがあります。 インバウンド・メッセージの場合、<BODYPART> 要素には、メッセージと、必要なすべての添付ファイルが含まれます。
5	<MESSAGE content-type="" content-transfer-encoding="" content-disposition="" src="">	content-type 属性は、<MESSAGE> 要素のメディア・タイプ定義を保持します。この content-type 属性の有効な値は、`text/plain`、`text/html`、`multipart/mixed` または `multipart/related` です。 content-transfer-encoding 属性は、`text/plain` コンテンツまたは `text/html` コンテンツのエンコーディングをより詳細に指定する、オプションの属性です。 content-disposition 属性は、コンポーネントが添付ファイルであることを示します。 src 属性は、通知 XML 文書の生成時に <MESSAGE> 要素の内容がまだ利用可能でない場合に、定義することができます。src 属性の値には、最終的な電子メール・メッセージの作成時に内容を取得するための URL を指定する必要があります。
-	<![CDATA[ ]]>	この要素は、メッセージ内容をそのまま保持します。
5	</MESSAGE>	このタグは、<MESSAGE> 要素の終わりを示します。

表 4-1（続き）

レベル	タグ	説明
5	<RESOURCE content-type="" content-transfer-encoding="" content-disposition="" content-id="" src="">	content-type 属性は、<RESOURCE> 要素のメディア・タイプ定義を保持します。この属性の有効な値は、media-type/subtype です。 content-transfer-encoding 属性は、text/plain コンテンツまたは text/html コンテンツのエンコーディングをより詳細に指定する、オプションの属性です。 content-disposition 属性は、コンポーネントが添付ファイルであることを示します。 content-id 属性は、コンポーネントの一意のコンテンツ識別子を保持します。この識別子は、<MESSAGE> 要素の内容の中で参照されます。 src 属性は、通知 XML 文書の生成時に <RESOURCE> 要素の内容がまだ利用可能でない場合に、定義することができます。src 属性の値には、最終的な電子メール・メッセージの作成時に内容を取得するための URL を指定する必要があります。
-	<![CDATA[ ]]>	この要素は、<RESOURCE> 要素の内容を保持します。
5	</RESOURCE>	このタグは、<RESOURCE> 要素の終わりを示します。
4	</BODYPART>	このタグは、<BODYPART> 要素の終わりを示します。
3	</CONTENT>	このタグは、<CONTENT> 要素の終わりを示します。
3	<RESPONSE>	<RESPONSE> タグは、インバウンド通知の場合にのみ実装されます。アウトバウンド通知の仕様には含まれていません。 <RESPONSE> 要素には、1 つ以上の <ATTRIBUTE> 要素が含まれます。<ATTRIBUTE> 要素は、着信する電子メール・メッセージに含まれている応答値を保持します。通知に関連付けられている応答属性ごとに、<ATTRIBUTE> タグが 1 つ必要です。ただし、RESULT メッセージ属性だけは必須です。他の応答属性はオプションです。応答属性の値が指定されていない場合、Oracle Workflow では、メッセージ属性のデフォルト値が使用されます。

表 4-1（続き）

レベル	タグ	説明
4	<ATTRIBUTE name="" type="" format="">	<ATTRIBUTE> タグは、着信する電子メール・メッセージに含まれている、特定の応答属性の応答値を保持します。<ATTRIBUTE> の構造は <RESPONSE> 内に複数存在します。 この要素の name 属性は、応答属性の内部名を示します。 この要素の type 属性は、応答属性の Oracle Workflow データ型を示します。有効な値は、TEXT、NUMBER、DATE、DOCUMENT または LOOKUP です。 この要素の format 属性は、応答属性の書式文字列を保持します。LOOKUP 型の応答属性の場合、name 属性の値に応じて選択肢タイプ・コードを識別するために format 属性が使用されます。他のデータ型の場合、format 属性は使用されません。
-	<![CDATA]]>	この要素は、属性に割り当てる応答情報を保持します。
4	</ATTRIBUTE>	このタグは、<ATTRIBUTE> 要素の終わりを示します。
3	</RESPONSE>	このタグは、<RESPONSE> 要素の終わりを示します。
2	</NOTIFICATION>	このタグは、<NOTIFICATION> 要素の終わりを示します。
1	</NOTIFICATIONGROUP>	このタグは、<NOTIFICATIONGROUP> 要素の終わりを示します。

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。2-4 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。

注意： 通知の送信、クローズ、取消しまたは再割当てが行われたときや、ユーザーが通知に応答したときに、通知システムはビジネス・イベントを呼び出します。これらのイベントの一部については、Oracle Workflow に事前定義済のサブスクリプションは含まれていないため、これらのイベントが発生したときに独自の処理を実行するために、独自のサブスクリプションを定義することもできます。『Oracle Workflow 開発者ガイド』の「通知イベント」および『Oracle Workflow 開発者ガイド』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。

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。4-15 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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。
role	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。
msg_type または messageType	メッセージに関連付けられている項目タイプ。
msg_name または messageName	メッセージの内部名。
due_date または dueDate	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。
callback	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。
context	コールバック関数に渡されるコンテキスト情報。
send_comment または sendComment	メッセージに表示するコメント。
priority	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、標準の仕様に従ってください。

コールバック関数を介してイベント・タイプの属性を処理する必要がない場合は、プロシージャで次の標準 API を使用する必要があります。

```
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);
```

コールバック関数でイベント・タイプの属性を処理する必要がある場合は、イベント値に対応する追加の引数を持つ別の実装でプロシージャ名をオーバーロードできます。この場合

は、下位互換性を保つために、元の実装も残しておいてください。ただし、プロシージャのオーバーロードは、イベント属性を処理する必要がある場合にのみ行うことをお薦めします。

イベント値に対応するプロシージャの実装では、次の標準 API を使用する必要があります。

```
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,
     event_value in out nocopy wf_event_t);
```

保守しやすいように、event_value 引数を持たないプロシージャを定義し、そこからこの引数を持つプロシージャをコールするようにすると、単一のコードを保守するだけですみます。次の例は、このようなコールの実装方法を示しています。

```
procedure your_callback
    (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)

is
  event_value wf_event_t;

begin
  your_package.your_callback(command, context, attr_name,
                          attr_type, text_value,
                          number_value, date_value,
                          event_value);

exception
  when others then
    Wf_Core.Context('your_package', 'your_callback',
                command, context, attr_name, attr_type,
                ':'||text_value||':'||to_char(number_value)
                ||':'||to_char(date_value)||':');
    raise;

end your_callback;
```

