WebSphere Process Server for z/OS

バージョン 6.0.2

Business Process Choreographer

お願い

本書をご使用になる前に、 413 ページの『特記事項』に記載されている情報をお読みください。

本書は、WebSphere Process Server for z/OS (製品番号 5655-N53) バージョン 6、リリース 0、モディフィケーション 2、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

IBM 発行のマニュアルに関する情報のページ

http://www.ibm.com/jp/manuals/

こちらから、日本語版および英語版のオンライン・ライブラリーをご利用いただけます。また、マニュアルに関するご意見やご感想を、上記ページよりお送りください。今後の参考にさせていただきます。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： WebSphere Process Server for z/OS
Version 6.0.2
Business Process Choreographer

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2007.3

この文書では、平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

第 1 章 Business Process Choreographer の使用計画

ビジネス・プロセスまたはヒューマン・タスクを含むエンタープライズ・アプリケーションをインストールする前に、ビジネス・プロセスまたはヒューマン・タスクを実行するアプリケーション・サーバーまたはクラスターごとに、ビジネス・プロセス・コンテナーおよびヒューマン・タスク・コンテナーを構成する必要があります。

1. Business Process Choreographer をクラスター上で使用する場合は、クラスターの計画を立ててください。
2. 使用するデータベース・システムを決定します。

   注: Business Process Choreographer Observer は、Cloudscape または DB2 のいずれかのデータベースを必要とします。

   - Cloudscape

     注:

     – Cloudscape™ Network Server では XA がサポートされないため、Business Process Choreographer で使用できるのは、リモート側でアクセスできない組み込みの Cloudscape バージョンだけです。この制限のため、Cloudscape をクラスター環境で Business Process Choreographer のデータベース・システムとして使用することはできません。

     – Cloudscape ではデータベース・アクセスが直列化されます。従って、アクティビティーの並列実行をサポートするようにモデル化されたフロー内であっても、アクティビティーは常に順次実行されます。

   - DB2® for z/OS®
3. z/OS Universal JDBC ドライバー・プロバイダーおよびデータ・ソース用の DB2 については、要件を確認してください。
4. データベースをホストするサーバーを決定します。 データベース・サーバーがリモートである場合は、適切なデータベース・クライアントまたは XA をサポートしているタイプ 4 の JDBC ドライバーが必要です。
5. 使用する Java™ Message Service (JMS) プロバイダーを決定します。
   - WebSphere® デフォルト・メッセージング
   - WebSphere MQ
6. 44 ページの『ビジネス・プロセス・コンテナーのインストール・ウィザードの設定』の説明に従って設定を計画します。
7. 60 ページの『ヒューマン・タスク・コンテナーのインストール・ウィザードの設定』の説明に従って設定を計画します。
8. ビジネス・プロセス・コンテナーを手動で構成する (推奨) か、インストール・ウィザードを使用してビジネス・プロセス・コンテナーを構成するかを決定します。
   - ビジネス・プロセス・コンテナーを手動で構成する場合は、54 ページの『ビジネス・プロセス・コンテナーの設定』の説明に従って設定を計画します。

- インストール・ウィザードを使用する場合は、 44 ページの『ビジネス・プロセス・コンテナーのインストール・ウィザードの設定』の説明に従って設定を計画します。

WebSphere Process Server をインストールした後、Business Process Choreographer の構成を実行できます。

Business Process Choreographer について

ビジネス・プロセス・コンテナーとヒューマン・タスク・コンテナーが提供する機能について説明します。

Business Process Choreographer は、WebSphere Application Server 環境にあるビジネス・プロセスとヒューマン・タスクの両方をサポートするエンタープライズ・ワークフロー・エンジンです。このような構成を使用して、人間による対話を必要とする J2EE リソース、サービス、およびアクティビティーを統合することができます。Business Process Choreographer は、ビジネス・プロセスのライフ・サイクルおよびヒューマン・タスクを管理し、関連したプロセス・モデルをナビゲートして、該当するサービスを呼び出します。

Business Process Choreographer は、次の機能を提供します。

- ビジネス・プロセスおよびヒューマン・タスクのサポート。ビジネス・プロセスは、Web Services Business Process Execution Language (WS-BPEL、略記 BPEL) を使用してビジネス・プロセスをモデル化する標準的な方法を提供します。ヒューマン・タスクでは、Task Execution Language (TEL) を使用して、人間から人間へ、人間からコンピューターへ、コンピューターから人間へ、といった人間が関わる対話をモデル化できます。ビジネス・プロセスおよびヒューマン・タスクは、両方ともサービス指向アーキテクチャー (SOA) またはサービス・コンポーネント・アーキテクチャー (SCA) でのサービスとして公開され、単純なデータ・オブジェクトおよびビジネス・オブジェクトもサポートします。
- ビジネス・プロセスおよびヒューマン・タスクとの対話用のカスタマイズ・アプリケーションを開発するためのアプリケーション・プログラミング・インターフェース。
- Business Process Choreographer Explorer。この Web アプリケーションは、ビジネス・プロセスとヒューマン・タスクの管理機能を提供します。 詳しくは、 102 ページの『Business Process Choreographer Observer について』を参照してください。
- Business Process Choreographer Observer。この Web アプリケーションによって、ビジネス・プロセスおよびヒューマン・タスクの実行状態を監視できます。詳しくは、 98 ページの『Business Process Choreographer Explorer について』を参照してください。

関連概念

3 ページの『Business Process Choreographer および Network Deployment』
Business Process Choreographer を Network Deployment 環境で使用する場合の特別な考慮事項について説明します。

5 ページの『クラスタリングに合わせた Business Process Choreographer のシナリオ』

クラスターを使用する Business Process Choreographer のシナリオに関するさまざまな構成オプションおよび考慮事項について説明します。

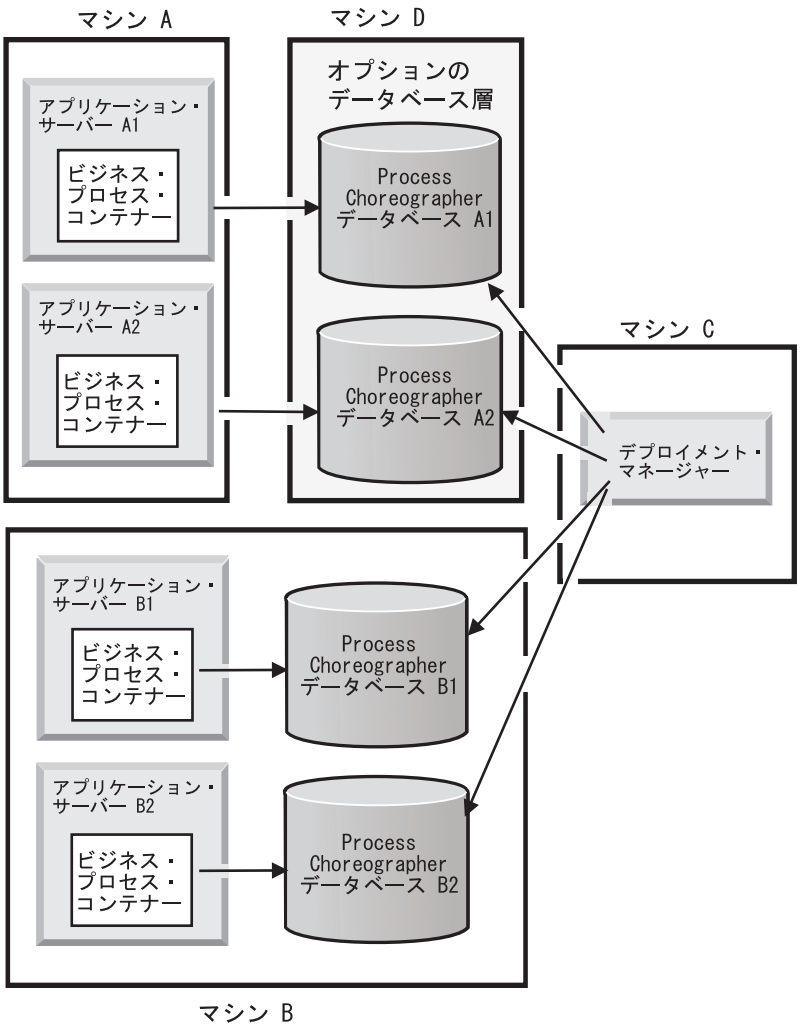
Business Process Choreographer および Network Deployment

Business Process Choreographer を Network Deployment 環境で使用する場合の特別な考慮事項について説明します。

Network Deployment (ND) を使用する場合は、以下の点を考慮する必要があります。

デプロイメント・マネージャーには Business Process Choreographer データベースへのアクセス権が必要

デプロイメント・マネージャーには、セル内にあるビジネス・プロセス・コンテナーおよびヒューマン・タスク・コンテナーが使用するすべての Business Process Choreographer データベースへのアクセス権が必要です。適切なデータベース・クライアントまたは JDBC ドライバーをデプロイメント・マネージャーをホストするコンピューターにインストールし、JDBC プロバイダーのクラスパスの WebSphere 環境変数をデプロイメント・マネージャーのノード・スコープに設定する必要があります。以下の図に、この構成を示します。



Business Process Choreographer をクラスター上にインストールする前に

「*WebSphere Process Server の管理*」の PDF に記載されているとおりにクラスターを作成してあることを確認します。

Business Process Choreographer のクラスター上へのインストールおよび構成後に必要なカスタマイズ

キュー・マネージャーの WebSphere MQ クラスターを使用するクラスター化セットアップを作成している場合は、何らかの手動カスタマイズを実行して、Business Process Choreographer の各インスタンスがそれ専用のキュー・マネージャーを使用するようにしなければなりません。必要な操作は、『ビジネス・プロセス・コンテナーの構成』で説明します。

Business Process Choreographer によるクラスタリング使用の詳細については、『クラスタリングに合わせた Business Process Choreographer のシナリオ』を参照してください。

ビジネス・プロセスとヒューマン・タスクのいずれかまたはその両方を含むアプリケーションをインストールする前に

アプリケーションをインストールする前に、アプリケーションのインストール先にするサーバーが稼働していることを確認します。少なくとも 1 つのサーバーは、Java Naming and Directory Interface (JNDI) 名を解決する目的で稼働している必要があります。

関連概念

2 ページの『Business Process Choreographer について』
ビジネス・プロセス・コンテナーとヒューマン・タスク・コンテナーが提供する機能について説明します。

クラスタリングに合わせた Business Process Choreographer のシナリオ

クラスターを使用する Business Process Choreographer のシナリオに関するさまざまな構成オプションおよび考慮事項について説明します。

Business Process Choreographer インスタンスに対して WebSphere Process Server クラスターを使用する主な利点は以下のとおりです。
- フェイルオーバーによる高いサービス可用性
- ワークロード容量の増大
- リソース使用効率の向上
- ワークロードの共用
- 容易な管理

構成オプション

Business Process Choreographer を構成する方法は多数あるため、クラスターの構成は、通常、きわめて複雑になります。アプリケーション・サーバーの作成を開始する前に検討するための主なオプションの一部について、以下にその概要を説明します。

WebSphere Process Server セル内のノード数

1 つ以上。すべてのノードは、1 つのデプロイメント・マネージャーから管理します。

各 WebSphere Process Server クラスター内のノード数

1 つ以上。WebSphere の水平方向のクラスタリングにより、サービス可用性と全体のワークロード容量が向上します。

各ノードでのアプリケーション・サーバーの数

1 つ以上。WebSphere Process Server の垂直方向のクラスタリングにより、リソースの使用効率が向上します。

データベース・ホスト
- 専用サーバー上で、リモート
- クラスター内のいずれかのアプリケーション・サーバーにローカル

データベースは、(なるべくホット・スタンバイ機能を備えた) 専用サーバー上でホストすることをお勧めします。

アプリケーション・メッセージ・キュー
- ローカル・キュー

- リモート・キュー

接続 (WebSphere Platform Messaging)

WebSphere Platform Messaging (WPM) を使用する場合は、同じクラスターまたはリモートのクラスター内でメッセージ・エンジンを構成できます。Business Process Choreographer の場合は、他の WebSphere Process Server コンポーネントに使用した手法と同じ手法を使用してください。考えられるシナリオの詳細については、「*WebSphere Process Server の管理*」PDF の『サービス・アプリケーションをサポートするサーバーまたはクラスターの作成 (Preparing a server or cluster to support service applications)』を参照してください。推奨の構成では、Business Process Choreographer のインストール先クラスターとは別のクラスターでメッセージング・エンジンを稼働します。この方法は、WebSphere MQ と組み合わせて使用できる中央キュー・マネージャー構成の場合に類似しています。

接続 (WebSphere MQ キュー・マネージャー)

制約事項: この製品での WebSphere MQ メッセージングのサポートは、Business Process Choreographer (BPC) に制限されます。 BPC 以外の WebSphere MQ メッセージングはこの製品ではサポートされていません。

- 1 つのクラスター内にあるアプリケーション・サーバーのキューをホストする 1 つの中央 (リモート) キュー・マネージャー通常はこの構成をお勧めします。
- アプリケーション・サーバーごとに 1 つのローカル・キュー・マネージャー。この場合はフェイルオーバー機能やプロセス内ワークロードの共用機能がありません。
- ノードごとに 2 つのローカル・キュー・マネージャー、および複数のアプリケーション・サーバー間のワークロードについてバランスを取るために使用する WebSphere MQ クラスタリング。

  異なる Business Process Choreographer インスタンス間でワークロードを配分するには、各アプリケーション・サーバーのビジネス・プロセス・コンテナーが使用するキュー・マネージャーが、同じ WebSphere MQ クラスターのメンバーであることが必要です。この構成ではフェイルオーバー機能が提供されないため、通常はお勧めしません。

クラスター化シナリオを使用する場合は、JMS プロバイダーとしては WebSphere MQ をお勧めしません。

データベース・システム

Cloudscape を除き、サポートされているすべてのデータベースを使用できます。

ホット・スタンバイ・サーバー

ホット・スタンバイ・サーバーのオプションは、以下のとおりです。

- なし
- データベース用
- 中央キュー・マネージャー用

クラスター・タイプ: このトピックでは、異なる 2 種類のクラスター について言及します。*WebSphere* クラスター は、アプリケーション・サーバーをグループ化してワークロードを共有し、サービス可用性を向上させます。*WebSphere MQ* クラスター (旧称 MQSeries® クラスター) は、WebSphere MQ キュー・マネージャー群を

グループ化したもので、プロセス内ワークロード・バランシングを実現するために使用できます。

高可用性

Business Process Choreographer サービスの高可用性を実現するには、以下の項目について検討してください。

- WebSphere クラスター内にアプリケーション・サーバーの複製を作成することにより、アプリケーション・サーバーが提供するサービスの可用性は向上します。
- Business Process Choreographer データベースは、ホット・スタンバイ・システムを使用して保護できる Single Point of Failure です。
- 中央キュー・マネージャーは、ホット・スタンバイ・ハードウェアで保護できます。

垂直クラスタリングによるリソース使用効率の最大化

パフォーマンスを向上させるためには、同じノード上に複数のアプリケーション・サーバー・インスタンスを作成する必要があります。これにより、Business Process Choreographer は使用可能なシステム・リソースを使用できます。

ワークロードの共用

WebSphere MQ メッセージングを使用しており、Business Process Choreographer の異なるインスタンスが同じワークロードを共用するようにするには、これらのインスタンスが使用するキュー・マネージャー構成は、以下のいずれかにする必要があります。

中央キュー・マネージャー

中央キュー・マネージャーは、Business Process Choreographer が必要とするキューのホストとして動作します。WebSphere クラスター内にあるすべての Business Process Choreographer インスタンスは、同じキューから読み取ります。

WebSphere MQ クラスター

各アプリケーション・サーバーには、2 つのキュー・マネージャーがあります。一方のキュー・マネージャーはローカル・キューのホストとして動作し、メッセージの読み取り用として使用されますが、もう一方のキュー・マネージャーにはホストの対象となるキューが存在せず、メッセージの書き込み専用として使用されます。WebSphere クラスター内にあるすべての Business Process Choreographer インスタンスのすべてのキュー・マネージャーは、WebSphere MQ クラスターの構成メンバーになっています。

ホストの対象となるキューが存在しないキュー・マネージャーにのみメッセージを書き込むと、その結果、メッセージはクラスター内のすべての get キュー・マネージャーに均等に分配されます。インストール・ウィザードを使用し、ビジネス・プロセス・コンテナーをクラスター上にインストールして構成したら、アプリケーション・サーバーごとに 2 つの接続ファクトリーを手動で変更して、ローカルの get キュー・マネージャーおよび put キュー・マネージャーを指すようにする必要があります。

Business Process Choreographer データベース

データベースのホスティングは、(なるべくホット・スタンバイ機能を備えた) 専用サーバー上での実行をお勧めします。データベースの格納先のサーバーは WebSphere セルの外側でも構いませんが、そこにはデプロイメント・マネージャーがアクセスできる必要があります。

データベースの計画を立てる場合は、以下の点を考慮してください。

- WebSphere クラスター内にあるすべてのビジネス・プロセス・コンテナーは、同じデータベースにアクセスします。対照的に、WebSphere クラスター内に存在しないビジネス・プロセス・コンテナーには、それ専用のデータベースが必要です。
- Business Process Choreographer のリモート・データベースにアクセスできるようにするには、ローカル・データベースのないすべてのアプリケーション・サーバーに、適切なデータベース・クライアントか、タイプ 4 の Java Database Connectivity (JDBC) ドライバーをインストールする必要があります。
- デプロイメント・マネージャーは、 WebSphere セル内の Business Process Choreographer インスタンスのすべてのデータベースにアクセスできる必要があります。デプロイメント・マネージャーを使用してビジネス・プロセスをインストールできるようにするには、その前にこのアクセスを使用可能にする必要があります。
- データベースはサポートされている任意のデータベースを使用できますが、Cloudscape は使用できません。これは、Cloudscape Network Server が XA をサポートしておらず、組み込みの Cloudscape にはリモート・マシンからアクセスできないためです。
- WebSphere セル内にある Business Process Choreographer インスタンスが使用する各データベースには、固有の名前を使用してアクセスできる必要があります。DB2 の場合は、デプロイメント・マネージャーおよびアプリケーション・サーバー上では同じデータベース名を使用する必要があります。
- データベースは Single Point of Failure です。この問題の唯一の解決策は、High Availability Cluster Multiprocessing (HACMP™) on AIX® などの高可用性ホット・スタンバイ・ソリューションを使用することです。

Business Process Choreographer Observer データベース

Business Process Choreographer Observer には、データベースが必要です。 Business Process Choreographer ビジネス・プロセス・コンテナーと同じデータベースを使用できますが、実動システムでは、パフォーマンス上の理由で別個のデータベースを使用するほうが望ましいと言えます。

WebSphere プラットフォーム・メッセージング JMS プロバイダー

Business Process Choreographer は、クラスタリング、ワークロード管理、フェイルオーバーをサポートする WebSphere プラットフォーム・メッセージング (WPM) を使用できます。

以下の 2 つのトポロジーがサポートされています。

- メッセージング・リソースのホストになっているのは、アプリケーションとは異なるクラスターです。このクラスターはフェイルオーバー機能を備えているだけ

ではなく、管理オーバーヘッドも少ないため、これは推奨のトポロジーです。このトポロジーは、WebSphere MQ の中央キュー・マネージャー手法の場合と似ています。

- メッセージング・リソースとアプリケーションのホストになっているのは、同じクラスターです。このトポロジーは高性能を発揮するには理想的ですが、管理上の労力が増えます。特に変更の適用時が顕著です。

WPM を使用するときに適用する考慮事項の詳細については、「*WebSphere Process Server の管理*」PDF の『クラスター化環境の構築 (Creating a clustered environment)』を参照してください。

WebSphere MQ

制約事項: この製品での WebSphere MQ メッセージングのサポートは、Business Process Choreographer (BPC) に制限されます。 BPC 以外の WebSphere MQ メッセージングはこの製品ではサポートされていません。

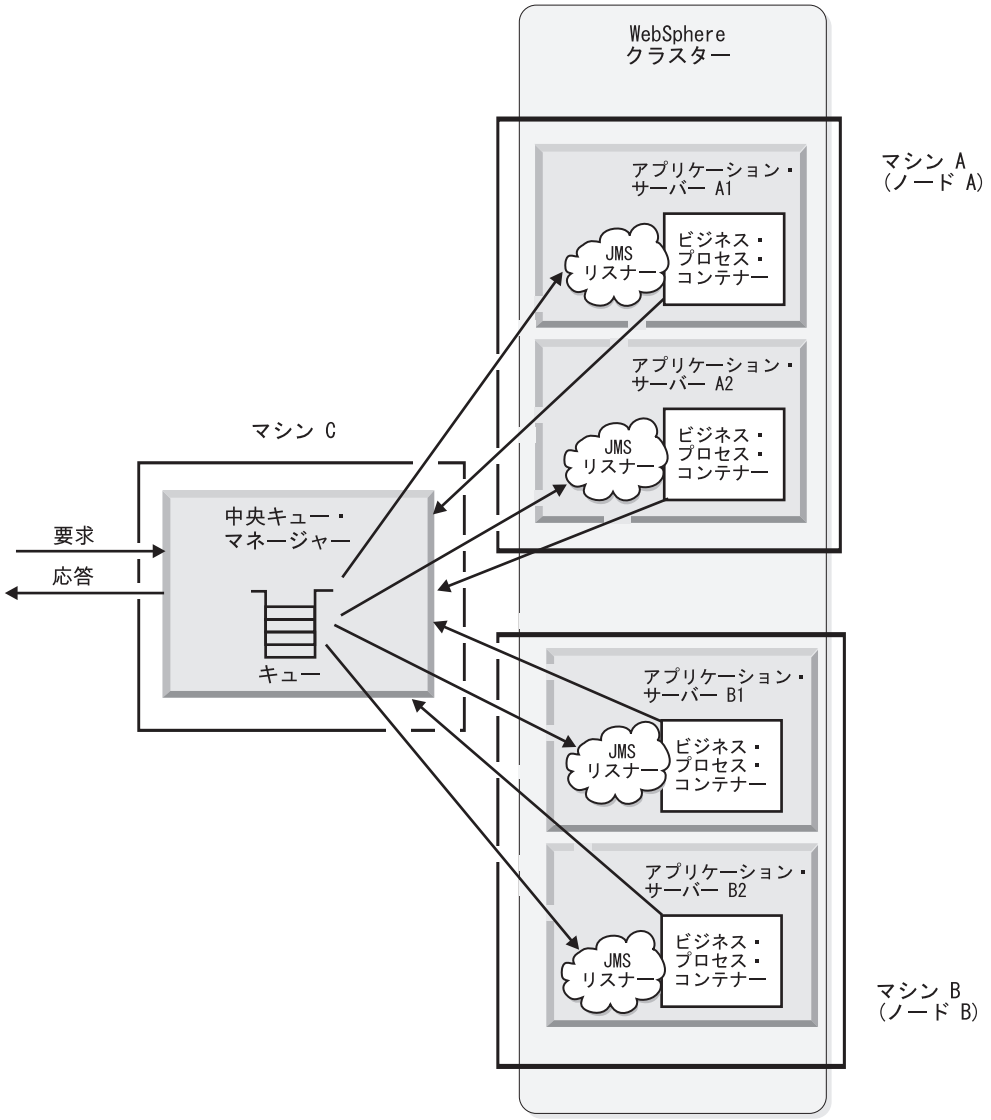
Business Process Choreographer は、WebSphere MQ キューを使用して要求の受信や応答の送信を実行できます。クラスター化シナリオを使用する場合は、JMS プロバイダーとしては WebSphere MQ をお勧めしません。WebSphere MQ を使用する場合は、Business Process Choreographer がインバウンドとアウトバウンドのサービス呼び出しのために使用する Service Component Architecture (SCA) のデフォルト・メッセージングを引き続き構成する必要があります。Business Process Choreographer のホストとして動作する各アプリケーション・サーバーには、次のいずれかのオプションが必要です。

- すべてのキューのホストとなる中央キュー・マネージャーへのアクセス機能
- WebSphere MQ クラスターのメンバーではないローカル・キュー・マネージャー
- WebSphere MQ クラスターのメンバーである 2 つのローカル・キュー・マネージャー

中央キュー・マネージャー

中央キュー・マネージャーをすべてのキューに対して使用すると、管理が容易になります。1 つのキュー・マネージャーが、ヒューマン・タスクおよびビジネス・プロセスのすべての複製コンテナーによって使用されます。ただし、中央キュー・マネージャーを使用すると、高可用性システムでのホスティングが必要な Single Point of Failure が発生します。

次の図に、別のサーバー上にある 1 つの中央キュー・マネージャーを使用する WebSphere クラスター内のすべてのアプリケーション・サーバーを示します。ビジネス・プロセス・コンテナーと一緒に表示されている各アプリケーション・サーバーは、ヒューマン・タスク・コンテナーも保有できます。



WebSphere MQ クラスタリングを使用しないローカル・キュー・マネージャー

この例では、Business Process Choreographer の標準的なスタンドアロン構成を示しています。各ビジネス・プロセス・コンテナーには、ローカル・キュー・マネージャーが 1 つずつ存在します。この方法では、プロセス内ワークロード共用機能もフェイルオーバー・サービスの可用性も提供されません。

WebSphere MQ クラスタリング

この複雑な手法はお勧めしません。この手法では、WebSphere クラスター内の Business Process Choreographer サービスに対してプロセス内ワークロード共用機能がサポートされます。クラスター内のすべてのビジネス・プロセスは、UNIX® ワークステーションか Windows® ワークステーションのいずれか一方のみの上で実行する必要があります。UNIXサーバーと Windowsサーバーを組み合わせた場合は動作しません。

各アプリケーション・サーバーには、2 つのローカル・キュー・マネージャーが必要です。1 つはメッセージの書き込み用で、もう 1 つはメッセージの読み取り用で

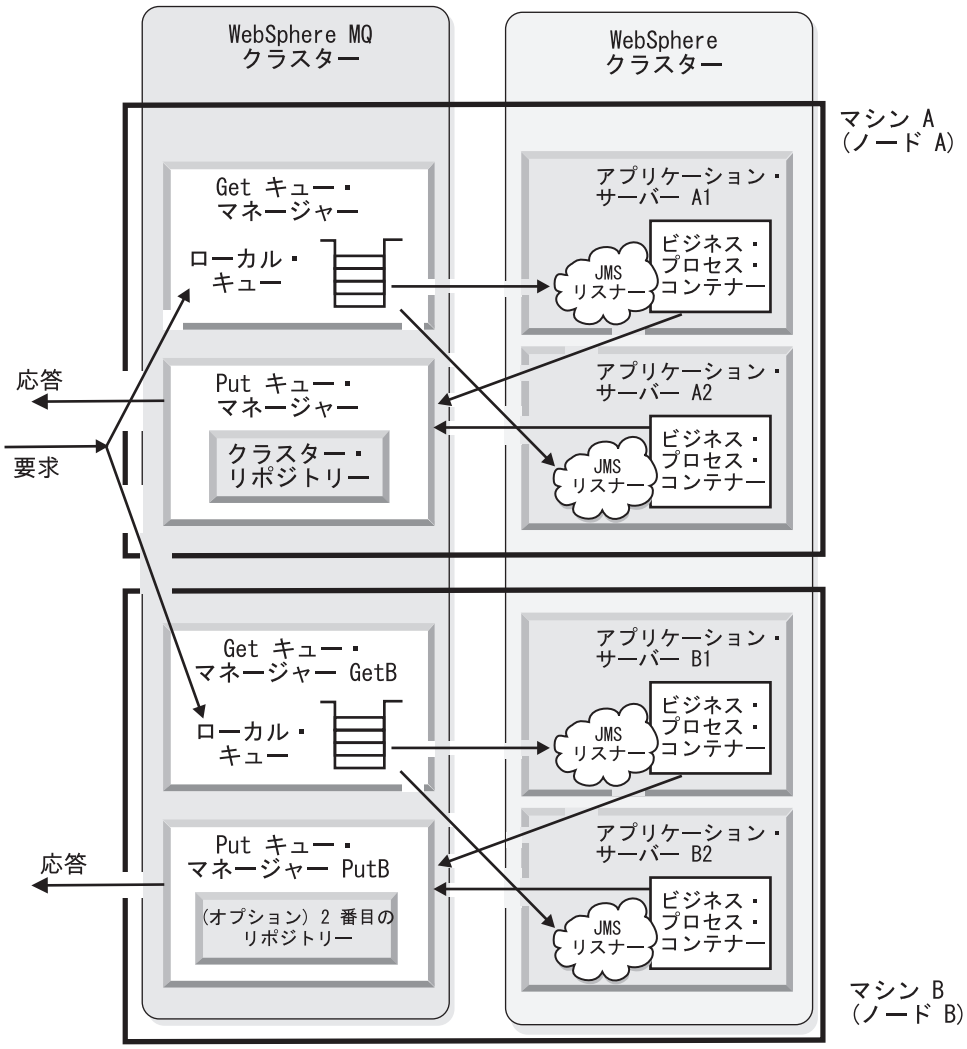
す。すべてのキュー・マネージャーは、同じ WebSphere MQ クラスターのメンバーになります。Windows システムの場合は、すべてのキュー・マネージャーが同じバインディング・プロトコルを使用する必要があります。UNIX システムの場合は、put キュー・マネージャーと get キュー・マネージャーが異なるプロトコルを使用する必要があります。例えば、すべての put キュー・マネージャーがバインディング・プロトコル (プロセス間通信) を使用し、すべての get キュー・マネージャーがデフォルトのクライアント (TCP/IP) プロトコルを使用するようにキューの接続ファクトリーを変更できます。

Windows システムおよび UNIX システムの場合、ローカルのバインディング・トランスポート・タイプを使用すると、クライアント・トランスポート・タイプを使用した場合より約 5% 高速になりますが、ローカルの WebSphere MQ キュー・マネージャーを停止するためにアプリケーション・サーバー全体を停止する必要があるという影響があります。

WebSphere クラスター内の各ビジネス・プロセス・コンテナーは、それ専用のキュー・マネージャーに合わせてカスタマイズする必要があります。

WebSphere MQ クラスター内の複数のキュー・マネージャーによってクラスター・リポジトリーを構成することをお勧めします。

次の図には、アプリケーション・サーバーが使用するキュー・マネージャーが WebSphere MQ クラスター内でグループ化される仕組みを示します。キュー・マネージャーの WebSphere MQ クラスターは、アプリケーション・サーバーの WebSphere クラスターと並列になっています。要求はクラスター内の読み取りキュー全体で共有されます。



WebSphere クラスターの作成方法

Business Process Choreographer に対してクラスターを作成するための手順は、いくつかあります。スタンドアロン・サーバーを既に構成している場合は、 116 ページの『Business Process Choreographer がクラスターに対して構成されているサーバーのプロモート』を実行し、まだ構成していない場合には、以下の手順を行ってください。

1. クラスター・メンバーのサーバー・テンプレートである defaultProcessServer テンプレートを使用して、クラスターを作成します。
2. クラスターにメンバーを追加します。
3. サービス・アプリケーションに対してクラスターを使用可能にします。
4. Business Process Choreographer Observer を使用してビジネス・プロセスおよびヒューマン・タスクをモニターする場合、Common Event Infrastructure (CEI) がクラスター上で構成されていることを確認します。
5. クラスター上で Business Process Choreographer を構成します。
6. WebSphere MQ を使用していて、WebSphere MQ の構成がローカル・キュー・マネージャーの WebSphere MQ クラスターである場合は、接続ファクトリーを変更する必要があります。各キュー・マネージャーの名前は異なるため、各複製

アプリケーション・サーバーの接続ファクトリーを変更して、クラスター全体にわたる標準的な Business Process Choreographer インストール・ウィザード構成との固有の違いを反映させる必要があります。

関連概念

2 ページの『Business Process Choreographer について』
ビジネス・プロセス・コンテナーとヒューマン・タスク・コンテナーが提供する機能について説明します。

第 2 章 Business Process Choreographer の構成

ここでは、ビジネス・プロセスおよびヒューマン・タスク用に Business Process Choreographer コンテナーを構成する方法について説明します。また、Business Process Choreographer Explorer および Business Process Choreographer Observer を構成する方法についても説明します。

Network Deployment (ND) 環境の場合、Service Component Architecture (SCA) が構成されていることを確認します。「**サーバー**」 → 「**アプリケーション・サーバー**」 → ***server_name*** をクリックし、「**ビジネス・インテグレーション**」セクションで、「**Service Component Architecture**」をクリックします。必要に応じて変更し、「**適用**」をクリックします。 Business Process Choreographer Observer を ND 環境にインストールする場合、Common Event Infrastructure (CEI) を既に構成している必要があります。

WebSphere Process Server for z/OS のインストール・スクリプトおよび構成スクリプトによる Business Process Choreographer のインストール (サンプル Business Process Choreographer 構成の作成方法を含む) については、WebSphere Process Server for z/OS のインストールと構成の PDF を参照してください。

Business Process Choreographer は、ビジネス・プロセスとヒューマン・タスクを含むエンタープライズ・アプリケーションをサポートします。ビジネス・プロセスのコンテナーおよびヒューマン・タスクのコンテナーを提供します。これらのコンテナーは、使用前にインストールおよび構成する必要があります。ヒューマン・タスク・コンテナーにはビジネス・プロセス・コンテナーおよびスタッフ・サービスが必要です。Business Process Choreographer Explorer は、人間による対話と、ビジネス・プロセスおよびヒューマン・タスクの管理のための、Web クライアント・インターフェースを提供します。 Business Process Choreographer Observer を使用して、完了したプロセスおよびタスクに関するレポートを作成できます。また、Business Process Choreographer Observer を使用して、実行中のプロセスおよびタスクの状況を表示することもできます。

1. Business Process Choreographer のサンプル構成パラメーターの値を応答ファイルに指定した場合には、ビジネス・プロセス・コンテナー、ヒューマン・タスク・コンテナー、および Business Process Choreographer Explorer を含むサンプル構成が既に存在しています。

   これらのコンポーネントが構成されているかどうかを確認するには、名前が `BPEContainer`、`BPCExplorer`、および `TaskContainer` で始まるエンタープライズ・アプリケーションの有無を管理コンソールで調べます。

   使用する応答ファイルにより、サンプル構成は Cloudscape データベース向けになることも、DB2 for z/OS データベースと WebSphere デフォルト・メッセージング・プロバイダー向けになることもあります。

   注: Cloudscape データベースを使用するサンプル Business Process Choreographer 構成は、実動システムには適していません。 Business Process Choreographer の構成は 1 種類に限られるため、Business Process Choreographer の構成を

続行して WebSphere MQ または別のデータベースを使用するには、 117 ページの『第 3 章 Business Process Choreographer 構成の除去』 で説明するように、Cloudscape を使用するサンプル構成を削除する必要があります。

2. リソースを構成します。

   以下のいずれかの方法で、リソースを構成できます。

   a. 手動 (**推奨**)。管理コンソールでプロパティーを設定するか、管理スクリプト処理によりプロパティーを設定します。

      管理スクリプトを使用して Business Process Choreographer を構成する場合は、 18 ページの『bpeconfig.jacl スクリプト・ファイルを使用した Business Process Choreographer の構成』 を参照してください。

   b. 自動。管理コンソールで使用できるインストール・ウィザードを使用します。

      管理コンソールで使用できるインストール・ウィザードを使用する場合は、以下の手順を両方とも実行してください。

      - 32 ページの『インストール・ウィザードを使用したビジネス・プロセス・コンテナーの構成』
      - 57 ページの『インストール・ウィザードを使用したヒューマン・タスク・コンテナーの構成』

      注: インストール・ウィザードは、WebSphere リソースのみを構成します。インストール・ウィザードを使用してビジネス・プロセス・コンテナーを構成する場合は、データベース (Cloudscape または DB2) および WebSphere MQ キュー (WebSphere MQ を Java Message Service (JMS) プロバイダーとして使用する場合) を作成するために、それぞれに対応する手動の手順を実行する必要があります。

3. LDAP スタッフ・プラグインを使用している場合は、次の操作を行います。 67 ページの『LDAP スタッフ・プラグイン・プロバイダーの構成』. システム・プラグイン・プロバイダーとユーザー・レジストリー・スタッフ・プラグイン・プロバイダーは、構成せずに使用できます。