引数（入力）

command 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、項目タイプのセレクタ / 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。

context SEND() または SendGroup() に渡されるコンテキスト。書式は *<itemtype>*:*<itemkey>*:*<activityid>* です。

attr_name 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。

attr_type コマンドが SET または GET の場合、属性タイプ。

text_value 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。

number_value 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。

date_value 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。

event_value コマンドが SET の場合、イベント属性値。コマンドが GET の場合、戻されるイベント属性値。イベント属性を処理する別の実装でプロシージャ名をオーバーロードする場合にのみ必須です。

注意： 引数 `text_value`、`number_value`、`date_value` および `event_value`（使用する場合）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 つのみを使用する必要があります。

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);
```

例 4

イベント・タイプの SEND 属性の場合は、event_value 引数を持つ実装をコールしてください。

```
your_callback('GET', context, 'RECEIVE_EVENT', 'EVENT', textval, numval,
dateval, eventval);
```

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戻します。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。4-15 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。『Oracle Workflow 開発者ガイド』の「投票アクティビティ」を参照してください。

引数（入力）

role	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。
msg_type	メッセージに関連付けられている項目タイプ。
msg_name	メッセージの内部名。
due_date	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。
callback	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。
context	コールバック関数に渡されるコンテキスト情報。
send_comment	メッセージに表示するコメント。
priority	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。

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）

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。2-5 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。
new_role または newRole	メモが再割当てされる個人のロール名。
forward_comment または comment	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）

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。2-5 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。
new_role または newRole	メモが転送される個人のロール名。
forward_comment または comment	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「取消」の電子メールが送信されます。

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。2-5 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。
cancel_comment または comment	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「取消」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。『Oracle Workflow 開発者ガイド』の「投票アクティビティ」を参照してください。

引数（入力）

gid 通知グループ ID。

cancel_comment 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 ページまたは電子メール・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 4-2

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

また、Respond() プロシージャは NtfSignRequirementsMet() を呼び出して、通知の電子署名ポリシーで指定されている署名要件を応答が満たしているかどうかを調べます。要件を満たしていない場合は、Respond() でエラーが発生します。『Oracle Workflow 開発者ガイド』の「#WF_SIG_POLICY 属性」および 4-30 ページの「NtfSignRequirementsMet」を参照してください。

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。2-5 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)
return varchar2;
```

Java 構文

```
public static String responder
    (WFContext wCtx,
     BigDecimal nid)
```

説明

この関数は、クローズ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クローズした場合、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クローズした場合、戻り値は電子メール・アドレスです。4-27 ページの「Respond」を参照してください。

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。2-5 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。

NtfSignRequirementsMet

PL/SQL 構文

```
function NtfSignRequirementsMet
(nid in number)
return boolean;
```

説明

通知に対する応答が、通知の電子署名ポリシーで指定されている署名要件を満たしている場合は、TRUE を返します。『Oracle Workflow 開発者ガイド』の「#WF_SIG_POLICY 属性」を参照してください。

- 通知で署名ポリシーが使用されていて、ユーザーの応答を検証するために電子署名が必要な場合、その要件を満たすには、応答に署名する権限を持つユーザーによる有効な署名を送信する必要があります。
- 署名を必要としないデフォルト・ポリシーが通知で使用されている場合や、通知に署名ポリシーが定義されていない場合は、署名のない応答でも要件を満たします。

ただし、通知の署名ポリシーで電子署名が必要な場合は、有効な署名を送信しないと、その応答は要件を満たしません。この場合、NtfSignRequirementsMet() は FALSE を返します。

引数（入力）

nid 通知 ID。

関連項目：

4-27 ページ「Respond」

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 WFTwoDArray 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。『Oracle Workflow 開発者ガイド』の「投票アクティビティ」を参照してください。

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

gid 通知グループ ID。

ResultCode 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 OPEN の場合は 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。『Oracle Workflow 開発者ガイド』の「投票アクティビティ」を参照してください。

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

gid 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クローズします。

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。2-5 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。
responder	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）

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。2-5 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。

aname 属性名。

avalue 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）

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。2-5 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。
aname	属性名。
avalue	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、WFTest.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 WFTwoDArray getAttrInfo
    (WFContext wCtx,
     BigDecimal nid,
     cdString 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、それを戻します。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）

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。2-5 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。

aname 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、WFTest.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 WFTwoDArray 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）

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。2-5 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。

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;
```

Java 構文

```
public static String getText
    (WFContext wCtx,
     String someText,
     BigDecimal nid,
     String dispType)
```

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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。
some_text または someText	置き換えられるテキスト。
nid	トークン値に使用する通知の通知 ID。

disptype または dispType

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。

- wf_notification.doc_text。text/plain を戻します。
- wf_notification.doc_html。text/html を戻します。
- wf_notification.doc_attach。NULL を戻します。

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）

some_text	置き換えられるテキスト。
nid	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）

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。2-5 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。

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、WFTest.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）

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。2-5 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。

aname メッセージ属性の名前。

disptype 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。

- wf_notification.doc_text。text/plain を戻します。
- wf_notification.doc_html。text/html を戻します。
- wf_notification.doc_attach。NULL を戻します。

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）

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。2-5 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。

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戻します。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 の文字を戻します。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）

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。2-5 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。

disptype 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。

- wf_notification.doc_text。text/plain を戻します。
- wf_notification.doc_html。text/html を戻します。
- wf_notification.doc_attach。NULL を戻します。

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戻します。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、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 を戻し、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

項目タイプのセレクタ / 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、またはセレクタ / 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）

nid 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、通知を送信する通知メーラ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。2-5 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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

username ロールの内部名。

getNotifications

Java 構文

```
public static WFTwoDArray 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）

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。2-5 ページの「Oracle Workflow のコンテキスト」を参照してください。

itemType 項目タイプの内部名。

itemKey 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 WFTwoDArray 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。2-5 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。

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,
 msg_string 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）

clob_loc 文字列の追加先の CLOB。

msg_string 文字データの文字列。

関連項目：

『Oracle Workflow 開発者ガイド』の「文書属性の定義」

『Oracle Workflow 開発者ガイド』の「PL/SQL CLOB 文書」

Denormalize_Notification

PL/SQL 構文

```
procedure Denormalize_Notification
    (nid in number,
     username in varchar2 default null,
     langcode in varchar2 default null);
```

説明

WF_NOTIFICATIONS 表で、一部の通知フィールド（通知の件名など）の正規化されていない値を格納します。ワークフロー・プロセスの外部で通知システムを使用して通知を送信する場合は、通知の各属性に値を設定してから Denormalize_Notification() をコールして、正規化されていないフィールドに値を格納する必要があります。

Denormalize_Notification() は、配信する通知に使用する言語と現行セッションの言語が一致するかどうかを確認し、一致する場合のみ、この設定に従って、正規化されていない情報を格納します。通知に使用する言語は、様々な方法で指定できます。