4. Network Deployment (ND) 環境で、ヒューマン・タスク・コンテナー・インストール・ウィザードを使用したか bpeconfig.jacl スクリプトの実行時にエラーが発生した場合は、SchedulerCalendars アプリケーションをセットアップする必要があります。 以下のいずれかを実行します。

   - SchedulerCalendars アプリケーションをサーバーにインストール済みの場合は、以下の手順を実行して追加のサーバーにスケジューラーをインストールします。

     a. 「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」を選択します。

     b. 「**SchedulerCalendars**」を選択します。

     c. 「追加プロパティー (Additional Properties)」セクションの下で、「**モジュールをサーバーにマップ (Map modules to servers)**」を選択します。

     d. 「**モジュール・カレンダー (Module Calendars)**」のチェック・ボックスを選択します。

e. ビジネス・プロセス・コンテナーを構成したサーバーおよびクラスターをすべて選択します。SchedulerCalendars アプリケーションを存続するサーバーおよびクラスターも、必ず選択するようにしてください。

f. 「**適用**」を選択します。

g. 「**OK**」を選択します。

h. 変更内容を保管してノードとの同期をとります。

- SchedulerCalendars アプリケーションをインストールするのが初めての場合は、以下の手順を実行します。

a. 「**アプリケーション**」 → 「**新規アプリケーションのインストール (Install New Application)**」を選択します。

b. ファイル・セレクターのウィンドウで、*install_root* ディレクトリーの `installableApps` サブディレクトリーを参照します。

c. 「**ScheduleCalendars.ear**」を選択します。

d. 「**次へ**」を選択します。

e. デフォルト値を確定して、もう一度「**次へ**」を選択します。

f. 「モジュールをサーバーにマップ (Map modules to servers)」のステップに到達するまでデフォルト値を確定し続け、次に ScheduleCalendars アプリケーションをロードするサーバーおよびクラスターを選択して、「**次へ**」を選択します。

g. まとめのステップで、「**完了**」を選択します。

h. アプリケーションがインストールを完了したら、「**マスター構成に保管 (Save to Master Configuration)**」を選択します。

i. 変更内容を保管して同期をとります。

5. Business Process Choreographer をアクティブにします。 109 ページの『Business Process Choreographer の活動化』を実行します。

6. **オプション:** Business Process Choreographer Explorer のインストールおよび構成が完了していない場合は、ここで構成できます。 99 ページの『Business Process Choreographer Explorer の構成』を実行します。

7. **オプション:** Business Process Choreographer Observer を使用したいけれど、ビジネス・プロセス・コンテナー・インストール・ウィザードのオプションを使用してそれをインストールしておらず、また bpeconfig.jacl スクリプト・ファイルをバッチ・モードで実行しなかった場合、スクリプトを実行することにより、Business Process Choreographer Observer をコマンド行からインストールできます。実行します。 101 ページの『Business Process Choreographer Observer インフラストラクチャーの構成』を実行します。

8. **オプション:** Business Process Choreographer が作動していることを確認します。 109 ページの『Business Process Choreographer の作動確認』を実行します。

Business Process Choreographer が構成され作動しています。

これで、ビジネス・プロセスまたはヒューマン・タスク、あるいはその両方を含むエンタープライズ・アプリケーションを実行できるようになりました。

bpeconfig.jacl スクリプト・ファイルを使用した Business Process Choreographer の構成

このスクリプト・ファイルは、Business Process Choreographer に必要なすべてのリソースを構成します。

目的

このスクリプトは、対話式に実行することも、バッチ・モードで実行することもできます。このスクリプトは、インストール・ウィザードや管理コンソールを使用せずに、作業ビジネス・プロセス・コンテナーおよびヒューマン・タスク・コンテナーを構成します。また、ローカル・データベースおよび必要なメッセージング・リソースを作成することができ、Business Process Choreographer Explorer を構成することもできます。クラスター内にあるアプリケーション・サーバーにスクリプトが適用された場合、そのクラスター内のすべてのサーバーが、Business Process Choreographer を使用するように構成されます。bpeconfig.jacl を実行する方法は、Business Process Choreographer を WebSphere Process Server for z/OS に合わせて構成するための推奨方法です。

場所

bpeconfig.jacl スクリプト・ファイルは、以下の Business Process Choreographer のサンプル・ディレクトリーにあります。*smpe_root*/ProcessChoreographer/config

制約事項

このスクリプトには、以下の制約事項があります。

ND 環境またはクラスター内

ND 環境またはクラスター内で複数のアプリケーション・サーバーを構成するには、bpeconfig.jacl スクリプトを対話式に実行する必要があります。非対話式に実行する場合は、このスクリプトを使用して、このタイプの構成を実行することはできません。

DB2 for z/OS データベースの使用

bpeconfig.jacl スクリプトで、DB2 for z/OS データベースを作成することはできません。手動で作成する必要があります。

データベースおよびストレージ・グループの作成については、WebSphere Process Server for z/OS のインストールと構成の資料を参照してください。

スタンドアロン・サーバー環境でのスクリプトの実行

スタンドアロン・サーバー環境の場合:

- -conntype NONE オプションを指定するのはアプリケーション・サーバーが稼働していない場合に限定します。
- サーバーが稼働していてグローバル・セキュリティーが使用可能な場合は、-username および -password オプションを指定します。
- デフォルト・プロファイルを構成していない場合は、-profileName オプションを追加します。

ND 環境でのスクリプトの実行

Network Deployment 環境の場合:

- bpeconfig.jacl スクリプトをデプロイメント・マネージャー・ノードで実行します。
- -conntype NONE オプションを指定するのはデプロイメント・マネージャーが稼働していない場合に限定します。
- グローバル・セキュリティーが使用可能な場合、-username および -password オプションを組み込んでください。
- デフォルト・プロファイルを構成していない場合は、-profileName オプションを追加します。

ビジネス・プロセス・コンテナー、Business Process Choreographer Explorer、および Business Process Choreographer Observer の非対話式の構成

コマンド行に必要なパラメーターを指定すると、そのパラメーターについてのプロンプトが出されなくなります。Business Process Choreographer を構成するには、現行ディレクトリーが *install_root*/ProcessChoreographer である場合、以下のコマンドを入力します。

```
bin/wsadmin.sh -f ProcessChoreographer/config/bpeconfig.jacl parameters
```

ここで、*parameters* は、以下のとおりです。

```
 -conntype NONE
 -user  userName
 -password  userPassword
 -profileName profileName
{-node nodeName -server serverName}
{-adminBFMUsers userList    | -adminBFMGroups groupList}
{-monitorBFMUsers userList | -monitorBFMGroups groupList}
 -jmsBFMRunAsUser userID
 -jmsBFMRunAsPwd password
{-adminHTMUsers userList    | -adminHTMGroups groupList}
{-monitorHTMUsers userList | -monitorHTMGroups groupList}
 -jmsHTMRunAsUser userID
 -jmsHTMRunAsPwd password
 -contextRootBFM contextRootBFM
 -contextRootHTM contextRootHTM
 -mailServerName mailServerName
 -mailUser mailUserID
 -mailPwd mailPassword
 -hostName explorerVirtualHostname
 -explorerHost explorerURL
 -remoteNodeName nodeName
 -remoteServerName serverName
 -remoteClusterName clusterName
 -contextRootExplorer explorerContextRoot
 -createDB { yes | no }
 -dbType databaseType
 -dbVersion version
 -dbHome databaseInstallPath
 -dbJava JDBCDriverPath
 -dbName databaseName
 -dbUser databaseUser
 -dbPwd databasePassword
 -dbAdmin databaseAdministratorUserID
 -driverType JDBCDriverType
 -dbTablespaceDir databaseTablespacePath
```

```
-dbServerName databaseServerName
-dbServerPort databaseServerPort
-dbStorageGroup DB2zOSStorageGroup
-dbSubSystem DB2zOSSubSystem
-dbSQLID DB2zOSSchemaQualifier
-dbInstance InformixInstance
-mqType JMSProviderType
-createQM { yes | no }
-qmNameGet getQueueManagerName
-mqClusterName appServerClusterName
-qmNamePut putQueueManagerName
-mqHome MQInstallationDirectory
-mqUser JMSProviderUserID
-mqPwd JMSProviderPassword
-mqSchemaName mqSchemaName
-mqCreateTables  { true | false }
-mqDataSource datasourceName
-shell shell
-createEventCollector { yes | no }
-createObserver { yes | no }
```

注: 上記の一部のパラメーターは、他のパラメーターに指定する値によってオプションになる場合があります。パラメーター間の依存関係と、パラメーターがオプションであるか必須であるかを判別する条件については、以下にパラメーターごとに説明します。必須パラメーターをコマンド行で指定しない場合は、対話式にプロンプトが出されます。

パラメーター

wsadmin を使用してスクリプトを起動する場合は、以下のパラメーターを使用できます。

conntype NONE
この指定によって、管理接続が使用不可になります。このオプションを指定するのは、アプリケーション・サーバー (スタンドアロンの場合) またはデプロイメント・マネージャー (ND 環境の場合) が稼働していない場合に限定してください。

user *userName*
グローバル・セキュリティーが有効になっている場合は、認証用のユーザー ID を指定する必要があります。

password *userPassword*
グローバル・セキュリティーが有効になっている場合は、ユーザー ID *userName* のパスワードを指定する必要があります。

profileName *profileName*
ここで *profileName* は、ユーザー定義プロファイルの名前です。 z/OS では、プロファイル名がデフォルトです。

node *nodeName*
nodeName は、Business Process Choreographer を構成するノードの名前です。ノードが 1 つだけでサーバーが 1 つしかない場合、このパラメーターはオプションです。

server *serverName*

serverName は、Business Process Choreographer を構成するサーバーの名前です。ノードが 1 つだけでサーバーが 1 つしかない場合、このパラメーターはオプションです。

adminBFMUsers *userList*

ここで、*userList* は、ユーザー・レジストリーから取得した、BPESystemAdministrator Java 2 Enterprise Edition (J2EE) ロールがマップされるユーザーの名前のリストです。分離文字は縦線 (|) です。ビジネス・プロセス・コンテナーをインストールするには、このプロパティーが必要です。このパラメーターにはデフォルト値はありません。adminBFMUsers オプションと adminBFMGroups オプションのいずれか一方または両方を設定する必要があります。

adminBFMGroups *groupList*

ここで *groupList* は、ユーザー・レジストリーから取得した、BPESystemAdministrator J2EE ロールがマップされるグループの名前のリストです。分離文字は縦線 (|) です。ビジネス・プロセス・コンテナーをインストールするには、このプロパティーが必要です。このパラメーターにはデフォルト値はありません。adminBFMUsers オプションと adminBFMGroups オプションのいずれか一方または両方を設定する必要があります。

monitorBFMUsers *userList*

ここで *userList* は、ユーザー・レジストリーから取得した、BPESystemMonitor J2EE ロールがマップされるユーザーの名前のリストです。分離文字は縦線 (|) です。ビジネス・プロセス・コンテナーをインストールするには、このプロパティーが必要です。このパラメーターにはデフォルト値はありません。monitorBFMUsers と monitorBFMGroups のいずれかまたは両方を設定する必要があります。

monitorBFMGroups *groupList*

ここで *groupList* は、ユーザー・レジストリーから取得した、BPESystemMonitor J2EE ロールがマップされるグループの名前のリストです。分離文字は縦線 (|) です。ビジネス・プロセス・コンテナーをインストールするには、このプロパティーが必要です。このパラメーターにはデフォルト値はありません。monitorBFMUsers と monitorBFMGroups のいずれかまたは両方を設定する必要があります。

jmsBFMRunAsUser *userID*

ここで *userID* は、ビジネス・プロセス・コンテナー JMS API のユーザー・レジストリーから取得した run-as ユーザー ID です。ビジネス・プロセス・コンテナーをインストールするには、このプロパティーが必要です。このパラメーターにはデフォルト値はありません。これは、設定する必要があります。

jmsBFMRunAsPwd *password*

ここで *password* は、ビジネス・プロセス・コンテナー JMS API のパスワードです。ビジネス・プロセス・コンテナーをインストールするには、このプロパティーが必要です。このパラメーターにはデフォルト値はありません。これは、設定する必要があります。

adminHTMUsers *userList*

ここで *userList* は、ユーザー・レジストリーから取得した、TaskSystemAdministrator Java 2 Enterprise Edition (J2EE) ロールがマップされる

ユーザーの名前のリストです。分離文字は縦線 (|) です。タスク・コンテナーをインストールするには、このプロパティーが必要です。このパラメーターにはデフォルト値はありません。adminHTMUsers オプションと adminHTMGroups オプションのいずれか一方または両方を設定する必要があります。

adminHTMGroups *groupList*
ここで *groupList* は、ユーザー・レジストリーから取得した、TaskSystemAdministrator J2EE ロールがマップされるグループの名前のリストです。分離文字は縦線 (|) です。タスク・コンテナーをインストールするには、このプロパティーが必要です。このパラメーターにはデフォルト値はありません。adminHTMUsers オプションと adminHTMGroups オプションのいずれか一方または両方を設定する必要があります。

monitorHTMUsers *userList*
ここで *userList* は、ユーザー・レジストリーから取得した、TaskSystemMonitor J2EE ロールがマップされるユーザーの名前のリストです。分離文字は縦線 (|) です。タスク・コンテナーをインストールするには、このプロパティーが必要です。このパラメーターにはデフォルト値はありません。monitorHTMUsers と monitorHTMGroups のいずれかまたは両方を設定する必要があります。

monitorHTMGroups *groupList*
ここで *groupList* は、ユーザー・レジストリーから取得した、TaskSystemMonitor J2EE ロールがマップされるグループの名前のリストです。分離文字は縦線 (|) です。タスク・コンテナーをインストールするには、このプロパティーが必要です。このパラメーターにはデフォルト値はありません。monitorHTMUsers と monitorHTMGroups のいずれかまたは両方を設定する必要があります。

jmsHTMRunAsUser *userID*
ここで *userID* は、ヒューマン・タスク・コンテナー JMS API のユーザー・レジストリーから取得した run-as ユーザー ID です。ヒューマン・タスク・コンテナーをインストールするには、このプロパティーが必要です。このパラメーターにはデフォルト値はありません。これは、設定する必要があります。

jmsHTMRunAsPwd *password*
ここで *password* は、ヒューマン・タスク・コンテナー JMS API のパスワードです。ヒューマン・タスク・コンテナーをインストールするには、このプロパティーが必要です。このパラメーターにはデフォルト値はありません。これは、設定する必要があります。

contextRootBFM *contextRootBFM*
ここで *contextRootBFM* は、Web サービス・エンドポイント URL のコンテキスト・ルートです。 Business Flow Manager (BFM) の場合、サーバー上のデフォルト・コンテキスト・ルートは `/BFMIF_${nodeName}_${serverName}` です。クラスター上のデフォルトは `/BFMIF_${clusterName}` です。これは、設定する必要があります。

contextRootHTM *contextRootHTM*
ここで *contextRootHTM* は、Web サービス・エンドポイント URL のコンテキスト・ルートです。 Human Task Manager (HTM) の場合、サーバー上のデフォルト・コンテキスト・ルートは `/HTMIF_${nodeName}_${serverName}` です。クラスター上のデフォルトは `/HTMIF_${clusterName}` です。これは、設定する必要があります。

mailServerName *mailServerName*

ここで *mailServerName* は、Human Task Manager が、通知メールの送信に使用するメール・サーバーのホスト名です。メール・セッションを構成する場合は、このパラメーターが必要です。このパラメーターを設定しない場合は、メール・セッション構成はスキップされます。デフォルト値は、ローカル・ホストの完全修飾ホスト名です。

mailUser *mailUserID*

ここで *mailUserID* は、メール・サーバーにアクセスするためのユーザー ID です。 Human Task Manager が通知メールを送信するためにメール・セッションを作成する必要があります。デフォルト値は空で、これは認証が不要な場合にのみ適切です。

mailPwd *mailPassword*

ここで *mailPassword* は、メール・サーバーにアクセスするためのパスワードです。 Human Task Manager が通知メールを送信するためにメール・セッションを作成する必要があります。

hostName *explorerVirtualHostname*

ここで *explorerVirtualHostname* は、Business Process Choreographer Explorer を実行する仮想ホストです。デフォルト値は `default_host` です。

explorerHost *explorerURL*

ここで *explorerURL* は、Business Process Choreographer Explorer の URL です。このパラメーターを非クラスター環境で指定しない場合は、デフォルト値が計算されます (例えば、http://localhost:9080)。このパラメーターの値は、Human Task Manager の EscalationMail.ClientDetailURL カスタム・プロパティーに使用されます。

precompileJSPs { yes | no }

Java Server Pages (JSPs) をプリコンパイルするかどうかを識別します。

remoteNodeName *nodeName*

ローカル側の Business Process Choreographer Explorer に接続しない場合は、このパラメーターと remoteServerName を使用します。node と server パラメーターまたは cluster パラメーターによって指定される Business Process Choreographer サーバーに接続する場合は、このパラメーターを使用しないでください。

remoteServerName *serverName*

ローカル側の Business Process Choreographer Explorer に接続しない場合は、このパラメーターと remoteNodeName を使用します。node と server パラメーターまたは cluster パラメーターによって指定される Business Process Choreographer サーバーに接続する場合は、このパラメーターを使用しないでください。

remoteClusterName *clusterName*

ローカル側の Business Process Choreographer Explorer に接続せず、remoteNodeName および remoteServerName を指定しない場合は、このパラメーターを使用します。node と server パラメーターまたは cluster パラメーターによって指定される Business Process Choreographer サーバーに接続する場合は、このパラメーターを使用しないでください。

contextRootExplorer *contextRootExplorer*

ここで *contextRootExplorer* は、Business Process Choreographer Explorer のコンテキスト・ルートです。デフォルト値は `/bpc` であり、 `http://host:port/contextRootExplorer` のデフォルト URL になります。コンテキスト・ルートは WebSphere セル内で固有のものでなければなりません。

createDB { yes | no }

指定可能な値は、`yes` または `no` です。 `yes` に設定すると、スクリプトはデータベースを作成します。z/OS データベースの場合、このスクリプトはデータベースを作成することはできず、テーブル・スペースおよびテーブルだけを作成できます。他のデータベース・タイプの場合、デフォルト値は `yes` です。

DB2 for z/OS 用のデータベースおよびストレージ・グループを作成する方法について詳しくは、『構成の計画』という表題のセクションで、『データベースおよびストレージ・グループの作成』を参照してください。

dbType *databaseType*

ここで *databaseType* は、データベース・タイプです。これは、ビジネス・プロセス・コンテナーのインストール、データベースまたはデータベース・テーブルの作成、およびデータ・ソースの作成を行う場合に必要です。デフォルト値はありません。指定可能な値は、以下のとおりです。

- Cloudscape
- zOS-DB2

dbVersion *DB2zOSversion*

ここで *DB2zOSversion* は、`7` または `8` のいずれかの値になります。このパラメーターは、データベース・タイプが DB2 for z/OS の場合のみに必須です。デフォルト値はありません。

dbHome *databaseInstallPath*

ここで *databaseInstallPath* は、データベース・システムのインストール・ディレクトリーです。この値は、使用する DB2 のバージョンと DB2 のインストール先によって異なります。

dbJava *JDBCDriverPath*

ここで *JDBCDriverPath* は、JDBC ドライバーがあるディレクトリーです。このパラメーターは、以下のようにデータベースとドライバー・タイプを組み合わせる場合にのみ必須です。

- DB2 または DB2 for z/OS とタイプ 4 ドライバー。デフォルト値は `${dbHome}/java` です。

dbName *databaseName*

ここで *databaseName* は、Business Process Choreographer データベースの名前です。これは、データベースまたはデータベース・テーブルの作成、およびデータ・ソースの作成に使用されます。デフォルト値は `BPEDB` です。

dbUser *databaseUser*

ここで *databaseUser* は、データベースにアクセスするためのユーザー ID です。これは、データベース・テーブルおよびデータ・ソースを作成するために使用されます。デフォルト値は、データベースおよびプラットフォームによって以下のように異なります。DB2 for z/OS の場合、デフォルト値は db2inst1 です。

dbPwd *databasePassword*

ここで *databasePassword* は、ユーザー ID *databaseUser* のパスワードです。

dbAdmin *databaseAdministratorUserID*

ここで *databaseAdministratorUserID* は、データベース管理者のユーザー ID です。これは、以下のデータベース・タイプ用のデータベースおよびデータベース・テーブルを作成する場合にのみ必須です。

- DB2 for z/OS の場合、デフォルトは `db2inst1` です。

driverType *JDBCDriverType*

ここで *JDBCDriverType* は、JDBC ドライバーのタイプです。これは、データ・ソースを作成するために使用されます。

- DB2 の場合、指定可能な値は `Universal` または `CLI` です。 これは、データベース・テーブルを作成する場合にも使用されます。

dbTablespaceDir *databaseTablespacePath*

ここで *databaseTablespacePath* は、データベース・テーブル・スペースが作成されるディレクトリーです。これは、データベースおよびデータベース・テーブルを作成するために使用されます。このパラメーターは、以下のデータベース・タイプの場合にのみ必須です。

- DB2 の場合、デフォルト値は空です。これは、テーブル・スペースが作成されないことを意味します。

dbServerName *databaseServerName*

ここで *databaseServerName* は、Business Process Choreographer 用のデータベースをホストするホスト名サーバーです。これは、データ・ソースを作成するために使用されます。Sybase の場合、これはデータベースを作成するためにも使用されます。

- DB2 の場合、デフォルト値は空です。
- 他のすべてのデータベース・タイプの場合、デフォルト値はローカル・ホストの完全修飾ホスト名です。

dbServerPort *databaseServerPort*

ここで *databaseServerPort* は、Business Process Choreographer 用のデータベース・サーバーの TCP/IP ポートです。このパラメーターは、dbServerName を指定する場合は必須です。DB2 の場合、デフォルト値は `50000` です。

dbStorageGroup *DB2zOSStorageGroup*

ここで *DB2zOSStorageGroup* は、Business Process Choreographer データベース・テーブルを作成するために使用されるストレージ・グループです。このパラメーターは、DB2 on z/OS の場合にのみ必須です。デフォルト値はなく、空にすることはできません。詳しくは、「データベースおよびストレージ・グループの作成」を参照してください。

dbSubSystem *DB2zOSSubSystem*

ここで *DB2zOSSubSystem* は、Business Process Choreographer データベース・テーブルおよびデータ・ソースを作成するために使用される DB2 サブシステムです。このパラメーターは、DB2 on z/OS の場合にのみ必須です。デフォルト値は `BPEDB` です。

dbSQLID *DB2zOSSchemaQualifier*

ここで、*DB2zOSSchemaQualifier* は、データベース・テーブルを作成するために

使用されるスキーマ修飾子です。このパラメーターは、DB2 on z/OS の場合にのみ必須です。デフォルト値はありません。この値は、空にすることができます。Universal JDBC ドライバー・タイプを使用する場合にのみ、値を指定します。

mqType *JMSProviderType*
ここで *JMSProviderType* は、Business Process Choreographer に使用する Java Message Service (JMS) プロバイダーのタイプです。これは、キュー・マネージャーおよびキュー、リスナー・ポートまたは ActivationSpecs、およびキュー接続ファクトリーを作成するために使用されます。

ここで *JMSProviderType* は、以下のいずれかの値です。

WPM デフォルト・メッセージングの場合 (WebSphere Platform Messaging)。このオプションは、常に有効です。

MQSeries
WebSphere MQ の場合。このオプションを使用する場合は、WebSphere MQ 製品がインストールされている必要があります。

MQSeries を mqType と指定すると、bpeconfig.sh ユーティリティーは createQueues.sh を実行します。

createQueues.sh を実行すると、その結果として /tmp/tmp_crt_ques.mqs というファイルが作成されます。 このファイルの内容は、WebSphere MQ 管理者に提出する必要がある WebSphere MQ 定義です。管理者は、WebSphere MQ を z/OS 上で構成するために、このファイルの内容を管理者の現在のジョブに追加できます。

createQM *{ yes | no})*
スクリプトがローカルの WebSphere MQ キュー・マネージャーを作成するかどうかを制御します。このオプションは、パラメーター mqType の値が MQSeries の場合にのみ有効です。このパラメーターのデフォルト値は yes です。スクリプトによって WebSphere MQ キュー・マネージャーを作成したくない場合、例えば、スクリプトを実行するサーバーとは別のサーバー上にキュー・マネージャーを作成する場合は、no の値を使用します。

qmNameGet *getQueueManagerName*
ここで *getQueueManagerName* は、GET 要求のキュー・マネージャーの名前です。これは、キュー・マネージャーとキュー、およびリスナー・ポートとキュー接続ファクトリーを作成するために使用されます。「-」文字を使用してはなりません。*getQueueManagerName* のデフォルト値は BPC_*nodeName_serverName* です。このオプションは、パラメーター mqType の値が MQSeries の場合にのみ有効です。

mqClusterName *appServerClusterName*
ここで *appServerClusterName* は、デフォルトの JMS プロバイダーのメッセージ・エンジンの作成先にする WebSphere Application Server クラスターの名前です。このことは、WebSphere MQ クラスターとは無関係です。 このオプションを使用するのは、Business Process Choreographer をクラスター内で構成し、mqType オプションを WPM に設定する場合に限られます。

qmNamePut *putQueueManagerName*
ここで *putQueueManagerName* は、PUT 要求のキュー・マネージャー名です。

これは、mqClusterName パラメーターを設定してある場合にのみ使用します。これは、キュー・マネージャーとキュー、およびリスナー・ポートとキュー接続ファクトリーを作成するために使用されます。「-」文字を使用することはできず、qmNameGet パラメーターに指定するキュー・マネージャー名と同じであってはなりません。*putQueueManagerName* のデフォルト値は BPCC_*nodeName_serverName* です。

mqHome *MQInstallationDirectory*
ここで *MQInstallationDirectory* は、WebSphere MQ のインストール・ディレクトリーです。これは、キュー・マネージャーとキュー (Windows システムのみ)、およびリスナー・ポートとキュー接続ファクトリーを作成するために使用されます。WebSphere 変数 MQ_INSTALL_ROOT を設定する場合は、その値が使用され、これは変更されません。このオプションは、パラメーター mqType の値が MQSeries の場合にのみ有効です。

MQ_INSTALL_ROOT を設定しない場合は、*MQInstallationDirectory* に使用されるデフォルト値は、プラットフォームによって以下のように異なります。

AIX: /usr/mqm

Solaris および HP-UX:
/opt/mqm

mqUser *JMSProviderUserID*
ここで *JMSProviderUserID* は、JMS プロバイダーにアクセスするためのユーザー ID です。

- mqType の値が WPM の場合、このパラメーターは ActivationSpecs および接続ファクトリーを作成するために使用されます。デフォルト値は現在のログオン・ユーザーです。
- mqType の値が MQSeries の場合、このパラメーターは非 Windows プラットフォーム上で、キュー・マネージャーおよびキューを作成するために使用されます。*JMSProviderUserID* のデフォルト値は、以下のとおりです。

  mqm

mqPwd *JMSProviderPassword*
ここで *JMSProviderPassword* は、mqUser に指定するユーザー ID のパスワードです。このパラメーターにはデフォルト値はありません。

mqSchemaName *mqSchemaName*
ここで *mqSchemaName* は、デフォルトの JMS プロバイダーのメッセージング・エンジン用のデータベース・スキーマの名前です。デフォルト値は BPEME です。 このオプションを使用するのは、Business Process Choreographer をクラスター内で構成し、mqType オプションを WPM に設定する場合に限られます。

mqCreateTables *{ true | false})*
このブール・パラメーターは、デフォルトの JMS プロバイダーが、最初の接続時にテーブルをメッセージ・エンジン・データベース内に自動的に作成するかどうかを制御します。デフォルト値は true です。 このオプションを使用するのは、Business Process Choreographer をクラスター内で構成し、mqType オプションを WPM に設定する場合に限られます。

mqDataSource *datasourceName*
ここで、*datasourceName* は、デフォルトの JMS プロバイダーのメッセージ・エンジンが使用するデータ・ソースの JNDI 名を表します。このデータ・ソースは、*mqClusterName* で識別される WebSphere クラスター内のクラスター・レベルのデータ・ソースである必要があります。デフォルトの JMS プロバイダーの基盤データベースは手動で作成する必要があります。 このオプションを使用するのは、Business Process Choreographer をクラスター内で構成し、mqType オプションを WPM に設定する場合に限られます。

shell *shell*
このパラメーターによって、外部コマンドを実行するために使用されるシェルが決まります。デフォルト値は `/bin/sh` です。

createEventCollector *{ yes | no}*
バッチ・モードで実行する場合、デフォルトは yes です。この設定では、Business Process Choreographer Event Collector アプリケーションが構成されます。これは、Business Process Choreographer Observer では必要な設定です。インストールしない場合は、このパラメーターの値を no に設定します。

createObserver *{ yes | no}*
バッチ・モードで実行する場合、デフォルトは yes です。この設定では、Business Process Choreographer Observer アプリケーションが構成されます。インストールしない場合は、このパラメーターの値を no に設定します。

構成スクリプトの対話式実行

この例では、bpeconfig.jacl スクリプトを実行して、既存の DB2 データベースを使用するビジネス・プロセス・コンテナー、ヒューマン・タスク・コンテナー、および Business Process Choreographer Explorer をインストールして構成する方法について説明します。

制約事項: 対話式に実行する場合、このスクリプトでは Business Process Choreographer Observer も、必要な Event Collector アプリケーションも構成できません。Business Process Choreographer Observer を使用する場合は、101 ページの『Business Process Choreographer Observer インフラストラクチャーの構成』を実行する必要があります。

1. サーバー上で、以下のコマンドを入力してスクリプトを開始します。ND 環境の場合は、デプロイメント・マネージャー上で行います。

```
install_root/bin/wsadmin.sh
      -f install_root/ProcessChoreographer/sample/bpeconfig.jacl
```

2. 表示される質問に対して以下のように対話式に応答します。
   a. ND 環境の場合は、構成先のクラスターが提供されます。 クラスターが正しくない場合は **No** を入力して次のクラスターを表示します。クラスターが正しい場合は、**Yes** と入力します。
   b. `Install the business process container?` という質問に対しては、**Yes** と入力します。
   c. `User(s) to add to role BPESystemAdministrator` という質問に対しては、ビジネス・プロセス管理者のロールを果たすユーザーのユーザー ID を入力します。

d. Group(s) to add to role BPESystemAdministrator という質問に対しては、ビジネス・プロセス管理者のロールにマップされているグループをドメイン・ユーザー・レジストリーから入力します。

e. User(s) to add to role BPESystemMonitor という質問に対しては、ビジネス・プロセス・モニターのロールを果たすユーザーのユーザー ID を入力します。

f. Group(s) to add to role BPESystemMonitor という質問に対しては、ビジネス・プロセス・モニターのロールにマップされているグループをドメイン・ユーザー・レジストリーから入力します。

g. Use WebSphere default messaging or WebSphere MQ という質問を受けた場合は、表示される 2 つのオプションの一方を入力します。

h. Run-as UserId for role JMSAPIUser という質問に対しては、JMSAPIUser のロールに使用される別名実行ユーザー ID を入力します。

i. 別名実行ユーザー ID のパスワードを入力します。

j. この例での Use a DB2, an Informix, an Oracle, or an SQL Server database [DB2/Informix/Oracle/MSSQL]? という質問に対しては、DB2 を入力します。他のデータベースを選択した場合は、他のデータベース固有の質問になります。

k. Use WebSphere default messaging or WebSphere MQ [WPM/MQSeries]? という質問に対しては、使用する JMS プロバイダーを選択します。

l. WebSphere Platform Messaging (WPM) を選択した場合、さらに以下のように入力してください。

   1) Virtual Host for the SCA Web Service [default_host]: という質問に対しては、**Enter** を押して、デフォルト値 default_host を受け入れます。

   2) Context root for the SCA Web Service [/BFMIF_PNODE_server1]: という質問に対しては、**Enter** を押して、デフォルト値 /BFMIF_PNODE_server1 を受け入れます。

m. Create the DataSource for the Process Choreographer database? という質問に対しては、**Yes** と入力します。

n. データベース名を入力してください。

o. Universal or CLI? という質問に対しては、JDBC ドライバーのタイプを入力します。

p. DB2 User ID という質問に対しては、データベース・テーブルおよびスキーマを作成するために使用したユーザー ID を入力します。

q. Database server name (may be empty, set to use the type 4 driver) という質問に対しては、データベースをホストするサーバーの名前を入力します。

r. Database server port という質問に対しては、データベース・サーバーのポート番号 (例: 50000) を入力します。

s. Create the Process Choreographer database? という質問に対しては、データベースを作成する十分な権限がユーザー ID にある場合は **Yes** と入力します。データベースが既に存在する場合、またはデータベースを作成する十分な権限がユーザー ID にない場合は **No** と入力します。

t. DB2 tablespace directory (may be empty) という質問に対しては、テーブル・スペースのディレクトリーを入力するか、空のままにしておきます。

u. Create the ActivationSpecs for the business flow manager? という質問に対しては、**Yes** または **No** と入力します。

v. User ID for access to default messaging という質問を受けた場合は、デフォルトの JMS プロバイダーにアクセスするときに使用するユーザー ID を入力します。

w. Name of the message engine cluster という質問を受けた場合は、メッセージ・エンジン・クラスターの名前を入力します。

x. Name of the message engine database schema という質問を受けた場合は、メッセージ・エンジン・データベース・スキーマの名前を入力します。

y. Automatically create the message engine database tables [true/false]? という質問を受けた場合、メッセージ・エンジン・データベース・テーブルを自動的に作成する場合は true を入力し、作成しない場合は false と入力します。

z. Message engine datasource JNDI name という質問を受けた場合は、メッセージ・エンジン・データ・ソースの JNDI 名を入力します。

aa. Install the task container? という質問に対しては、**Yes** と入力します。

ab. User(s) to add to role TaskSystemAdministrator という質問に対しては、タスク管理者のロールを果たすユーザーのユーザー ID を入力します。

ac. Group(s) to add to role TaskSystemAdministrator という質問に対しては、タスク管理者のロールにマップされているグループをドメイン・ユーザー・レジストリーから入力します。

ad. User(s) to add to role TaskSystemMonitor という質問に対しては、タスク・モニターのロールを果たすユーザーのユーザー ID を入力します。

ae. Run-as UserID for role EscalationUser という質問に対しては、エスカレーション・ユーザーのロールを果たすための別名実行ユーザー ID を入力します (例えば、db2admin)。

af. Context root for the SCA Web Service [/HTMIF_PNODE_server1]: という質問に対しては、Service Component Architecture (SCA) Web サーバーのコンテキスト・ルートを入力するか、または **Enter** を押して、デフォルト値を受け入れます。

ag. Create the mail notification session for the human task manager? という質問に対しては、Human Task Manager に対してメール通知セッションを作成しない場合、**No** と入力します。作成する場合は **Yes** を入力し、メール・トランスポート・ホストとユーザー ID を指定します。

ah. Create the ActivationSpecs for the human task manager? という質問に対しては、Human Task Manager のメッセージ駆動型 Bean (MDB) に対して J2EE ActivationSpecs を作成する場合は **Yes** と入力します。作成しない場合は **No** と入力します。

ai. Configure in cluster 'MECluster' [Yes/no]? という質問を受けた場合、指定したクラスターで構成する場合は **Yes** と入力します。そうでない場合は **No** と入力します。

aj. `Add JDBC provider permissions to server.policy [Yes/no]?` という質問を受けた場合、JDBC プロバイダーのアクセス権を `server.policy` ファイルに自動的に追加する場合は **Yes** と入力し、そうでない場合は **No** と入力します。

ak. `Install the Business Process Choreographer Explorer?` という質問に対しては、Business Process Choreographer Explorer をインストールする場合は **Yes** と入力し、次の `Virtual host for the Business Process Choreographer Explorer` には Business Process Choreographer Explorer の仮想ホストの名前 (例えば、**default_host**) を入力します。さらにその後の質問である `Precompile JSPs?` に対しては、Java Server Pages (JSP) をプリコンパイルする場合は **Yes** と入力します。そうでない場合は **No** と入力します。リモート Business Process Choreographer Explorer の場合、 `Node of Process Choreographer to connect to [PNODE]:` という質問に対しては、接続先の Business Process Choreographer ノードの名前を入力し、`Server of Process Choreographer to connect to [server1]:` という質問に対しては、接続先の Business Process Choreographer サーバーの名前を入力するか、**Enter** を押してデフォルトを受け入れます。

al. `Context root for the Business Process Choreographer Explorer [/bpc]:` という質問を受けた場合は、Business Process Choreographer Explorer のコンテキスト・ルートを入力するか、**Enter** を押してデフォルト値 `/bpc` を使用します。

am. `Create aliases for` *`your_server`* `in host` *`your_host`*`?`という質問に対しては、仮想ホストでサーバーの別名を作成する場合は**Yes** と入力し、そうでない場合は **No** と入力します。

an. さまざまな情報が表示されます。例えば、Business Process Choreographer Explorer の URL や、Business Process Choreographer Observer を構成するために使用できるスクリプト・ファイルの位置の記述が示されます。

ao. `Enable global security using the Local OS user registry?` という質問に対しては、ローカルのオペレーティング・システム・ユーザー・レジストリーを使用してグローバル・セキュリティーを使用可能にする場合は **Yes** と入力し、そうでない場合は **No** と入力します。

ap. `Server user ID` という質問に対しては、サーバー・ユーザー ID を入力します。

aq. `Enforce Java 2 security?` という質問に対しては、Java 2 セキュリティーを施行する場合は **Yes** と入力し、そうでない場合は **No** と入力します。

ar. `Set 'com.ibm.SOAP.loginUserid' in soap.client.props?` という質問に対しては、SOAP クライアント・プロパティーにログイン・ユーザー ID を設定する場合は **Yes** と入力し、そうでない場合は **No** と入力します。

as. `Delete the temporary directory?` という質問に対しては、指定した一時ディレクトリーを削除する場合は **Yes** と入力し、そうでない場合は **No** と入力します。