- この API をコールするときにロール名を指定すると、そのロールの言語設定に従って通知の言語が決定されます。
- ロール名を指定しない場合は、目的の言語に対応する言語コードを指定できます。

> **注意：** ロール名と言語コードを両方とも指定すると、言語コードは無視され、ロール名に従って通知の言語が決定されます。

- ロール名も言語コードも指定しない場合は、通知のデフォルトの言語として、通知の受信者ロールの言語設定が使用されます。

通知の言語と現行セッションの言語が一致しない場合、プロシージャは正規化されていない情報を格納しません。この場合は、通知の受信者が通知を表示するために使用するインタフェースで、言語の確認と非正規化を実行する必要があります。ユーザーが「ワークリスト」Web ページを通して通知にアクセスする場合は、Oracle Workflow の通知の「ワークリスト」がこれらの作業を実行します。

引数（入力）

nid	通知 ID。
username	通知の言語を決定するロールの内部名（オプション）。
langcode	ロール名が指定されていない場合に通知の言語を決定する言語コード（オプション）。

通知メーラー・ユーティリティ API

通知メーラー・ユーティリティ API を使用すると、バイナリ・ラージ・オブジェクト（BLOB）のデータを Base64 にエンコードできます。この API は、WF_MAIL_UTIL という PL/SQL パッケージに定義されています。

注意： このパッケージを使用できるのは、Oracle9*i* Database 以降のみです。Oracle8*i* Database では、Base64 エンコーディングはサポートされていません。

EncodeBLOB

PL/SQL 構文

```
procedure EncodeBLOB
    (pIDoc in blob,
     pODoc in out nocopy clob);
```

説明

指定された BLOB を Base64 にエンコードし、エンコードされたデータをキャラクタ・ラージ・オブジェクト（CLOB）として戻します。このプロシージャを使用すると、通知メッセージに含める PL/SQL CLOB 文書に BLOB を格納できます。

注意： この API を使用できるのは、Oracle9*i* Database 以降のみです。Oracle8*i* Database では、Base64 エンコーディングはサポートされていません。

引数（入力）

pIDoc エンコードする BLOB。

pODoc エンコードされたデータの格納先となる CLOB。

関連項目：

『Oracle Workflow 開発者ガイド』の「PL/SQL 文書の標準 API」

5

ビジネス・イベント・システム API

この章では、Oracle Workflow ビジネス・イベント・システムの API について説明します。API は、ビジネス・イベント・システムへのアクセスに使用できるデータ型および PL/SQL の関数とプロシージャで構成されています。

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する、イベント情報の送信

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：

『Oracle Workflow 開発者ガイド』の「ビジネス・イベントの管理」

『Oracle Workflow 開発者ガイド』の「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。

- 5-4 ページ「エージェント構造」: WF_AGENT_T
- 5-6 ページ「パラメータ構造」: WF_PARAMETER_T
- 5-8 ページ「パラメータ・リスト構造」: WF_PARAMETER_LIST_T
- 5-8 ページ「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 に組み込まれている 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：

『Oracle9*i* データベース概要』

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 5-1

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。

- 5-4 ページ「getName」
- 5-4 ページ「getSystem」
- 5-5 ページ「setName」
- 5-5 ページ「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：

『Oracle Workflow 開発者ガイド』の「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 5-2

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。

- 5-6 ページ「getName」
- 5-7 ページ「getValue」
- 5-7 ページ「setName」
- 5-7 ページ「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。Oracle Workflow でビジネス・イベント・システムに対して用意されている標準キューの多くは、ペイロード・タイプとして WF_EVENT_T を使用します。

> **注意 :** カスタム・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。『Oracle Workflow 管理者ガイド』の「キューの設定」および『Oracle Workflow 開発者ガイド』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 5-3

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。

表 5-3（続き）

属性名	データ型	説明
SEND_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。
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。プロセス・インスタンスの項目キーに使用できるのは、シングル・バイト・キャラクタのみであることに注意してください。マルチバイトの値を含めることはできません。
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。

表 5-3（続き）

属性名	データ型	説明
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。

- 5-12 ページ「Initialize」
- 5-12 ページ「getPriority」
- 5-13 ページ「getSendDate」
- 5-13 ページ「getReceiveDate」
- 5-13 ページ「getCorrelationID」
- 5-14 ページ「getParameterList」
- 5-14 ページ「getEventName」
- 5-14 ページ「getEventKey」
- 5-15 ページ「getEventData」
- 5-15 ページ「getFromAgent」
- 5-15 ページ「getToAgent」
- 5-16 ページ「getErrorSubscription」
- 5-16 ページ「getErrorMessage」
- 5-16 ページ「getErrorStack」
- 5-17 ページ「setPriority」
- 5-17 ページ「setSendDate」
- 5-18 ページ「setReceiveDate」
- 5-18 ページ「setCorrelationID」
- 5-19 ページ「setParameterList」
- 5-19 ページ「setEventName」

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。5-23 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。5-24 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）

pName EVENT_NAME 属性の値。

pKey EVENT_KEY 属性の値。

pData 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）

pOutAgent	FROM_AGENT 属性の値。
pToAgent	TO_AGENT 属性の値。
pPriority	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 wfevtenq.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。『Oracle Workflow 管理者ガイド』の「wfevtenq.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

Oracle8*i* Database 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ある必要があります。

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。『Oracle Workflow 管理者ガイド』の「WF_EVENT_OMB_QH キュー・ハンドラの設定」および『Oracle Workflow 開発者ガイド』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。

次の一覧は、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 5-4

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]

表 5-4（続き）

WF_EVENT_T	OMBAQ_TEXT_MSG
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 = FROMAGENTSYSTEM]
WF_EVENT_T.TO_AGENT.NAME	ombaq_properties(3).str_value [name = TOAGENTNAME]
WF_EVENT_T.TO_AGENT.SYSTEM	ombaq_properties(4).str_value [name = TOAGENTSYSTEM]
WF_EVENT_T.ERROR_SUBSCRIPTION	ombaq_properties(11).str_value [name = ERRORSUBSCRIPTION]
WF_EVENT_T.ERROR_MESSAGE	ombaq_properties(12).str_value [name = ERRORMESSAGE1]
WF_EVENT_T.ERROR_MESSAGE	ombaq_properties(13).str_value [name = ERRORMESSAGE2]
WF_EVENT_T.ERROR_STACK	ombaq_properties(14).str_value [name = ERRORSTACK1]
WF_EVENT_T.ERROR_STACK	ombaq_properties(15).str_value [name = ERRORSTACK2]
WF_EVENT_T.PARAMETER_LIST	ombaq_properties(16).str_value [name = <*first_parameter_name*>]
...	...
WF_EVENT_T.PARAMETER_LIST	ombaq_properties(115).str_value [name = <*hundredth_parameter_name*>]