3. 問題がある場合は、ログ・ファイルを確認します。

ログ・ファイル

bpeconfig.jacl スクリプト・ファイルを使用して構成ファイルを作成しているときに問題が発生した場合は、以下のログ・ファイルを確認します。

- bpeconfig.log
- wsadmin.traceout

どちらのファイルも、各ユーザーのプロファイルのログ・ディレクトリー内にあります。

- *install_root*/profiles/*profileName*/logs/ ディレクトリー内

インストール・ウィザードを使用したビジネス・プロセス・コンテナーの構成

ここでは、必要なリソースを作成し、その後ビジネス・プロセス・コンテナーのインストール・ウィザードを実行する方法について説明します。

ビジネス・プロセスまたはヒューマン・タスクを含むアプリケーションを実行する前に、必要なリソースを構成してビジネス・プロセス・コンテナー・アプリケーションをインストールする必要があります。必要なリソースを構成するための推奨方法は、bpeconfig.jacl を実行する方法です。

注: インストール・ウィザードは、WebSphere リソースのみを構成します。インストール・ウィザードを使用してビジネス・プロセス・コンテナーを構成する場合は、データベース (Cloudscape または DB2 for z/OS) および WebSphere MQ キュー (WebSphere MQ を Java Message Service (JMS) プロバイダーとして使用する場合) を作成するために、それぞれに対応する手動の手順を実行する必要があります。

1. クラスター化 Business Process Choreographer セットアップを準備している場合は、以下の手順を実行します。
   a. クラスターを作成します。 「WebSphere Process Server の管理」PDF の『クラスター化環境の構築 (Creating a clustered environment)』の手順を実行します。
   b. クラスターに対してデフォルトの JMS メッセージング・プロバイダーを使用している場合は、以下の手順を実行します。
      1) 「WebSphere Process Server の管理」PDF の『Service Component Architecture アプリケーションをサポートするサーバーまたはクラスターの作成 (Preparing a server or cluster to support Service Component Architecture applications)』で説明されているように、クラスターがサービス・アプリケーションをサポートしていることを確認します。
      2) メッセージ・エンジンのデータ・ストア用のデータベースを作成します。Service Component Architecture (SCA) のメッセージ・エンジンに使用したのと同じデータベースを使用しても、別個のデータベースを使用しても構いません。WebSphere Process Server によって作成されたすべてのバス、つまり SCA システム・バス、SCA アプリケーション・バス、Common Event Infrastructure バス、および Business Process Choreographer バスに対して 1 つのメッセージング・データベースを使

用することをお勧めします。このデータベースは、メッセージ・エンジンのフェイルオーバー可用性を確保するために、メッセージ・エンジンのホストとして動作するクラスターのすべてのメンバーがアクセス可能である必要があります。

- JMS ユーザーが表の作成を許可されている場合、デフォルトのメッセージ・エンジンは、最初のアクセス時にデータベース内に必要な表を作成します。
- JMS ユーザーが表の作成を許可されていない場合は、デフォルトのメッセージング・プロバイダーがデータベースにアクセスを試行する前に表を作成してください。*install_root* ディレクトリーの bin サブディレクトリーにある sibDDLGenerator ユーティリティーを使用すると、表を作成するときに使用できる DDL ファイルを生成できます。

2. Business Process Choreographer のデータベースを作成します。 39 ページの『ビジネス・プロセス・コンテナー用データベースの作成』を実行します。
3. サーバーが始動していることと、十分な管理権限のあるユーザー ID で管理コンソールにログオンしていることを確認します。
4. 管理コンソールでは、ビジネス・プロセス・コンテナーのインストール先となるサーバーまたはクラスターを選択します。 以下のいずれかをクリックします。
   - 「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***serverName***」
   - 「**サーバー**」 → 「**クラスター**」 → 「***clusterName***」

   ここで、*serverName* または *clusterName* は、ビジネス・プロセス・コンテナーのインストール先となるアプリケーション・サーバーまたはクラスターの名前です。クラスターで任意のアプリケーション・サーバーを選択すると、そのクラスター内のすべてのアプリケーション・サーバーにビジネス・プロセス・コンテナーが同時にインストールされます。
5. ビジネス・プロセス・コンテナー設定に移動します。 「**構成**」タブの「**コンテナーの設定 (Container Settings)**」で、「**ビジネス・プロセス・コンテナーの設定**」を展開し、「**ビジネス・プロセス・コンテナー**」をクリックします。
6. ビジネス・プロセス・コンテナーがインストールされていないことを確認します。 ビジネス・プロセス・コンテナーが現在インストールされていないことを示すメッセージが表示されます。 ビジネス・プロセス・コンテナーが既にインストールされている場合は、インストール・ウィザードを開始する前に 117 ページの『第 3 章 Business Process Choreographer 構成の除去』を実行します。
7. インストール・ウィザードを開始します。 「**追加プロパティー**」セクションで、「**ビジネス・プロセス・コンテナーのインストール・ウィザード**」リンクをクリックします。
8. データベース構成を選択します (ウィザードのステップ 1)。
   a. 「**JDBC プロバイダー**」ドロップダウン・リストで、使用しているデータベース・システム、システム・バージョン、および Java Database Connectivity (JDBC) ドライバーのあるエントリーを選択します。 可能な場合は、インストール・ウィザードによってパラメーター・フィールドに適切なデフォルト値が表示されます。ただし、ブラウザーとプラットフォームの組み合わせに

よっては、デフォルト値は提供されません。その場合は、 44 ページの『ビジネス・プロセス・コンテナーのインストール・ウィザードの設定』で推奨値を参照してください。

b. 「**インプリメンテーション・クラス名**」では、JDBC ドライバー・インプリメンテーションに対して提供されるデフォルトのクラス名を使用します。

c. 「**クラスパス**」に、JDBC ドライバーを含む Java アーカイブ (JAR) または圧縮ファイルのロケーションを入力します。テキスト・フィールドに表示されたパス変数を使用するには、「**環境**」 → 「**WebSphere 変数の管理**」で、そのパス変数を定義する必要があります。

d. 「**データ・ソースのユーザー名**」は、データベースに接続し、データを変更する権限を持つユーザー ID でなければなりません。 ユーザー ID が、データベース内で表および索引を作成する権限を持つ場合、サービスまたはフィックスパックの適用後、データベース・スキーマが必要に応じて自動的に更新されます。これは、Cloudscape データベースでは必須ではありません。

e. データ・ソースのユーザー名の「**データ・ソースのパスワード**」を入力します。これは、Cloudscape データベースでは必須ではありません。

f. 「**カスタム・プロパティー**」フィールドには、選択したデータベースのデフォルト値が含まれています。

- デフォルトのディレクトリーに存在しない Cloudscape データベースを使用している場合は、カスタム・プロパティー **databaseName** の値を変更して、データベースの完全修飾場所を指定します。
- 場合によっては、いくつかの他のプロパティーを変更または追加する必要があります。詳細については、 ビジネス・プロセス・コンテナーのインストール・ウィザードの設定ページおよびデータベース・システムの製品資料を参照してください。

g. 「**次へ**」をクリックして、インストール・ウィザードで次のステップに移動します。

9. JMS プロバイダーとセキュリティーを選択します (ウィザードのステップ 2)。

a. 「**JMS プロバイダー**」のドロップダウン・リストで、使用するビジネス・プロセス・コンテナーのメッセージング・サービスを選択します。

- デフォルト・メッセージングの場合、「デフォルト・メッセージング・プロバイダー (`Default messaging provider`)」を選択します。
- WebSphere MQ の場合、`WebSphere MQ` を選択します。

b. 「**キュー・マネージャー**」のデフォルト値 (`BPC_`*`nodeName_serverName`*) を使用します。デフォルト・メッセージング・プロバイダーを使用している場合、このフィールドは無視されます。

c. WebSphere MQ JMS プロバイダーを使用しており、WebSphere 環境変数 `${MQ_INSTALL_ROOT}` を定義していない場合は、「**クラスパス**」 が WebSphere MQ Javalib ディレクトリーを指していることを確認します。デフォルトでは、`MQ_INSTALL_ROOT` は `${WAS_INSTALL_ROOT}/lib/WMQ` という値で事前定義されています。

d. 「**JMS ユーザー ID**」に、メッセージング・サービス用の管理権限を持つユーザー ID を入力します。`root` を使用します。

e. 「**JMS パスワード**」に、JMS ユーザー ID のパスワードを入力します。

f. 「**Web サービス・エンドポイント**」に、Web サービス API の Web サービス・エンドポイントを入力します。

g. 「**JMS API ユーザー ID**」に、ユーザー・レジストリーからユーザー ID を入力します。このユーザー ID は非同期 API 呼び出しを処理するときに使用します。

h. 「**JMS API パスワード**」に、JMS API ユーザー ID のパスワードを入力します。

i. 「**管理者のセキュリティー・ロール・マッピング**」に、Business Process Administrator のロールにマップする、ドメインのグループ (ユーザー・レジストリーで定義されている) を入力します。

j. 「**システム・モニター・セキュリティー・ロール・マッピング**」に、Business Process System Monitor のロールにマップするグループ (ユーザー・レジストリー内に存在) の名前を入力します。

k. 「**次へ**」をクリックして、インストール・ウィザードで次のステップに移動します。

10. JMS リソースと Business Process Choreographer Explorer を選択します (ウィザードのステップ 3)。 「**デフォルト値を使用して新規 JMS リソースを作成する**」を選択するか、次の手順を実行します。

a. 「**既存の JMS リソースを選択します**」を選択します。

b. 「**接続ファクトリー**」ドロップダウン・リストを使用して「**BPECF**」を選択します。

c. 「**内部キュー**」ドロップダウン・リストを使用して 「**BPEIntQueue**」を選択します。

d. 「**外部要求処理キュー**」ドロップダウン・リストを使用して、「**BPEApiQueue**」を選択します。

e. 「**保留キュー**」ドロップダウン・リストを使用して 「**BPEHldQueue**」を選択します。

f. 「**保存キュー**」ドロップダウン・リストを使用して 「**BPERetQueue**」を選択します。

11. **オプション: Business Process Choreographer Explorer** をインストールするには、チェック・ボックスをオンにします。インストールしない場合は、チェック・ボックスをオフにします。オプションで、コンテキスト・ルートを指定できます。複数の Business Process Choreographer Explorer を同じサーバーにインストールする場合、そのうちの多くても 1 つがデフォルトのコンテキスト・ルート `/bpc` を使用できます。

12. **オプション:** 監査ログを使用可能にするには、「**このコンテナーで実行するすべてのプロセスについて、監査ロギングを使用可能にします。**」を選択します。

13. **オプション:** Common Event Infrastructure を使用するには、「**このコンテナーで実行するすべてのプロセスについて、Common Event Infrastructure のロギングを使用可能にします。**」を選択します。

14. **オプション: Business Process Choreographer Observer** をインストールするには、チェック・ボックスをオンにします。インストールしない場合は、チェック・ボックスをオフにします。チェック・ボックスをオンにできない場合、「**このコンテナーで実行するすべてのプロセスについて、 Common Event**

Infrastructure のロギングを使用可能にします」が選択されていることを確認してください。「**JMS ユーザー ID**」に、ユーザー・レジストリーから、CEI バスへの接続に使用できるユーザー ID を入力します。「**JMS パスワード**」に、関連したパスワードを入力します。

15. 「**次へ**」をクリックして、要約を表示します (ウィザードのステップ 4)。
16. 要約ページが正しいことを確認してください。 要約には、どの外部リソースが必要かについての覚え書が含まれています。リソースをまだ作成していない場合でもビジネス・プロセス・コンテナーの構成は継続できますが、ビジネス・プロセス・コンテナーを活動化する前にそれらを作成する必要があります。要約ページを印刷すると、正しいリソースの作成に役立ちます。
    a. 修正する場合は、「**戻る**」をクリックします。
    b. ビジネス・プロセス・コンテナーをインストールしてそのリソースを定義するには、「**終了**」をクリックします。 「**インストール**」ページに進行状況が表示されます。
17. インストールが正常に実行されなかった場合は、問題の訂正に役立つエラー・メッセージを確認して、再度試行します。
18. インストールが正常に実行された場合は、「**マスター構成の保管 (Save Master Configuration)**」をクリックしてから「**保管**」をクリックします。
19. Business Process Choreographer をクラスター内で構成し、WebSphere MQ JMS プロバイダーを使用している場合は、 55 ページの『クラスター内での WebSphere MQ JMS リソースのカスタマイズ』を実行します。
20. アプリケーション・サーバーを再始動します。
21. ビジネス・プロセス・コンテナーが正常に開始したことを確認します。 管理コンソールで、「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」を選択し、BPEContainer_*scope* という名前のアプリケーションの状況が開始済みになっていることを確認します。アプリケーション・サーバーにビジネス・プロセス・コンテナーをインストールした場合は、*scope* は *nodeName_serverName* です。クラスターにビジネス・プロセス・コンテナーをインストールした場合は、*scope* はクラスター名です。

    Business Process Choreographer Explorer もインストールした場合、デフォルトのコンテキスト・ルート `/bpc` を使用していれば `BPCExplorer_scope` という名前のアプリケーションを実行していること、またデフォルトのコンテキスト・ルートを使用していなければ `BPCExplorer_scope_contextroot` という名前のアプリケーションを実行していることも表示されます。

    また、Business Process Choreographer Observer をインストールした場合、`BPCObserver_scope` および `BPCECollector_scope` という名前の 2 つのアプリケーションを実行していることも表示されます。

ビジネス・プロセス・コンテナーが構成されました。

『インストール・ウィザードを使用したヒューマン・タスク・コンテナーの構成』の手順で構成を続行してください。

ビジネス・プロセス・コンテナー用のキュー・マネージャーとキューの作成

このトピックでは、z/OS での WebSphere MQ キュー・マネージャーおよびキューの作成方法について説明します。

WebSphere MQ が既にインストールされている必要があります。

z/OS 上でキュー・マネージャーを作成する場合は、スクリプト記述は使用できません。 z/OS 上でのキュー・マネージャー (QMGR) およびリスナーの作成については、MQ 管理者に問い合わせてください。

WebSphere MQ を外部 Java Message Service (JMS) プロバイダーとして使用している場合は、キュー・マネージャーとキューを作成する必要があります。

1. キュー定義を作成します。 次のように入力します。

   ```
   cd install_root/ProcessChoreographer createQueues.sh z/OS
   ```

   このコマンドは、ファイル `/tmp/tmp_crt_ques.mqs` を作成します。
2. Tailor `/tmp/tmp_crt_ques.mqs`

   `/tmp/tmp_crt_ques.mqs` の内容は、使用するサイト固有の標準に合致するように、適宜編集してください。

ファイル `/tmp/tmp_crt_ques.mqs` は WebSphere MQ キューおよびキュー・マネージャーの定義に関連しているため、このファイルの内容は、サイト固有の標準になっています。

MQ 管理者は、WebSphere MQ z/OS コマンドを JCL スクリプトに使用することにより、**tmp_crt_ques.mqs** を使用して WebSphere MQ リソースを構成できます。

『Business Process Choreographer のデータベースを作成する』の手順で構成を続行してください。

ビジネス・プロセス・コンテナー用のクラスター化キュー・マネージャーとキューの作成

WebSphere MQ クラスターを使用して Business Process Choreographer の WebSphere クラスター・セットアップを作成する場合は、キュー・マネージャー、クラスター、リポジトリー、チャネル、およびリスナーを作成する必要があります。

注: createQueues.sh を実行した場合に生成されるのは、WebSphere MQ のサンプル定義だけです。サンプル定義の使用方法については、WebSphere MQ 管理者に問い合わせてください。

1. 各ノードで、次のアクションを実行します。
   a. ご使用のユーザー ID に WebSphere MQ キューを作成する権限があることを確認します。
   b. get および put キュー・マネージャーを作成し、それらを WebSphere MQ クラスターのメンバーにします。

```
cd install_root/ProcessChoreographer/config
createQueues.sh getQueueManager clusterName putQueueManagerName
```

クラスター化キュー・マネージャーおよびキューをビジネス・プロセス・コンテナー用に作成するときに必要なパラメーターについては、『bpeconfig.jacl スクリプト・ファイルを使用した Business Process Choreographer の構成』を参照してください。

キュー・マネージャーが既に存在する場合は、それらが使用されます。キュー・マネージャーが存在しない場合は、作成され、使用されます。

c. WebSphere MQ コマンド・プロセッサーを開始します。

d. 複雑なセットアップの場合は、次の MQ コマンドを入力して、キュー・マネージャーのリモート管理を使用可能にすることをお勧めします。

```
DEFINE CHANNEL('SYSTEM.ADMIN.SVRCONN') TYPE(CHLTYPE)
```

e. このキュー・マネージャーが WebSphere MQ クラスターのリポジトリーである場合は、MQ コマンドを入力します。

```
ALTER QMGR REPOS('clusterName') REPOSNL(' ')
```

f. 次の MQ コマンドを入力して、このサーバーでホストされていない各リポジトリーに対してキュー・マネージャーの送信側および受信側チャネルを定義します。 各クラスター受信側チャネルに対して:

```
DEFINE CHANNEL('TO.repositoryQueueManager.TCP') +
     CHLTYPE(CLUSRCVR) +
     CLUSTER('clusterName') +
     CLUSNL(' ') +
     CONNAME('repositoryIP-Address(port)') +
     DESCR('Cluster receiver channel at repositoryQueueManager TCPIP') +
     MAXMSGL(4194304) +
     TRPTYPE(TCP) +
     MCAUSER('principal') +
     REPLACE
```

各クラスター送信側チャネルに対して:

```
DEFINE CHANNEL('TO.repositoryQueueManager.TCP') +
     CHLTYPE(CLUSSDR) +
     CONNAME('repositoryIP-Address(port)') +
     CLUSTER('clusterName') +
     CLUSNL(' ') +
     DESCR('Cluster sender channel to repositoryQueueManager TCPIP') +
     MAXMSGL(4194304) +
     TRPTYPE(TCP) +
     MCAUSER('targetPrincipal') +
     REPLACE +
     NPMSPEED (NORMAL)
```

各部の意味は、次のとおりです。

repositoryQueueManager
: リポジトリーをホストするキュー・マネージャーの名前。

clusterName
: すべてのキュー・マネージャーがメンバーになっている WebSphere MQ クラスターの名前。

repositoryIP-Address
: リポジトリー・キュー・マネージャーがあるノードの IP アドレス。

port　リポジトリー・キュー・マネージャーが使用している IP ポート。

principal、*targetPrincipal*

受信および送信チャネルに使用する MCAUSER。この値についての詳細は、WebSphere MQ の資料を参照してください。

g. 各キュー・マネージャーごとに、MQ コマンドを入力してリスナーを開始します。

2. サーバー上のチャネルの状況を確認するには、次の MQ コマンドを入力します。

```
display chstatus(*)
```

キュー・マネージャー、キュー、クラスター、リポジトリー、チャネル、およびリスナーが作成されました。

ビジネス・プロセス・コンテナー用データベースの作成

ビジネス・プロセス・コンテナーにはデータベースが必要です。このトピックでは、Business Process Choreographer 用のデータベースの作成方法を説明します。

Business Process Choreographer のクラスター化セットアップでは、1 つのデータベースが、WebSphere クラスター内のすべてのビジネス・プロセス・コンテナーに対応します。非クラスター化セットアップでは、データベースは、1 つのアプリケーション・サーバーのビジネス・プロセス・コンテナー専用になります。

1. データベースをホストするサーバーで、使用しているデータベース・システムの説明に従って、データベースを作成します。
   - 40 ページの『Business Process Choreographer 用の Cloudscape データベースの作成』.
   - 42 ページの『Business Process Choreographer 用の DB2 (z/OS 用) データベースの作成』.
2. ローカル・データベースなしで Business Process Choreographer を実行する各サーバーでは、リモート・データベースをアクセス可能にする必要があります。
   a. 適切なデータベース・クライアントまたは Java Database Connectivity (JDBC) ドライバーをアプリケーション・サーバーをホストするサーバーにインストールします。
   b. タイプ 4 JDBC Driver を使用していない場合は、データベース・クライアントに既知の新規データベースを作成する必要があります。

      Cloudscape の場合

      Business Process Choreographer は、リモート・アクセスをサポートしない組み込みバージョンの Cloudscape だけをサポートしているので、アクションは必要ありません。 Cloudscape Network Server は、XA サポートがないため、サポートされません。

      DB2 Universal Database™ の場合

      データベースはカタログされ、別名でアクセス可能です。

   c. データベースへの接続をテストします。
      1) 「**リソース**」 → 「**JDBC プロバイダー**」をクリックします。
      2) 必要に応じて別の有効範囲を選択し、「**適用**」をクリックします。クラスター化された Business Process Choreographer 構成の場合は、データ・ソ

ースをクラスター・レベルまたはサーバー・レベルで定義できます。非クラスター化構成の場合、データ・ソースはサーバー・レベルで定義します。

3) ***provider_name*** → 「**データ・ソース**」をクリックします。
4) 適切なデータ・ソース `BPEDataSource`*`database`* を探します。例えば、`BPEDataSourceCloudscape` などです。
5) データ・ソースのチェック・ボックスを選択して、「**テスト接続**」をクリックします。
6) テスト接続が正常であったことを示すメッセージが表示されます。

Business Process Choreographer データベースが作成され、アプリケーション・サーバーおよびデプロイメント・マネージャーをホストするサーバーからアクセスできるようになりました。

ステップ 3 (33 ページ) で構成を続行してください。

Business Process Choreographer 用の Cloudscape データベースの作成

このタスクを使用して、Business Process Choreographer 用の Cloudscape データベースを作成します。

Cloudscape データベース・システムは Java 言語にインプリメントされています。これは、いくつかの Java Archive (JAR) ファイルとして WebSphere Process Server に同梱されています。

WebSphere Process Server に同梱されている Cloudscape ライセンスは、開発とテストのためだけのものであり、実動用ではありません。Network Deployment 環境で Cloudscape を Business Process Choreographer のデータベース・システムとして使用することはできません。この製品に同梱されている Cloudscape バージョンには、Distributed Relational Database Architecture™ (DRDA®) プロトコルでのクライアント/サーバー JDBC アクセスをサポートする Cloudscape Network Server が含まれています。本バージョンの WebSphere Process Server に付属の Cloudscape Network Server のバージョンでは XA をサポートしていないので、Business Process Choreographer はリモート側ではアクセスできない組み込み Cloudscape バージョンのみ使用できます。

`BPEDB` という Cloudscape データベースを作成するには、次のアクションを実行します。

1. 以下のいずれかの操作を実行して、データベース作成スクリプト・ファイルを実行する準備を整えます。
   - デフォルト・ロケーションでデータベースを作成できるようにするには、適切なプロファイル・ディレクトリーに `databases` サブディレクトリーを手動で作成します。 *`install_root`*`/profiles/`*`Profile_name`*`/databases` を作成します。新規ディレクトリーに移動します。
   - デフォルト・ロケーションと異なるデータベース・ロケーションを作成するには、新規データベースを作成したいディレクトリーに移動します。ビジネス・プロセス・コンテナーのインストール・ウィザードを実行する場合は必ず、デ

ータベースの完全修飾ロケーションを、カスタム・プロパティー databaseName の値として指定する必要があります。

2. データベース作成スクリプトを現行ディレクトリーにコピーします。 *install_root*/ProcessChoreographer/Cloudscape/createDatabase.sql ファイルをコピーします。
3. 使用サーバーに Java が構成済みであるかどうかを確認します。 次のコマンドを入力します。

   ```
   java -version
   ```

   エラー・メッセージが表示された場合は、ステップ 5 でデータベース作成スクリプトを実行するときに、Java コマンドの前に、Java 実行可能ファイルまでの絶対パスを接頭部として記述する必要があります。*install_root*/java/bin/ というパスを追加します。
4. データベース作成スクリプト createDatabase.sql のヘッダーに記載されている説明をエディターを使用して読みます。 サンプル・ファイルは、ASCII フォーマットで提供されます。 このファイルの表示、編集、および実行に使用するツールの機能によっては、ファイルを読み取り可能なフォーマット (例: EBCDIC) に変換することが必要になる場合があります。例えば、以下に示すように、viascii (viascii createDatabase.sql) を使用して ASCII フォーマットで直接編集し、次に iconv を使用してファイルを EBCDIC に変換することにより、vi を使用できます。

   ```
   iconv –t IBM-1047 –f ISO8859-1 createDatabase.sql >
   createDatabase_EBCDIC.sql
   ```
5. データベース作成スクリプト・ファイルを実行します。 次のように入力します。

   ```
   java -Djava.ext.dirs=install_root/cloudscape/lib
        -Dij.protocol=jdbc:db2j: com.ibm.db2j.tools.ij
        install_root/ProcessChoreographer/Cloudscape/createDatabase.sql
   ```
6. Business Process Choreographer Observer でもこのデータベースを使用する場合は、次の手順を実行します。
   a. 以下の SQL スクリプトをデータベース・サーバーにコピーします。

      ```
      clearSchema_Observer.sql
      createDatabase_Observer.sql
      createSchema_Observer.sql
      dropSchema_Observer.sql
      ```

      - SQL ファイルは *install_root*/dbscripts/ProcessChoreographer/Cloudscape/ にあります。

   b. テキスト・エディターで、スクリプト・ファイル createSchema_Observer.sql のヘッダーに記載されている説明を読み取ります。 メモ帳を使用しないでください。メモ帳では、このファイルを正しく読める形式で表示できません。
   c. スキーマを作成します。 データベースを作成したディレクトリーから、スクリプトのヘッダーの説明に従って、スクリプト・ファイル createSchema_Observer.sql を実行します。 サンプル・ファイルは、ASCII フォーマットで提供されます。 このファイルの表示、編集、および実行に使用するツールの機能によっては、ファイルを読み取り可能なフォーマット (例: EBCDIC) に変換することが必要になる場合があります。例えば、viascii

(viascii createSchema_Observer.sql) を使用して ASCII フォーマットで直接編集し、次に iconv を使用してファイルを EBCDIC に変換できます。

d. エラーの場合は、スクリプト・ファイル `dropSchema_Observer.sql` を実行してスキーマを除去できます。

Business Process Choreographer 用のデータベースが作成されます。

ステップ 2 で構成を継続してください。

Business Process Choreographer 用の DB2 (z/OS 用) データベースの作成

このタスクを使用して、Business Process Choreographer 用の DB2 for z/OS データベースを作成します。

WebSphere Process Server for z/OS での DB2 Universal JDBC ドライバーのサポートについては、DB2 Universal JDBC Driver のサポートを参照してください。

DB2 for z/OS を使用する場合は、以下の更新が必要となる場合があります。

- Business Process Choreographer LOB をサポートするよう DB2 構成パラメーター (zParms) を増やす必要があります。
  - _LOBVALA
  - _LOBVALS
- 必要な DB2 変換サービスは、以下のとおりです。
  - CONVERSION 367,1208,ER;
  - CONVERSION 1208,367,ER;

このトピックでは、DB2 for z/OS データベースを作成する方法、およびアプリケーション・サーバーをホストするサーバーが、そのデータベースにアクセス可能であることを確認する方法について説明します。

1. WebSphere Process Server は、 z/OS サーバーに既にインストールされているはずです。
2. データベースをホストする z/OS サーバーで、以下のようにします。
   a. ネイティブ z/OS 環境にログオンします。
   b. 複数の DB2 システムがインストールされている場合は、使用するサブシステムを決定します。
   c. DB2 サブシステムが listen している IP ポートを書き留めます。
   d. DB2 管理メニューを使用して、新しいデータベースを (例えば、`BPEDB` という名前で) 作成します。データベースの名前を書き留めます。
   e. ストレージ・グループを作成し、その名前を書き留めます。
   f. WebSphere Process Server を実行しているリモート・サーバーから、データベースに接続するために使用するユーザー ID を決定します。通常、セキュリティー上の理由により、このユーザー ID はデータベースを作成するために使用したものと同じではありません。
   g. データベースおよびストレージ・グループにアクセスする権限をユーザー ID に付与します。このユーザー ID には、データベースの新しいテーブルを作成する権限も必要です。

h. 接続中のユーザー ID のスキーマにテーブルおよびビューを作成するのか、スキーマ修飾子 (_SQLID) をカスタマイズするのかを決定します。単一のユーザー ID を使用して、同じ名前の複数のテーブルを持つ複数のデータベースにアクセスする場合は、名前の衝突を避けるために別のスキーマ修飾子を使用する必要があります。

3. WebSphere Process Server をホストするサーバーで、以下のようにします。

a. 以下の情報に注意してください。

DB2 for z/OS と Linux、UNIX、および Windows 用 DB2 の間に存在する重要な違い。 Linux、UNIX、および Windows 用 DB2 にはサブシステムの概念はありませんが、DB2 for z/OS にはあります。 DB2 for z/OS はサブシステムで実行されるため、catalog node および catalog database コマンドは、適切なサブシステムを識別しなければならないことを理解し、データベース名とサブシステム名を混同しないようにすることが重要です。DB2 for Linux、UNIX、および Windows では、サブシステム名は既知の概念ではないため、カタログ・コマンドのリンク先のデータベース名は、DB2 for z/OS サブシステムの実際の名前です。

b. アプリケーション・サーバーをホストするサーバーで、データベース・システム用の Business Process Choreographer 構成スクリプトがあるディレクトリーに移動します。

- Windows システムでは、ご使用の DB2 のバージョンに応じて、次のコマンドのいずれかを入力します。

```
cd install_root¥dbscripts¥ProcessChoreographer¥DB2zOSV7
cd install_root¥dbscripts¥ProcessChoreographer¥DB2zOSV8
```

- UNIX および Linux® システムでは、ご使用の DB2 のバージョンに応じて、次のコマンドのいずれかを入力します。

```
cd install_root/dbscripts/ProcessChoreographer/DB2zOSV7
cd install_root/dbscripts/ProcessChoreographer/DB2zOSV8
```

c. createTablespace.sql スクリプトを編集します。@STOGRP@ をストレージ・グループ名で置き換え、@DBNAME@ をデータベース名 (サブシステム名ではない) で置き換えます。

d. カスタマイズした createTablespace.sql スクリプトを、スクリプトのヘッダーに記述されている内容に従って実行します。 テーブル・スペースを除去する場合は、dropTablespace.sql スクリプトを使用します。

e. createSchema.sql スクリプトを編集します。

1) @STOGRP@ をストレージ・グループ名で置き換えます。

2) @DBNAME@ をデータベース名 (サブシステム名ではない) で置き換えます。

3) @_SQLID@ をスキーマ修飾子で置き換えるか、@_SQLID@ (後ろのドットも含む) をスクリプトから除去します。カスタム・スキーマ修飾子は DB2 Universal JDBC ドライバーでのみ使用できます。また、カスタム・スキーマ修飾子を使用するには、構成 customSQLID プロパティーを適切な値に設定しておく必要があります。

f. カスタマイズした createSchema.sql スクリプトを、スクリプトのヘッダーに記述されている内容に従って実行します。 このスクリプトが機能しない場

合、またはテーブルおよびビューを除去する場合は、dropSchema.sql スクリプトを使用してスキーマをドロップしますが、スクリプトを実行する前に @_SQLID@ を置き換えてください。

Business Process Choreographer 用のデータベースが作成されます。

注: SQL 定義が提供されていますが、これらの定義を DB2 環境に手動で追加する必要があります。

ビジネス・プロセス・コンテナーのインストール・ウィザードの設定

ビジネス・プロセス・コンテナーをインストールして構成するには、インストール・ウィザードを使用します。

ビジネス・プロセス・コンテナーのインストール・ウィザードを利用するには、**「サーバー」** → **「アプリケーション・サーバー」** → ***server_name*** → **「ビジネス・プロセス・コンテナーの設定」** → **「ビジネス・プロセス・コンテナー」** → **「ビジネス・プロセス・コンテナーのインストール・ウィザード」**をクリックします。このページでは、インストール・ウィザードのフィールドについて、ウィザードに表示される順に説明します。

ステップ 1 データベースの構成

- JDBC プロバイダー
- インプリメンテーション・クラス名
- クラスパス (JDBC プロバイダー)
- データ・ソースのユーザー名
- データ・ソースのパスワード
- カスタム・プロパティー

ステップ 2 JMS プロバイダーおよびセキュリティー

- JMS プロバイダー
- キュー・マネージャー
- クラスパス (JMS プロバイダー)
- JMS ユーザー ID (JMS プロバイダー)
- JMS パスワード (JMS プロバイダー)
- Web サービス・エンドポイントのコンテキスト・ルート
- JMS API ユーザー ID
- JMS API パスワード
- 管理者セキュリティー・ロールのマッピング
- システム・モニター・セキュリティー・ロールのマッピング

ステップ 3 Business Process Choreographer Explorer とロギング

- Business Process Choreographer Explorer のインストール
- コンテキスト・ルート

- CEI ロギングを使用可能にする
- 監査ログを使用可能にする
- Business Process Choreographer Observer のインストール
- JMS ユーザー ID (Observer 用)
- JMS パスワード (Observer 用)

重要: コンテナーの構成後に変更できるのは、ロギング・オプション、再試行限度、保存キュー・メッセージ限度だけです。それ以外の値を変更する場合は、既存の Business Process Choreographer 構成を削除してから新たに構成を作成する必要があります。

JDBC プロバイダー

Business Process Choreographer でのみ使用される新規データ・ソースを作成する必要があります。JDBC プロバイダーを選択すると、「インプリメンテーション・クラス名」フィールドに適切なデフォルト値が挿入されます。

タイプ	**値**
必須	はい
データ型	ドロップダウン・リスト
z/OS での選択項目	z/OS 用の新規 XA データ・ソースを作成します。 • Cloudscape 5.1 (Cloudscape JDBC プロバイダー (XA)) • z/OS 上の DB2 Universal JDBC ドライバー (タイプ 2) **注:** このドライバーがドロップダウン・リストから選択できない場合は、 42 ページの『Business Process Choreographer 用の DB2 (z/OS 用) データベースの作成』の説明に従って、このデータ・ソースを構成します。

インプリメンテーション・クラス名

Java Database Connectivity (JDBC) ドライバー・インプリメンテーションの Java クラス名。JDBC プロバイダーを選択すると、このフィールドに適切なデフォルト値が挿入されます。その値を変更する必要はありません。