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, FROMAGENTSYSTEM, TOAGENTNAME,
TOAGENTSYSTEM, ERRORSUBSCRIPTION, ERRORMESSAGE1, ERRORMESSAGE2,
ERRORSTACK1, ERRORSTACK2
```

注意： Oracle Message Broker および OMBAQ_TEXT_MSG データ型は、Oracle9*i* Database 以降では使用されていません。それらの Oracle Database 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 Administrator's Guide』の「Oracle AQ Driver ADTs」

WF_EVENT_T および SYS.AQ$_JMS_TEXT_MESSAGE 間のマッピング

Java Message Service（JMS）は、Sun 社、オラクル社、IBM 社およびその他のベンダーによって定められたメッセージングの標準規格です。JMS は、JMS クライアントが企業メッセージング製品の機能にアクセスする方法を定義する、一連のインターフェイスと関連するセマンティクスで構成されています。

Oracle Java Message Service には、JMS 規格に基づく Oracle Advanced Queuing（AQ）用の Java API が用意されています。Oracle JMS では、標準の JMS インタフェースがサポートされ、さらに規格には含まれていない、AQ 管理操作や他の AQ 機能をサポートする拡張機能も提供されます。JMS テキスト・メッセージを AQ キューに格納するための抽象データ型は、SYS.AQ$_JMS_TEXT_MESSAGE と呼ばれます。

Oracle Workflow では、ビジネス・イベント・システムを通して JMS テキスト・メッセージの通信をサポートするために、WF_EVENT_OJMSTEXT_QH というキュー・ハンドラが用意されています。このキュー・ハンドラは、Workflow 標準の WF_EVENT_T メッセージ構造と SYS.AQ$_JMS_TEXT_MESSAGE とを変換します。Oracle Workflow には、JMS テキスト・メッセージに使用できる標準の受信キューと送信キューも用意されています。これらのキューは、それぞれ WF_JMS_IN および WF_JMS_OUT と呼ばれ、WF_EVENT_OJMSTEXT_QH キュー・ハンドラを使用します。『Oracle Workflow 開発者ガイド』の「エージェント」を参照してください。

SYS.AQ$_JMS_TEXT_MESSAGE データ型は、次の属性で構成されます。

- HEADER: SYS.AQ$_JMS_HEADER データ型のヘッダー・プロパティ
- TEXT_LEN: メッセージ・ペイロードのサイズ（自動的に設定）
- TEXT_VC: VARCHAR2 形式のメッセージ・ペイロード（ペイロードが 4000 バイト以下の場合）
- TEXT_LOB: CLOB 形式のメッセージ・ペイロード（ペイロードが 4000 バイトより大きい場合）

SYS.AQ$_JMS_HEADER データ型は、次の属性で構成されます。

- REPLYTO: メッセージの送信時にクライアントによって指定された宛先
- TYPE: メッセージのタイプ
- USERID: メッセージの送信元ユーザーの識別情報
- APPID: メッセージの送信元アプリケーションの識別情報
- GROUPID: クライアントによって設定された、メッセージが属しているメッセージ・グループの識別情報
- GROUPSEQ: グループ内でのメッセージの連番
- PROPERTIES: SYS.AQ$_JMS_USERPROPARRAY データ型の追加のメッセージ・プロパティ

SYS.AQ$_JMS_USERPROPARRAY データ型は名前付き可変配列で、その最大サイズは 100 です。その要素のデータ型は、SYS.AQ$_JMS_USERPROPERTY と呼ばれる ADT です。

次の一覧は、WF_EVENT_T メッセージ構造の属性と、SYS.AQ$_JMS_TEXT_MESSAGE 構造内の属性の対応表です。

表 5-5

WF_EVENT_T	SYS.AQ$_JMS_TEXT_MESSAGE
WF_EVENT_T.PRIORITY	SYS.AQ$_JMS_USERPROPARRAY
WF_EVENT_T.SEND_DATE	SYS.AQ$_JMS_USERPROPARRAY
WF_EVENT_T.RECEIVE_DATE	SYS.AQ$_JMS_USERPROPARRAY
WF_EVENT_T.CORRELATION_ID	SYS.AQ$_JMS_USERPROPARRAY
WF_EVENT_T.EVENT_NAME	SYS.AQ$_JMS_USERPROPARRAY
WF_EVENT_T.EVENT_KEY	SYS.AQ$_JMS_USERPROPARRAY
WF_EVENT_T.EVENT_DATA	TEXT_VC または TEXT_LOB
WF_EVENT_T.PARAMETER_LIST	SYS.AQ$_JMS_HEADER.REPLYTO
WF_EVENT_T.PARAMETER_LIST	SYS.AQ$_JMS_HEADER.TYPE
WF_EVENT_T.PARAMETER_LIST	SYS.AQ$_JMS_HEADER.USERID
WF_EVENT_T.PARAMETER_LIST	SYS.AQ$_JMS_HEADER.APPID
WF_EVENT_T.PARAMETER_LIST	SYS.AQ$_JMS_HEADER.GROUPID
WF_EVENT_T.PARAMETER_LIST	SYS.AQ$_JMS_HEADER.GROUPSEQ
WF_EVENT_T.PARAMETER_LIST（JMS ヘッダー・プロパティ以外の任意のパラメータ）	SYS.AQ$_JMS_USERPROPARRAY
WF_EVENT_T.FROM_AGENT	SYS.AQ$_JMS_USERPROPARRAY
WF_EVENT_T.TO_AGENT	SYS.AQ$_JMS_USERPROPARRAY
WF_EVENT_T.ERROR_SUBSCRIPTION	SYS.AQ$_JMS_USERPROPARRAY
WF_EVENT_T.ERROR_MESSAGE	SYS.AQ$_JMS_USERPROPARRAY
WF_EVENT_T.ERROR_STACK	SYS.AQ$_JMS_USERPROPARRAY

関連項目：

『Oracle9*i* 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「Oracle JMS を使用した AQ へのアクセス」または『Oracle Streams アドバンスト・キューイング・ユーザーズ・ガイドおよびリファレンス』の「Oracle Streams AQ にアクセスするための Oracle Java Message Service（OJMS）の使用」

『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の「パッケージ oracle.jms」

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。

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。『Oracle Workflow 開発者ガイド』の「イベント」および『Oracle Workflow 開発者ガイド』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。

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）

p_event_name イベントの内部名。

p_event_key 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。

p_event_data 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。『Oracle Workflow 開発者ガイド』の「イベント」および『Oracle Workflow 開発者ガイド』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。

p_parameters 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）。

p_send_date 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：

『Oracle Workflow 開発者ガイド』の「Any イベント」

『Oracle Workflow 開発者ガイド』の「Unexpected イベント」

Raise3

PL/SQL 構文

```
procedure Raise3
      (p_event_name in varchar2,
       p_event_key in varchar2,
       p_event_data in clob default NULL,
       p_parameter_list in out nocopy wf_parameter_list_t,
       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、そのイベントのパラメータ・リストを返します。Raise3() は Raise() プロシージャと同じ処理を実行しますが、イベント・サブスクリプションの処理が完了した後に Raise3() はイベントのパラメータ・リストをコール元のアプリケーションに返します。5-34 ページの「Raise」を参照してください。

Raise3() 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。イベントが遅延されていない場合は、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。

イベントのサブスクリプション処理が完了すると、Raise3() はイベントのパラメータ・リストを返します。このリストには、サブスクリプションのルール関数によってパラメータに加えられた変更内容も含まれています。この方法により、イベントを呼び出したアプリケーションにイベント・サブスクリプションからパラメータを戻すことができます。

注意： Raise3() 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、Raise3() 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）

p_event_name イベントの内部名。

p_event_key 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。

p_event_data 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。『Oracle Workflow 開発者ガイド』の「イベント」および『Oracle Workflow 開発者ガイド』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。

p_parameter_list 追加パラメータの名前と値のペアのリスト。

p_send_date 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）。

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アウトバウンド・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、『Oracle9*i* 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」または『Oracle Streams アドバンスト・キューイング・ユーザーズ・ガイドおよびリファレンス』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、『Oracle9*i* 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」または『Oracle Streams アドバンスト・キューイング・ユーザーズ・ガイドおよびリファレンス』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。5-4 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）

p_agent_guid　　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

アウトバウンド・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、アウトバウンド・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）

p_event イベント・メッセージ。

p_out_agent_override アウトバウンド・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,
      p_wait in binary_integer default dbms_aq.no_wait,
      p_correlation in varchar2 default null,
      p_deq_condition in varchar2 default null);
```

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。ただし、追加のメッセージを待機するためにキュー上でブロックするように待機時間を指定した場合を除きます。

アプリケーション・コード内から Listen() をコールしないでください。このプロシージャを直接コールする場合は、SQL*Plus からこのプロシージャを実行します。別の方法として、Oracle Workflow のバージョンに応じて Oracle Applications Manager、Oracle Enterprise Manager または他の方法を使用して、インバウンド・エージェントのリスナーをスケジュールできます。『Oracle Workflow 管理者ガイド』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。

イベント名から成る AQ 相関 ID を指定することで、Listen() プロシージャで処理するイベント・メッセージを制限することもできます。このとき、イベント名の一部分のみを指定し、末尾にパーセント記号（%）をワイルドカード文字として付加することもできます。また、Oracle9*i* Database 以降では、メッセージのプロパティまたは内容を参照するデキュー条件を指定することで、Listen() プロシージャで処理するイベント・メッセージを制限することもできます。ただし、両方のパラメータを同時に指定することはできません。一方を指定する場合は、他方を NULL にする必要があります。

引数（入力）

p_agent_name インバウンド・エージェントの名前。

p_wait メッセージを待機するためにエージェントのキュー上でエージェント・リスナーをブロックしておく秒数（オプション）。デフォルトでは、キューに入っているメッセージがすべてデキューされると、エージェント・リスナーは待機せずに終了します

p_correlation エージェント・リスナーで処理するイベント・メッセージを指定するための AQ 相関 ID（オプション）。ビジネス・イベント・システムでは、イベント・メッセージを表す AQ 相関 ID は、通常はイベント名で指定されます。イベント名の一部分のみを指定し、末尾にパーセント記号（%）をワイルドカード文字として付加することもできます。このパラメータに AQ 相関 ID を指定すると、エージェント・リスナーは指定されたイベントのインスタンスであるメッセージのみをリスニングするようになります。たとえば、「`oracle.apps.wf.notification%`」を指定すると、この値で始まる名前を持つ通知に関連するすべてのイベントをリスニングできます。この相関 ID のデフォルト値は NULL です。この場合は、任意のイベントのインスタンスであるメッセージがエージェント・リスナーで処理されます。次のパラメータにデキュー条件を指定する場合、このパラメータは NULL にする必要があります。『Oracle9*i* 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「デキューの方法」または『Oracle Streams アドバンスト・キューイング・ユーザーズ・ガイドおよびリファレンス』を参照してください。

注意： AQ 相関 ID は、WF_EVENT_T イベント・メッセージ構造内の相関 ID とは異なります。

---

p_deq_condition

エージェント・リスナーで処理するイベント・メッセージを指定するためのデキュー条件（オプション）。デキュー条件は、SQL 問合せの WHERE 句に似た構文を持つ式です。デキュー条件は、メッセージのプロパティやメッセージの内容を表す属性によって表現されます。キュー内のメッセージがこの条件に対して評価されるため、エージェント・リスナーはこの条件を満たすメッセージのみをリスニングするようになります。デフォルト値は NULL です。この場合、エージェント・リスナーで処理するメッセージに制限はありません。前のパラメータに AQ 相関 ID を指定する場合、このパラメータは NULL にする必要があります。『Oracle9*i* 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「デキューの方法」または『Oracle Streams アドバンスト・キューイング・ユーザーズ・ガイドおよびリファレンス』を参照してください。

関連項目：

『Oracle Workflow 開発者ガイド』の「Any イベント」

『Oracle Workflow 開発者ガイド』の「Unexpected イベント」

『Oracle Workflow 管理者ガイド』の「`wfagtlst.sql`」

『Oracle Workflow 開発者ガイド』の「キュー・ハンドラの標準 API」

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）

p_event イベント・メッセージ。

p_type 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）

p_mode　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：

『Oracle Workflow 開発者ガイド』の「遅延サブスクリプション処理」

5-34 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）

p_name パラメータ名。

p_value パラメータ値。

p_parameterlist 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）

p_name パラメータ名。

p_value パラメータ値。

p_parameterlist 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）

p_name パラメータ名。

p_parameterlist 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）

p_position	パラメータ・リスト内のパラメータの位置を表す索引。
p_parameterlist	パラメータ・リスト。

SetMaxNestedRaise

PL/SQL 構文

```
procedure SetMaxNestedRaise
     (maxcount in number default 100);
```

説明

ネストされた呼出しの最大数を設定します。指定した値まで実行できます。ネストされた呼出しは、あるイベントが呼び出されたときに、そのイベントのローカル・サブスクリプションが実行され、それによって別のイベントが呼び出されると発生します。デフォルト値は 100 です。

引数（入力）

max_count　　実行できるネストされた呼出しの最大数。

GetMaxNestedRaise

PL/SQL 構文