タイプ	**値**
必須	はい
データ型	ストリング
Cloudscape 5.1 のデフォルト (Cloudscape JDBC プロバイダー (XA))	`com.ibm.db2j.jdbc.DB2jXADataSource`
z/OS 上の DB2 Universal JDBC ドライバー (タイプ 2) のデフォルト	`com.ibm.db2.jcc.DB2ConnectionPoolDataSource`

データベースのプロパティーおよび設定についての詳細は、『ベンダー固有データ・ソースで最低限必要な設定』を参照してください。

クラスパス (JDBC プロバイダー)

Java Database Connectivity (JDBC) ドライバーを含む Java アーカイブ (JAR) ファイルまたは ZIP ファイルのパス。 JDBC ドライバーは、データ・ソース・インプリメンテーション・クラスを提供します。データベースがリモート・データベースである場合、このパスはクライアント・コンピューター上で JDBC ドライバーがインストールされている場所を示します。

タイプ
値

必須
Cloudscape の場合
いいえ。JDBC ドライバーは既に WebSphere クラスパスにあります。
z/OS 上の DB2 Universal JDBC ドライバーの場合 はい

データ型
ストリング

Cloudscape 5.1 のデフォルト
`${CLOUDSCAPE_JDBC_DRIVER_PATH}/db2j.jar`

`${CLOUDSCAPE_JDBC_DRIVER_PATH}` の値は定義済みであるため、設定する必要はありません。

z/OS 7 上の DB2 Universal JDBC ドライバー (タイプ 2) のデフォルト
```
${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar
${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar
${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cisuz.jar
```

`${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar` の値は、対応する DB2 クライアントまたは DB2 Connect のインストール・ルート・ディレクトリーによって異なり、**「環境」>「WebSphere 変数の管理 (Manage WebSphere Variables)」**で明示的に設定する必要があります。標準的な値は以下のとおりです。

z/OS の場合:
`/home/db2inst1/sqllib/java`

データ・ソースのユーザー名

データベースに接続し、データを変更する権限を持つユーザー ID。ユーザー ID が、データベース内で表および索引を作成する権限を持つ場合、サービスまたはフィックスパックの適用後、データベース・スキーマが必要に応じて自動的に更新されます。

タイプ
値

必須
Cloudscape の場合
いいえ
z/OS 上の DB2 Universal JDBC ドライバーの場合 はい

データ型
ストリング

デフォルト
現在管理コンソールにログオンしているユーザー ID。

データ・ソースのパスワード

データ・ソースのユーザー ID のパスワード。

タイプ	**値**
必須	**Cloudscape の場合** いいえ **z/OS 上の DB2 Universal JDBC ドライバーの場合** はい
データ型	ストリング
デフォルト	なし

カスタム・プロパティー

データベース・システムに必要な追加のパラメーター。

注意:
ビジネス・プロセス・コンテナーを構成してコンテナーが機能していることを確認する前にいずれかのオプション・プロパティーを変更することはお勧めしません。こうした変更作業は高度なシステム調整およびトラブルシューティングの範疇に入り、システムの機能が停止する恐れがあります。

タイプ	**値**
必須	はい
データ型	ストリング
データ・フォーマット	複数行の *Property=Value*
最低限必要なプロパティー	『ベンダー固有データ・ソースで最低限必要な設定』を参照してください。
この表に記載されていないプロパティー	オプション・プロパティーや無視されるプロパティーは、この表には記載されていません。そのようなプロパティーについては、使用している JDBC プロバイダーの資料を参照してください。
必須プロパティー	JDBC プロバイダーごとのすべての必須プロパティーについては、以下で説明します。
Cloudscape の必須プロパティー	**databaseName = ${USER_INSTALL_ROOT}/ databases/BPEDB** 必須のストリング。アクセスするデータベースを定義します。値は、完全修飾パスでなければなりません。 **要確認:** ウィザードを実行後、databaseName で指定された場所にデータベースを作成します。

タイプ	値
DB2 z/OS 7 (DB2 Universal JDBC ドライバー・プロバイダー) のプロパティー	**databaseName=BPEDB** 必須のストリング。DB2 UDB の場合、アクセスするデータベースを定義します。DB2 z/OS の場合、DB2 z/OS データベースを含むサブシステムを定義します。 **driverType=2** 必須の整数。データ・ソースの JDBC 接続タイプです。許可値は 2 のみです。 **serverName=""** オプションのストリング。DRDA サーバーの TCP/IP アドレスまたはホスト名です。 **portNumber=50000** オプションの整数。DRDA サーバーが常駐する TCP/IP ポート番号です。 **enableSQLJ=false** オプションのブール値。この値は、このデータ・ソースで SQLJ 操作を実行できるかどうかを指定する場合に使用します。有効にした場合は、JDBC 呼び出しと SQLJ 呼び出しの両方に対してこのデータ・ソースを使用できます。そうでない場合は、JDBC を使用した方法のみが許可されます。 **description=DataSource for Business Process Choreographer** オプションのストリング。データ・ソースの説明。データ・ソース・オブジェクトは使用しない。通知目的専用で使用されます。 **fullyMaterializeLobData=true** オプションのブール値。これを設定すると、LOB データの取り出しに LOB ロケーターを使用するかどうかを制御できます。 有効にすると LOB データはストリーム化されなくなりますが、ユーザーが LOB 列でストリームを要求すると、LOB データはロケーターによって実体化できます。デフォルト値は true です。 **resultSetHoldability=2** オプションの整数。トランザクションをコミットするときに ResultSets を閉じるか、開いたままにしておくかを指定します。可能な値は、1 (HOLD_CURSORS_OVER_COMMIT) または 2 (CLOSE_CURSORS_AT_COMMIT) です。 **currentPackageSet=""** オプションのストリング。このプロパティーは DB2Binder コレクション・オプションとともに使用します (DB2Binder コレクション・オプションは、インストール中に JDBC/CLI パッケージ・セットをバインドするときに DBA によって指定されます)。 **readOnly=false** オプションのブール値。このプロパティーは読み取り専用の接続を作成します。 **deferPrepares=false** オプションのブール値。このプロパティーは、ドライバーの入力データ型変換機能の内部セマンティクスに影響するパフォーマンス・ディレクティブを提供します。このプロパティーを「true」に設定すると、Universal ドライバーは「内部準備要求」を据え置きます。この場合、ドライバーは、実行時まで記述されたパラメーターまたは結果セット・メタデータがない状態で動作します。このため、記述されていない入力データは、入力に対するデータ型の相互変換を行わずに「そのまま」サーバーに送信されます。 **currentSchema=""** オプションのストリング。動的準備済み SQL ステートメントで適用可能な場合に修飾されていないデータベース・オブジェクト参照を修飾するためのデフォルト・スキーマ名を示します。 currentSchema を使用しない場合、デフォルト・スキーマ名は現行セッション・ユーザーの許可 ID です。 **cliSchema=""** オプションのストリング。データベース・メタデータ・カタログ照会を実行するときに検索する DB2 シャドー・カタログ・テーブルまたはビューのスキーマを指定します。 **enableMultithreadedAccessDetection=false** オプションのブール値。`true` に設定すると、接続およびそれに対応する Statement、ResultSet、および MetaData へのマルチスレッド化されたアクセスが自動的に検出されます。 **retrieveMessagesFromServerOnGetMessage=true** true に設定すると、すべての呼び出しが標準の JDBC SQLException.getMessage() に転送され、サーバー・サイド・ストアード・プロシージャーが呼び出され、エラーに対する読みやすいメッセージ・テキストが取得されます。 **preTestSQLString=SELECT 1 FROM TABLE1** オプションのストリング。この SQL ステートメントは事前テスト接続機能のために使用します。`j2c.properties` ファイルで事前テスト接続が使用可能になっている場合は、接続に対してこの SQL ステートメントを実行し、接続が良好であることを確認します。このフィールドがブランクの場合は、デフォルトの SQL ステートメントである SELECT 1 FROM TABLE1 を実行時に使用します。このとき、テーブル TABLE1 がデータベースに定義されていない場合は、例外処理のために実行が遅くなります。パフォーマンスを改善するには、ユーザーが独自の SQL ステートメントを提供することをお勧めします。

JMS プロバイダー

ビジネス・プロセス・コンテナーが使用するメッセージング・サービスを指定します。

タイプ	値
必須	はい
データ型	ドロップダウン・リスト
選択項目	デフォルト・メッセージング・プロバイダー (Default messaging provider) WebSphere MQ

キュー・マネージャー

ビジネス・プロセス・コンテナーが使用するキュー・マネージャーの名前。

タイプ	値
必須	**WebSphere MQ JMS プロバイダー**を選択した場合。そうでない場合は、このフィールドは使用不可です。
データ型	ストリング
値	キュー・マネージャーの名前 (例えば BPC_*nodeName_serverName*)。

クラスパス (JMS プロバイダー)

MQ Java lib ディレクトリーへのパス。

タイプ	値
必須	Websphere MQ インストールのルート・ディレクトリーを指す WebSphere 環境変数 ${MQ_INSTALL_ROOT} が未定義の場合。
使用可能	WebSphere MQ JMS プロバイダーを選択した場合。そうでない場合は、このフィールドは使用不可です。
データ型	ストリング
デフォルト	クラスパスのデフォルト値は、ローカルの MQ のインストールによって異なります。 **z/OS の場合** /opt/mqm/java/lib

JMS ユーザー ID

Java Message Service (JMS) プロバイダーとの接続を認証するために使用されます。このユーザー ID には、メッセージング・サービスのための管理権限がある必要があります。

タイプ	値
必須	はい
データ型	ストリング
制約事項	WebSphere デフォルト・メッセージングを使用している場合は、JMS ユーザー ID は 12 文字以下にする必要があります。
デフォルト	管理コンソールにログインするために使用したユーザー ID。
z/OS の場合	root を使用します。このユーザー ID は、グループ mqm のメンバーである必要があります。

JMS パスワード

Java Message Service (JMS) ユーザー ID のパスワード。

タイプ	**値**
必須	**WebSphere JMS プロバイダー**を選択した場合。そうでない場合は、このフィールドは使用不可です。
データ型	ストリング
デフォルト	なし

Web サービス・エンドポイント・コンテキスト・ルート

Web サービスに使用するルート・コンテキスト。

タイプ	**値**
必須	はい
データ型	ストリング
サーバーでの構成時デフォルト	/BFMIF_${*nodeName*}_${*serverName*}
クラスターでの構成時デフォルト	/BFMIF_${*clusterName*}

JMS API ユーザー ID

ビジネス・プロセス・コンテナーのメッセージ駆動型 Bean (MDB) が非同期 API 呼び出しを処理するときに使用するユーザー ID。

タイプ	**値**
必須	はい
データ型	ストリング

タイプ
値

説明
WebSphere のセキュリティーを使用可能にした場合は、JMS API を使用しなくても、有効なユーザー ID を指定する必要があります。この ID には特殊な権限は必要ありません。

WebSphere セキュリティーを使用可能にした際に JMS API を使用する場合、このユーザー ID は、プロセスをモデル化する際に適切な権限を付与された ID であるか、より一般的なケースとして、モデル化時に必要な権限を付与されたグループのメンバーであるかのいずれかでなければなりません。プロセスと関連付けられる担当者権限として考えられるものには、管理者、読者、およびスターターがあります。アクティビティーに関しては、ユーザー ID が、関連付けられた receiveEvent の潜在的な所有者である場合、そのユーザー ID は sendEvent アクションのみを実行することができます。

プロセス上のすべてのアクションを JMS API によってサポートする場合、J2EE のロール BPESystemAdministrator のメンバーであるユーザー ID を指定することができます。ただし、実動システムにおいては、より細分化されたセキュリティーのアプローチを取ることをお勧めします。

RACF セキュリティーを使用したロールの設定: この RACF 許可は、以下のセキュリティー・フィールドを指定した場合に適用されます。

- `com.ibm.security.SAF.delegation= true`

```
RDEFINE EJBROLE JMSAPIUser  UACC(NONE)
APPLDATA(' userid')
```

JMS API パスワード

JMS API ユーザー ID のパスワード。

タイプ
値

必須
WebSphere のセキュリティーが使用可能な場合 (JMS API を使用しない場合にも必要です)。

データ型
ストリング

管理者セキュリティー・ロールのマッピング

ビジネス・プロセス管理者のロールにマップされる、ドメインのユーザー・レジストリーからのグループ。

タイプ
値

必須
はい

データ型
ストリング

デフォルト
なし

制約事項
指定されたグループは、使用されるドメイン・ユーザー・レジストリーに既に存在している必要があります。ローカル・オペレーティング・システム、Lightweight Directory Access Protocol (LDAP)、またはカスタム・レジストリーをユーザー・レジストリーにすることができます。

RACF セキュリティーを使用したロールの設定: この RACF 許可は、以下のセキュリティー・フィールドを指定した場合に適用されます。

- `com.ibm.security.SAF.authorization=true`

```
RDEFINE EJBROLE BPESystemAdministrator  UACC(NONE)
PERMIT BPESystemAdministrator CLASS(EJBROLE)
ID(userid) ACCESS(READ)
```

システム・モニター・セキュリティー・ロールのマッピング

ビジネス・プロセス・モニターのロールにマップされる、ドメインのユーザー・レジストリーからのグループ。

タイプ
値

必須
はい

データ型
ストリング

デフォルト
なし

制約事項
指定されたグループは、使用されるドメイン・ユーザー・レジストリーに既に存在している必要があります。ローカル・オペレーティング・システム、Lightweight Directory Access Protocol (LDAP)、またはカスタム・レジストリーをユーザー・レジストリーにすることができます。

RACF セキュリティーによるロールの設定: この RACF 許可は、以下のセキュリティー・フィールドを指定した場合に適用されます。

- `com.ibm.security.SAF.authorization=true`

```
RDEFINE EJBROLE BPESystemMonitor  UACC(NONE)
PERMIT BPESystemMonitor  CLASS(EJBROLE)
ID(userid) ACCESS(READ)
```

Business Process Choreographer Explorer

このチェック・ボックスが選択されている場合、Business Process Choreographer Explorer もインストールされます。

タイプ	**値**
データ型	チェック・ボックス
デフォルト	選択

コンテキスト・ルート

このコンテキスト・ルートは、Business Process Choreographer Explorer の URL の一部になります。

タイプ	**値**
データ型	ストリング
デフォルト	/bpc
制約事項	Business Process Choreographer Explorer の複数インスタンスを構成する場合、各インスタンスは WebSphere セルで一意であるルート・コンテキストを持つ必要があります。

監査ログを使用可能にする

監査ログを、使用可能または使用不可にすることができます。

タイプ	**値**
データ型	チェック・ボックス
デフォルト	非選択状態

Common Event Infrastructure ロギングの使用可能化

Common Event Infrastructure (CEI) ロギングを使用可能または使用不可にすることができます。

タイプ	**値**
データ型	チェック・ボックス
デフォルト	非選択状態

Business Process Choreographer Observer

このチェック・ボックスが選択されている場合は、Business Process Choreographer Observer と Event Collector もインストールされます。

タイプ	**値**
データ型	チェック・ボックス
必須	いいえ
デフォルト	選択されません (Business Process Choreographer Observer はインストールされません)
依存関係	このオプションを選択する前に、CEI ロギングを使用可能にする必要があります。

JMS ユーザー ID (Observer 用)

Business Process Choreographer Observer が CEI バスへの認証に使用するユーザー ID。

タイプ	**値**
必須	Business Process Choreographer Observer をインストールする場合
データ型	ストリング

JMS パスワード (Observer 用)

JMS API ユーザー ID のパスワード。

タイプ	**値**
必須	Business Process Choreographer Observer をインストールする場合
データ型	ストリング

ビジネス・プロセス・コンテナーの設定

このパネルを使用して、ビジネス・プロセス・コンテナーを管理します。

ビジネス・プロセス・コンテナーは、アプリケーション・サーバー内でビジネス・プロセスを実行するためのサービスを提供します。この管理コンソール・ページを表示するには、**「サーバー」** → **「アプリケーション・サーバー」** → ***server_name*** → **「コンテナーの設定 (Container Settings)」** → **「ビジネス・プロセス・コンテナー」**をクリックします。

注: このパネルで行った変更は、再始動するまでサーバーに影響を及ぼしません。

Common Event Infrastructure ロギングの使用可能化

Common Event Infrastructure (CEI) ロギングを使用可能または使用不可にすることができます。

タイプ	**値**
データ型	チェック・ボックス
デフォルト	非選択状態

監査ログを使用可能にする

監査ログを、使用可能または使用不可にすることができます。

タイプ	**値**
データ型	チェック・ボックス
デフォルト	非選択状態

再試行限度

メッセージ処理の最大試行回数を指定します。この限度に達すると、メッセージは「未処理メッセージのリスナー・ポート」に送信されます。

タイプ	値
データ型	整数
デフォルト	5
範囲	2 から 10 (推奨)

保存キュー・メッセージ限度

保存キューに保管できるメッセージの最大数。この限度に達すると、メッセージは「内部メッセージのキュー」に再度送信され、プロセス・コンテナーが静止モードに切り替わります。

タイプ	値
データ型	整数
デフォルト	20

保存キュー

その時点で処理できず、後から再試行されるメッセージを含むキューの JNDI 名。

タイプ	値
データ型	読み取り専用ストリング
デフォルト	jms/BPERetQueue

保留キュー

再試行回数以上、処理を失敗したメッセージを保持するキューの JNDI 名。

タイプ	値
データ型	読み取り専用ストリング
デフォルト	jms/BPEHldQueue

クラスター内での WebSphere MQ JMS リソースのカスタマイズ

クラスター内にあるビジネス・プロセス・コンテナーの接続ファクトリー・リソースをカスタマイズして WebSphere MQ JMS プロバイダーを使用するには、このタスクを使用します。

デフォルトのメッセージングを使用している場合は、このタスクを実行しないでください。WebSphere MQ JMS プロバイダーを使用している場合は、クラスター内のアプリケーション・サーバーごとに以下の手順を実行します。