```
function GetMaxNestedRaise
     return number;
```

説明

現在実行できるネストされた呼出しの最大数を返します。ネストされた呼出しは、あるイベントが呼び出されたときに、そのイベントのローカル・サブスクリプションが実行され、それによって別のイベントが呼び出されると発生します。

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。

- 5-57 ページ「Default_Rule()」
- 5-59 ページ「Log」
- 5-60 ページ「Error」
- 5-61 ページ「Warning」
- 5-62 ページ「Success」
- 5-63 ページ「Workflow_Protocol」
- 5-64 ページ「Error_Rule」
- 5-65 ページ「SetParametersIntoParameterList」

関連項目：

『Oracle Workflow 開発者ガイド』の「イベント・サブスクリプション」

『Oracle Workflow 開発者ガイド』の「イベント・サブスクリプションのルール関数の標準 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相関 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。『Oracle Workflow 開発者ガイド』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）

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。5-22 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・メッセージ名である必要があります。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一意識別子。

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。5-22 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・メッセージ名である必要があります。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）

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。

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される WF_STANDARD.Noop プロシージャに似ています。

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。

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インバウンド・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・メッセージをインバウンド・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アウトバウンド・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・メッセージがアウトバウンド・エージェントのキューに格納され、インバウンド・エージェントに伝播されます。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、デフォルトのアウトバウンド・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。『Oracle Workflow 開発者ガイド』の「ワークフロー送信プロトコル」および『Oracle Workflow 開発者ガイド』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）

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_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）

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：

『Oracle Workflow 開発者ガイド』の「Unexpected イベント」

『Oracle Workflow 開発者ガイド』の「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。

注意： イベント・メッセージが後でワークフロー・プロセスに送られる場合、ITEMKEY パラメータや CORRELATION_ID パラメータの値には、シングル・バイト・キャラクタのみ使用できます。これは、イベント・メッセージの CORRELATION_ID 属性がプロセスの項目キーとして使用されるためです。プロセス・インスタンスの項目キーには、シングル・バイト・キャラクタのみ使用できます。マルチバイトの値を含めることはできません。

引数（入力）

p_subscription_guid サブスクリプションのグローバル一意識別子。

p_event イベント・メッセージ。

関連項目：

5-8 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。

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 5-6

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）

p_string 連結したパラメータを含むテキスト文字列。

p_numvalues 文字列に含まれているパラメータの数。

p_separator 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、スペースで区切られ、次の形式で指定されている必要があります。

<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）

p_string 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。

p_key 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。5-69 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。5-68 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。

> **注意：** プロセス・インスタンスの項目キーには、シングル・バイト・キャラクタのみ使用できます。マルチバイトの値を含めることはできません。

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。『Oracle Workflow 開発者ガイド』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）

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。

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.agent.group.create
- oracle.apps.wf.agent.group.update
- oracle.apps.wf.agent.group.delete
- oracle.apps.wf.event.subscription.create
- oracle.apps.wf.event.subscription.update
- oracle.apps.wf.event.subscription.delete
- oracle.apps.wf.event.all.sync

イベント、イベント・グループ、システム、エージェント、エージェント・グループ・メンバー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、エージェント・グループ・メンバー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。

注意： 現在、エージェント・グループを使用できるのは Oracle Applications に組み込まれている Workflow のみです。

引数（入力）

p_event_name イベントの内部名。

p_event_key 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：

5-77 ページ「WF_EVENTS_PKG.Generate」

5-80 ページ「WF_EVENT_GROUPS_PKG.Generate」

5-83 ページ「WF_SYSTEMS_PKG.Generate」

5-86 ページ「WF_AGENTS_PKG.Generate」

5-89 ページ「WF_AGENT_GROUPS_PKG.Generate」

5-92 ページ「WF_EVENT_SUBSCRIPTIONS_PKG.Generate」

『Oracle Workflow 開発者ガイド』の「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.agent.group.create
- oracle.apps.wf.agent.group.update
- oracle.apps.wf.agent.group.delete

- oracle.apps.wf.event.subscription.create
- oracle.apps.wf.event.subscription.update
- oracle.apps.wf.event.subscription.delete
- oracle.apps.wf.event.all.sync

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、エージェント・グループ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。

注意： 現在、エージェント・グループを使用できるのは Oracle Applications に組み込まれている Workflow のみです。

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：

5-78 ページ「WF_EVENTS_PKG.Receive」

5-81 ページ「WF_EVENT_GROUPS_PKG.Receive」

5-84 ページ「WF_SYSTEMS_PKG.Receive」

5-87 ページ「WF_AGENTS_PKG.Receive」

5-90 ページ「WF_AGENT_GROUPS_PKG.Receive」

5-93 ページ「WF_EVENT_SUBSCRIPTIONS_PKG.Receive」

『Oracle Workflow 開発者ガイド』の「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 Generate 関数および Receive 関数が提供されます。

- WF_EVENTS_PKG
  - 5-77 ページ「WF_EVENTS_PKG.Generate」
  - 5-78 ページ「WF_EVENTS_PKG.Receive」
- WF_EVENT_GROUPS_PKG
  - 5-80 ページ「WF_EVENT_GROUPS_PKG.Generate」
  - 5-81 ページ「WF_EVENT_GROUPS_PKG.Receive」
- WF_SYSTEMS_PKG
  - 5-83 ページ「WF_SYSTEMS_PKG.Generate」
  - 5-84 ページ「WF_SYSTEMS_PKG.Receive」
- WF_AGENTS_PKG
  - 5-86 ページ「WF_AGENTS_PKG.Generate」
  - 5-87 ページ「WF_AGENTS_PKG.Receive」
- WF_AGENT_GROUPS_PKG
  - 5-89 ページ「WF_AGENT_GROUPS_PKG.Generate」
  - 5-90 ページ「WF_AGENT_GROUPS_PKG.Receive」
- WF_EVENT_SUBSCRIPTIONS_PKG
  - 5-92 ページ「WF_EVENT_SUBSCRIPTIONS_PKG.Generate」
  - 5-93 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。5-71 ページの「Generate」および 5-73 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。

- 5-77 ページ「WF_EVENTS ドキュメント・タイプ定義」
- 5-79 ページ「WF_EVENT_GROUPS ドキュメント・タイプ定義」
- 5-82 ページ「WF_SYSTEMS ドキュメント・タイプ定義」
- 5-85 ページ「WF_AGENTS ドキュメント・タイプ定義」
- 5-88 ページ「WF_AGENT_GROUPS ドキュメント・タイプ定義」
- 5-91 ページ「WF_EVENT_SUBSCRIPTIONS ドキュメント・タイプ定義」

注意： 現在、エージェント・グループを使用できるのは Oracle Applications に組み込まれている Workflow のみです。

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>
    <CUSTOMIZATION_LEVEL></CUSTOMIZATION_LEVEL>
  </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）

x_guid　　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）

x_message　　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）

x_group_guid イベント・グループのグローバル一意識別子。

x_member_guid 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）

x_message 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）

x_guid　　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）

x_message 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）

x_guid 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）

x_message 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_AGENT_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_AGENT_GROUPS 表のエージェント・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： エージェント・グループのヘッダー情報は、個別のエージェントと同様に、WF_AGENTS 表に定義します。ただし、エージェント・グループ・メンバーの定義は、WF_AGENT_GROUPS 表に格納します。

注意： 現在、エージェント・グループを使用できるのは Oracle Applications に組み込まれている Workflow のみです。

```
<WF_TABLE_DATA>
  <WF_AGENT_GROUPS>
    <VERSION></VERSION>
    <GROUP_GUID></GROUP_GUID>
    <MEMBER_GUID></MEMBER_GUID>
  </WF_AGENT_GROUPS>
</WF_TABLE_DATA>
```

WF_AGENT_GROUPS_PKG.Generate

PL/SQL 構文

```
function Generate
     (x_group_guid in raw,
      x_member_guid in raw)
     return varchar2;
```

説明

指定されたエージェント・グループ・メンバー定義の WF_AGENT_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。

> **注意：** 現在、エージェント・グループを使用できるのは Oracle Applications に組み込まれている Workflow のみです。

引数（入力）

x_group_guid	エージェント・グループのグローバル一意識別子。
x_member_guid	個々のメンバー・エージェントのグローバル一意識別子。

WF_AGENT_GROUPS_PKG.Receive

PL/SQL 構文

```
procedure Receive
     (x_message in varchar2);
```

説明

エージェント・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_AGENT_GROUPS 表に格納します。

注意： 現在、エージェント・グループを使用できるのは Oracle Applications に組み込まれている Workflow のみです。

引数（入力）

x_message エージェント・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_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>
    <CUSTOMIZATION_LEVEL></CUSTOMIZATION_LEVEL>
  </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）

x_guid 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）

x_message 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。

ビジネス・イベント・システムのクリーン・アップ API

Workflow ビジネス・イベント・システムのクリーン・アップ API を使用すると、ビジネス・イベント・システムの標準の WF_CONTROL キューからアクティブでないサブスクライバを削除して、キューをクリーン・アップできます。この API は、WF_BES_CLEANUP という PL/SQL パッケージに定義されています。

Cleanup_Subscribers

PL/SQL 構文

```
procedure Cleanup_Subscribers
     (errbuf out varchar2,
      retcode out varchar2);
```

説明

標準の WF_CONTROL キューのクリーン・アップを実行します。

Oracle Applications またはスタンドアロン版 Oracle Workflow の中間層プロセスは、起動時に WF_CONTROL キューへの JMS サブスクライバを作成します。このキューにイベント・メッセージが格納されると、このキューへの各サブスクライバに対して、イベント・メッセージのコピーが作成されます。ただし、中間層プロセスが終了すると、対応するサブスクライバがデータベースに残ります。処理の効率を向上させるために、WF_CONTROL を定期的にクリーン・アップしてください。それには、Cleanup_Subscribers() を実行して、アクティブでなくなった中間層プロセスのサブスクライバをすべて削除します。

Cleanup_Subscribers() プロシージャは、oracle.apps.wf.bes.control.ping というイベントを送信して、WF_CONTROL キューへの各サブスクライバのステータスをチェックします。対応する中間層プロセスが実行中であれば、応答が返されます。

クリーン・アップ・プロシージャは、次に実行されるとき、前回の実行時に送信した各 Ping イベントに対する応答を受信しているかどうかをチェックします。あるサブスクライバからの応答を受信していない場合、そのサブスクライバは削除されます。

クリーン・アップ・プロシージャは、アクティブでなくなったサブスクライバをすべて削除した後、残りのサブスクライバに新しい Ping イベントを送信します。

クリーン・アップは 12 時間おきに実行することをお薦めします。時間内に各サブスクライバが Ping イベントに応答できるよう、クリーン・アップの実行間隔は 30 分以上にしてください。前回の実行から 30 分未満で再度このプロシージャを実行すると、何の処理も実行されません。

サブスクライバに送信した Ping イベントに関する情報は、最長で 30 日間保持されます。送信から 30 日を超えると、その Ping イベントに関する情報は Cleanup_Subscribers() によって削除されます。

クリーン・アップ処理でアクティブでないサブスクライバを削除できなかった場合、このプロシージャはエラー・メッセージを含むエラー・バッファを返します。また、クリーン・アップのステータスを示すために、次のいずれかのコードを返します。

- 0: 成功
- 1: 警告
- 2: エラー

関連項目：

『Oracle Workflow 管理者ガイド』の「Workflow 制御キューのクリーン・アップ」

『Oracle Workflow 開発者ガイド』の「標準エージェント」

『Oracle Workflow 開発者ガイド』の「ビジネス・イベント・システム制御イベント」

6

Workflow QUEUE API

この章では、Oracle Workflow のアドバンスト・キューイング処理の API について説明します。この API は、ワークフローのアドバンスト・キューイング処理を行う PL/SQL 関数とプロシージャで構成されています。下位互換性を保つためにこれらの API は引き続きサポートされますが、Oracle Workflow リリース 2.6 以降を使用している場合は、Oracle Advanced Queuing との統合にキュー API ではなくビジネス・イベント・システムを使用するようにしてください。

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、ワークフローのアドバンスト・キューイング処理を行います。

注意： 下位互換性を保つためにこれらの API は引き続きサポートされますが、Oracle Workflow リリース 2.6 以降を使用している場合は、Oracle Advanced Queuing との統合にキュー API ではなくビジネス・イベント・システムを使用するようにしてください。

今後のリリースでは、このワークフローのアドバンスト・キューイング処理はビジネス・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。

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、『Oracle9*i* 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』または『Oracle Streams アドバンスト・キューイング・ユーザーズ・ガイドおよびリファレンス』を参照してください。

キュー API

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。

ペイロード構造

Oracle Workflow のキューはすべて、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 6-1

属性名	データ型	説明
ITEMTYPE	VARCHAR2(8)	イベントの項目タイプ。
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：

『Oracle Workflow 開発者ガイド』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」

『Oracle9*i* 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』または『Oracle Streams アドバンスト・キューイング・ユーザーズ・ガイドおよびリファレンス』

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）

itemtype イベントの項目タイプ。

itemkey 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。

actid 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。

result 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」によって決まります。

attrlist 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 varchar2 型の項目属性が作成されます。

correlation 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。Oracle Advanced Queuing 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。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相関識別子が作成されます。

error_stack Oracle Workflow 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
    (dequeuemode in number,
     navigation in number default 1,
     correlation in varchar2 default null,
     itemtype in varchar2 default null,
     payload out system.wf_payload_t,
     message_handle in out raw,
     timeout out boolean);
```

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）

dequeuemode 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、取出し可能で相関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、取出し可能で相関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。

correlation デキュー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。Oracle Advanced Queuing 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 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相関識別子が作成されます。

itemtype イベントの項目タイプ。

message_handle 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 を指定すると相関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);
   queuename       varchar2(30);
   navigation_mode number;
   end_of_queue    boolean;

begin
   queuename:=wf_queue.OUTBOUNDQUEUE;
   i:=0;
   LOOP
      i:=i+1;

      -- always start with the first message then progress to next
      if i = 1 then
         navigation_mode := dbms_aq.FIRST_MESSAGE;
      else
         navigation_mode := dbms_aq.NEXT_MESSAGE;
      end if;
      -- not interested in specific msg_id. Leave it null so
      --as to loop through all messages in queue
      msg_id :=null;
```

```
      wf_queue.DequeueOutbound(
                        dequeuemode     => dbms_aq.BROWSE,
                        payload         => event,
                        navigation      => navigation_mode,
                        message_handle  => msg_id,
                        timeout         => end_of_queue);
      if end_of_queue then
         exit;
      end if;

      -- print the correlation itemtype:itemKey
      dbms_output.put_line('Msg '||to_char(i)||' = '||
                        event.itemtype||':'||event.itemkey
                        ||' '||event.actid||' '
                        ||event.param_list);
   END LOOP;

end;
/
```

DequeueEventDetail

構文

```
procedure DequeueEventDetail
    (dequeuemode in number,
     navigation in number default 1,
     correlation in varchar2 default null,
     itemtype in out varchar2,
     itemkey out varchar2,
     actid out number,
     function_name out varchar2,
     param_list out varchar2,
     message_handle in out raw,
     timeout out boolean);
```

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）

dequeuemode 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。

navigation
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、取出し可能で相関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、取出し可能で相関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。

correlation
デキュー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。Oracle Advanced Queuing 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 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相関識別子が作成されます。

acctname
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。

itemtype
相関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。

message_handle
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 を指定すると相関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）

queuename イベントが削除されるキューの名前。

message_handle 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）

queuename イベントが削除されるキューの名前。

itemtype 削除するイベントのオプションの項目タイプ。

correlation 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。Oracle Advanced Queuing 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 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。6-5 ページの「EnqueueInbound」を参照してください。

引数（入力）

itemtype 処理するイベントのオプションの項目タイプ。

correlation 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 Workflow 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）

queuename メッセージのハンドルが取り出されるキューの名前。

itemtype メッセージの項目タイプ。

itemkey 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。

actid 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。

correlation 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 Workflow 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：

『Oracle Workflow 開発者ガイド』の「デフォルト・イベント・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。6-3 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。6-3 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）

itemtype メッセージの項目タイプ。

itemkey 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。

actid 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。6-3 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）

itemtype メッセージの項目タイプ。

itemkey 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。

actid 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。6-3 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）

itemtype メッセージの項目タイプ。

itemkey 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。

actid 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。

attrName メッセージに追加する項目属性の内部名。

attrValue 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。6-3 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）

itemtype メッセージの項目タイプ。

itemkey 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。

actid 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。

result 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。

7

文書管理 API

この章では、Oracle Workflow の文書管理の API について説明します。この API は、文書管理システムとの統合を行う PL/SQL 関数とプロシージャで構成されています。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。

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。

関連項目：

『Oracle Workflow 開発者ガイド』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」

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）

username 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。

document_identifier 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。2-61 ページ「GetItemAttrDocument」および 2-54 ページ「SetItemAttrDocument」を参照してください。

display_icon TRUE または FALSE。TRUE の場合は、URL とともに、ペーパー・クリップの添付ファイル・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 を戻し、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）

username 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。

callback_function 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。

display_icon TRUE または FALSE。TRUE の場合は、URL とともに、ペーパー・クリップの添付ファイル・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）

frame_name 現行の UI で対話する HTML フレームの名前。

form_name 現行の UI で対話する HTML フォームの名前。

document_id_field_name 結果の文書 ID を書き込む、現行の UI の HTML フィールドの名前。結果の文書 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。

DM:<*node_id*>:<*document_id*>:<*version*>

<*nodeid*> は、「文書管理ノード」Web ページで定義されている、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。

document_name_field_name 結果の文書名を書き込む、現行の UI の HTML フィールドの名前。

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・レベルで操作する。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が定義される。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ある。

アクティビティ（Activity）

ビジネス・プロセス中に実行される作業単位。

アクティビティ属性（Activity Attributes）

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する。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する。

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。

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する。

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、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る。

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情報。イベント・データは XML 文書として作成できる。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する。

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る。

エージェント（Agent）

システム内の通信の名前付きポイント。

エージェント・リスナー（Agent Listener）

インバウンド・エージェントでイベント・メッセージを処理するサービス・コンポーネントの一種。

外部 Java 関数（External Java Functions）

Oracle Database 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、自動化されている作業単位。

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ある。

項目属性（Item Attribute）

「項目タイプ属性」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ある。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る。

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る。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ある。

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る。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る。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、処理されない。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る。

サービス・コンポーネント（Service Component）

汎用サービス・コンポーネント・フレームワークで管理できるように、このフレームワーク標準に従って定義された Java プログラムのインスタンス。

サービス・コンポーネント・コンテナ（Service Component Container）

所有する個々のサービス・コンポーネントの実行を管理するサービスまたはサーブレットのインスタンス。コンテナは、そのコンポーネントのステータスを監視し、コンテナ自体とそのコンポーネントの制御イベントを処理する。

サブスクリプション（Subscription）

「イベント・サブスクリプション」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る。

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る。

属性（Attribute）

「アクティビティ属性」、「項目タイプ属性」または「メッセージ属性」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のある期間。

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 ページ（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 ページ。

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作業単位。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する。

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サービス・コンポーネントの一種。

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。

汎用サービス・コンポーネント・フレームワーク（Generic Service Component Framework）

バックグラウンド Java サービスの管理の簡略化と自動化に役立つ機能。

ビジネス・イベント（Business Event）

「イベント」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の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）に設定したり、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る。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ある。

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る。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る。

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る。「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る。「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る。

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管理、関数の自動実行、通知の送信、完了したアクティビティの履歴保存、およびエラー条件の検出とエラー・プロセスの開始処理が行われる。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にされる。

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。

索引

F

G

T

U

V

W

あ

い

え

お

か

き

こ

さ

し

せ

そ

ち

つ

て

と

は

ひ

ふ

へ

ゆ

る

れ

ろ

わ