1. 接続ファクトリー・ページを次のようにして開きます。 **「リソース」** → **「JMS プロバイダー」** → **「WebSphere MQ」** → **「スコープ: サーバー」** → **「適用」** → **「WebSphere MQ 接続ファクトリー」**をクリックします。
2. ビジネス・プロセス・コンテナーの接続ファクトリー **BPECF** を選択し、使用しているキュー・マネージャー構成のタイプに対してプロパティー値を設定します。
   - 中央キュー・マネージャーの場合:

プロパティー	説明
ホスト	中央キュー・マネージャーをホストしているサーバーのホスト名。

プロパティー	説明
ポート	中央キュー・マネージャーが使用しているポート番号。
トランスポート・タイプ	Client
クライアント ID	使用するメッセージ・チャネル・エージェント (MCA) ユーザー ID。これは、一般にキュー・マネージャーの所有者または作成者です。通常 これは root ユーザーです。
CCSID	値 819 を使用します。 を使用します。

- キュー・マネージャーのクラスターの場合:

プロパティー	説明
トランスポート・タイプ	Bindings または Client
キュー・マネージャー	サーバーの get キュー・マネージャーの名前。

WebSphere MQ を使用する場合、ローカルのバインディング・トランスポート・タイプを使用すると、クライアント・トランスポート・タイプを使用した場合より処理が若干高速になりますが、ローカルの WebSphere MQ キュー・マネージャーを停止するために、アプリケーション・サーバー全体を停止する必要があるという影響があります。 Client を指定する場合は、get キュー・マネージャーのホスト名とポート番号も提供する必要があります。

3. ビジネス・プロセス・コンテナーの接続ファクトリー **BPECFC** を選択し、使用しているキュー・マネージャー構成のタイプに対してプロパティー値を設定します。
   - 中央キュー・マネージャーの場合:

プロパティー	説明
ホスト	中央キュー・マネージャーをホストしているサーバーのホスト名。
ポート	中央キュー・マネージャーが使用しているポート番号。
トランスポート・タイプ	Client
クライアント ID	使用するメッセージ・チャネル・エージェント (MCA) ユーザー ID。これは、一般にキュー・マネージャーの所有者または作成者です。通常これは root ユーザーです。
CCSID	値 819 を使用します。

   - キュー・マネージャーのクラスターの場合:

プロパティー	説明
ホスト	アプリケーション・サーバー・ノードのホスト名。
ポート	このアプリケーション・サーバーの put キュー・マネージャーによって使用されるポート番号。
トランスポート・タイプ	Client
クライアント ID	使用するメッセージ・チャネル・エージェント (MCA) ユーザー ID。これは、一般にキュー・マネージャーの所有者または作成者です。通常これは root ユーザーです。
CCSID	819

4. ヒューマン・タスク・マネージャーの接続ファクトリー **HTMCF** を選択し、使用しているキュー・マネージャー構成のタイプに対してプロパティー値を設定します。
   - 中央キュー・マネージャーの場合:

プロパティー	説明
ホスト	中央キュー・マネージャーをホストしているサーバーのホスト名。
ポート	中央キュー・マネージャーが使用しているポート番号。
トランスポート・タイプ	`Client`
クライアント ID	使用するメッセージ・チャネル・エージェント (MCA) ユーザー ID。これは、一般にキュー・マネージャーの所有者または作成者です。通常 これは root ユーザーです。
CCSID	値 `819` を使用します。 を使用します。

   - キュー・マネージャーのクラスターの場合:

プロパティー	説明
トランスポート・タイプ	`Bindings` または `Client`
キュー・マネージャー	サーバーの get キュー・マネージャーの名前。

   WebSphere MQ を使用する場合、ローカルのバインディング・トランスポート・タイプを使用すると、クライアント・トランスポート・タイプを使用した場合より処理が若干高速になりますが、ローカルの WebSphere MQ キュー・マネージャーを停止するために、アプリケーション・サーバー全体を停止する必要があるという影響があります。 `Client` を指定する場合は、get キュー・マネージャーのホスト名とポート番号も提供する必要があります。

ビジネス・プロセス・コンテナーの接続ファクトリーがクラスターにインストールされ、構成されました。

ステップ 20 (36 ページ) で構成を続行してください。

インストール・ウィザードを使用したヒューマン・タスク・コンテナーの構成

このタスクを使用して、ヒューマン・タスク・コンテナーを構成します。

ヒューマン・タスク・コンテナーを構成する前に、 15 ページの『第 2 章 Business Process Choreographer の構成』にある、ビジネス・プロセス・コンテナーの手動構成について参照してください。

bpeconfig.jacl スクリプトを実行済みの場合、ヒューマン・タスク・コンテナーは既に構成されています。 以下の手順では、インストール・ウィザードを使用してヒューマン・タスク・コンテナーを構成する方法について説明します。

1. 管理コンソールでは、ビジネス・プロセス・コンテナーのインストール先となるサーバーまたはクラスターを選択します。 以下のいずれかをクリックします。

- 「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***serverName***」
- 「**サーバー**」 → 「**クラスター**」 → 「***clusterName***」

ここで、*serverName* または *clusterName* は、ヒューマン・タスク・コンテナーのインストール先となるアプリケーション・サーバーまたはクラスターの名前です。

2. 「**コンテナーの設定 (Container Settings)**」セクションで、「**ヒューマン・タスク・コンテナーの設定**」 → 「**ヒューマン・タスク・コンテナー**」 → 「**ヒューマン・タスク・コンテナーのインストール・ウィザード**」 (「**追加プロパティー**」セクション) をクリックします。 可能な場合は、インストール・ウィザードによってパラメーター・フィールドに適切なデフォルト値が表示されます。60 ページの『ヒューマン・タスク・コンテナーのインストール・ウィザードの設定』で推奨値を参照できます。
3. ヒューマン・タスク・コンテナーが構成されていないことを確認します。 Human Task Manager が現在インストールされていないことを示すメッセージが表示されます。ヒューマン・タスク・コンテナーが既に構成されている場合は、インストール・ウィザードを開始する前に構成を除去します。構成を除去する方法の詳細については、Business Process Choreographer 構成の除去を参照してください。
4. JMS プロバイダーとセキュリティー設定を選択します (ステップ 1)。
   a. 「**JMS プロバイダー**」のドロップダウン・リストで、ビジネス・プロセス・コンテナーが使用するメッセージング・サービスを選択します。
      - デフォルト・メッセージングの場合、「デフォルト・メッセージング・プロバイダー (`Default messaging provider`)」を選択します。
      - WebSphere MQ の場合、`WebSphere MQ` を選択します。
   b. 「**キュー・マネージャー**」のデフォルト値 (BPC_*nodeName_serverName*) を使用します。デフォルト・メッセージング・プロバイダーを使用している場合、このフィールドは無視されます。
   c. 外部メッセージング (WebSphere MQ JMS プロバイダー) を使用し、WebSphere 環境変数 `${MQ_INSTALL_ROOT}` を定義していない場合は、**Classpath** が WebSphere MQ Java `lib` ディレクトリーを指していることを確認します。
   d. 「**JMS ユーザー ID**」に、メッセージング・サービス用の管理権限を持つユーザー ID を入力します。このユーザー ID は、JMS キュー・マネージャーに接続するときに使用します。`root` を使用します。
   e. 「**JMS パスワード**」に、JMS ユーザー ID のパスワードを入力します。
   f. 「**Web サービス・エンドポイント**」に、Web サービス API の Web サービス・エンドポイントを入力します。
   g. 「**エスカレーション・ユーザー ID**」では、スケジュール済みアクションを実行するためにヒューマン・タスク・コンテナーが使用するユーザー ID を入力します。アクションの例としては、エスカレーションを起動して、予想されるタスク状態、時刻指定されたタスクの削除、およびタスクの期限切れを確認することなどが挙げられます。`root` を使用します。
   h. 「**エスカレーション・パスワード**」に、エスカレーション・ユーザー ID のパスワードを入力します。

i. **「管理者のセキュリティー・ロール・マッピング」**に、Business Process Administrator のロールにマップする、ドメインのグループ (ユーザー・レジストリーで定義されている) を入力します。 例えば Windows システムの場合は、管理者グループを指定できます。

j. **「システム・モニター・セキュリティー・ロール・マッピング」**に、Business Process System Monitor のロールにマップするグループ (ユーザー・レジストリー内に存在) の名前を入力します。 例えば Windows システムの場合は、管理者グループを指定できます。

k. インストール・ウィザードで**「次へ」**をクリックして、ステップ 2 に移動します。

5. **オプション: 「メール・セッション」**を選択し、セルの有効範囲を持ったデフォルトのメール・セッション・リソースを作成します。
   - サーバー上にヒューマン・タスク・コンテナーを構成する場合、デフォルトのメール・セッション名は `mail/HTMNotification_nodeName_serverName` になります。
   - クラスター上にヒューマン・タスク・コンテナーを構成する場合、デフォルトのメール・セッション名は `mail/HTMNotification_clusterName` になります。

   重要: これを設定していないと、エスカレーション・メールは送信されません。
6. **オプション:** Common Event Infrastructure を使用するには、**「Common Event Infrastructure のロギングを使用可能にします」**を選択します。
7. **オプション:** 監査ログを使用可能にするには、**「監査ロギングを使用可能にします」**を選択します。
8. **「次へ」**をクリックして、要約を表示します (ステップ 3)。
9. 要約ページが正しいことを確認してください。 要約には、どの外部リソースが必要かについての覚え書が含まれています。リソースをまだ作成していない場合でもヒューマン・タスク・コンテナーの構成は継続できますが、ヒューマン・タスク・コンテナーを活動化する前にそれらを作成する必要があります。要約ページを印刷すると、正しいリソースの作成に役立ちます。
   a. 修正する場合は、**「戻る」**をクリックします。
   b. ヒューマン・タスク・コンテナーをインストールしてそのリソースを定義するには、**「終了」**をクリックします。 「インストール」ページに進行状況が表示されます。
   c. エラー・メッセージが表示されていないことを確認します。
10. ステップ 5 で「メール・セッション」オプションを選択した場合は、次の手順に従ってメール・トランスポート・ホストを設定する必要があります。
    a. **「リソース」 → 「メール・プロバイダー」**をクリックします。
    b. セル・スコープの**「組み込みメール・プロバイダー」**を選択します。
    c. **「メール・セッション」**の下で、**「HTMMailSession_*scope*」 (ここで *scope* はクラスター名、またはノードおよびサーバー名から構成)**をクリックし、次に**「メール・トランスポート・ホスト」**を設定します。

d. メール・トランスポート・ホストが保護されている場合は、「**メール・トランスポートのユーザー ID**」および「**メール・トランスポートのパスワード**」も設定します。

e. 「**OK**」をクリックします。

11. 「**マスター構成の保管 (Save Master Configuration)**」をクリックしてから、「**保管**」をクリックします。

12. アプリケーション・サーバーを再始動します。

13. コンテナーが正常にインストールされなかった場合は、問題の訂正に役立つエラー・メッセージを確認して、このタスクを繰り返します。

    管理コンソールまたは SystemOut.log ファイルを調べて、アプリケーション・サーバーについてチェックします。クラスターでは、クラスター内のすべてのアプリケーション・サーバーのログをチェックします。

ヒューマン・タスク・コンテナーが構成されました。

ステップ 3 (16 ページ) で構成を続行してください。

ヒューマン・タスク・コンテナーのインストール・ウィザードの設定

ヒューマン・タスク・コンテナーをインストールして構成するには、インストール・ウィザードを使用します。

ヒューマン・タスク・コンテナー・インストール・ウィザードを利用するには、「**サーバー**」 → 「**アプリケーション・サーバー**」 → ***server_name*** をクリックします。次に「**コンテナーの設定 (Container Settings)**」セクションで、「**ヒューマン・タスク・コンテナーの設定**」 → 「**ヒューマン・タスク・コンテナー**」 → 「**ヒューマン・タスク・コンテナーのインストール・ウィザード**」をクリックします。このページでは、インストール・ウィザードのフィールドについて、ウィザードに表示される順に説明します。

ステップ 1 JMS プロバイダーおよびセキュリティー

- JMS プロバイダー
- キュー・マネージャー
- クラスパス
- JMS ユーザー ID (JMS プロバイダー)
- JMS パスワード (JMS プロバイダー)
- Web サービス・エンドポイント・コンテキスト・ルート
- エスカレーション・ユーザー ID
- エスカレーション・パスワード
- 管理者セキュリティー・ロールのマッピング
- システム・モニター・セキュリティー・ロールのマッピング

ステップ 2 メール・セッションとロギング:

- メール・セッション
- CEI ロギングを使用可能にする

- 監査ログを使用可能にする

重要: これらのフィールドを適用後は、ロギング・オプションを使用可能および使用不可にすることのみ可能です。

JMS プロバイダー

ヒューマン・タスク・コンテナーが使用するメッセージング・サービスを指定します。

タイプ	**値**
必須	はい
データ型	ドロップダウン・リスト
選択項目	WebSphere MQ デフォルト・メッセージング・プロバイダー (Default messaging provider)
デフォルト	デフォルト・メッセージング・プロバイダー (Default messaging provider)

キュー・マネージャー

ヒューマン・タスク・コンテナーが使用するキュー・マネージャーの名前。

タイプ	**値**
必須	**WebSphere MQ JMS プロバイダー**を選択した場合。そうでない場合は、このフィールドは使用不可です。
データ型	ストリング
値	キュー・マネージャーの名前 (例えば BPC_*nodeName_serverName*)。

クラスパス

MQ Java lib ディレクトリーへのパス。

タイプ	**値**
必須	Websphere MQ インストールのルート・ディレクトリーを指す WebSphere 環境変数 MQ_INSTALL_ROOT が未定義の場合。
使用可能	WebSphere MQ JMS プロバイダーを選択した場合。そうでない場合は、このフィールドは使用不可です。
データ型	ストリング
デフォルト	クラスパスのデフォルト値は、ローカルの MQ のインストールによって異なります。 **z/OS の場合** /opt/mqm/java/lib

JMS ユーザー ID

Java Message Service (JMS) プロバイダーとの接続を認証するために使用されます。このユーザー ID には、メッセージング・サービスのための管理権限がある必要があります。このユーザー ID は、JMS キュー・マネージャーに接続するときに使用します。

タイプ	値
必須	はい
データ型	ストリング
制約事項	WebSphere デフォルト・メッセージングを使用している場合は、JMS ユーザー ID は 12 文字以下にする必要があります。
デフォルト	管理コンソールにログインするために使用したユーザー ID。
z/OS の場合	root を使用します。このユーザー ID は、グループ mqm のメンバーである必要があります。

JMS パスワード

Java Message Service (JMS) ユーザー ID のパスワード。

タイプ	値
必須	**WebSphere JMS プロバイダー**を選択した場合。そうでない場合は、このフィールドは使用不可です。
データ型	ストリング
デフォルト	なし

Web サービス・エンドポイント・コンテキスト・ルート

Web サービスに使用するルート・コンテキスト。

タイプ	値
必須	はい
データ型	ストリング
サーバーでの構成時デフォルト	/HTMIF_*nodeName*}_*serverName*
クラスターでの構成時デフォルト	/HTMIF_*clusterName*

ここで入力されたコンテキスト・ルートは、Web サービス・エンドポイントの以下の URL に組み込まれます。http://*host*:*port*/*contextRoot*/sca/com/ibm/task/api/sca/HTMWS

エスカレーション・ユーザー ID

スケジュール済みの処置を実行するためにヒューマン・タスク・コンテナーで使用するユーザー ID。

タイプ	値
必須	エスカレーション・メールが送信されない場合でも必須です。
データ型	ストリング
説明	これは、Human Task Manager のメッセージ駆動型 Bean (MDB) がスケジュール済みのアクション (エスカレーション、削除、有効期限満了) を実行するための run-as ユーザー ID です。

エスカレーション・パスワード

エスカレーション・ユーザー ID のパスワード。

タイプ	**値**
必須	はい
データ型	ストリング

管理者セキュリティー・ロールのマッピング

ユーザー・レジストリーから、タスク管理者のロールにマップされるグループ。

タイプ	**値**
必須	はい
データ型	ストリング
デフォルト	なし
制約事項	ローカル・オペレーティング・システム、Lightweight Directory Access Protocol (LDAP)、またはカスタム・レジストリーをユーザー・レジストリーにすることができます。指定されたグループは、使用されるユーザー・レジストリーに既に存在している必要があります。

システム・モニター・セキュリティー・ロールのマッピング

ユーザー・レジストリーから、タスク・モニターのロールにマップされるグループ。

タイプ	**値**
必須	はい
データ型	ストリング
デフォルト	なし
制約事項	ローカル・オペレーティング・システム、Lightweight Directory Access Protocol (LDAP)、またはカスタム・レジストリーをユーザー・レジストリーにすることができます。指定されたグループは、使用されるユーザー・レジストリーに既に存在している必要があります。

メール・セッション

「メール・セッション」チェック・ボックスを選択すると、メール・セッションがセルの有効範囲で作成されます。これはエスカレーション・メールの送信に必要です。

タイプ	**値**
データ型	チェック・ボックス
デフォルト	非選択状態

- サーバーでは、デフォルトのメール・セッション名は`mail/HTMNotification_`*`node_name_server_name`*です。

- クラスターでは、デフォルトのメール・セッション名は mail/HTMNotification_*cluster_name* です。

Common Event Infrastructure ロギングの使用可能化

Common Event Infrastructure (CEI) ロギングを使用可能または使用不可にすることができます。

タイプ	**値**
データ型	チェック・ボックス
デフォルト	非選択状態

監査ログを使用可能にする

監査ログを、使用可能または使用不可にすることができます。

タイプ	**値**
データ型	チェック・ボックス
デフォルト	非選択状態

ヒューマン・タスク・コンテナーの設定

このパネルを使用して、ヒューマン・タスク・コンテナーを管理します。

ヒューマン・タスク・コンテナーは、アプリケーション・サーバー内でヒューマン・タスクを実行するためのサービスを提供します。この管理コンソール・ページを表示するには、**「サーバー」** → **「アプリケーション・サーバー」** → ***server_name*** → **「ヒューマン・タスク・コンテナー設定」** → **「ヒューマン・タスク・コンテナー」**をクリックします。

注: このパネルで行った変更は、再始動するまでサーバーに影響を及ぼしません。

E メール・セッション JNDI 名

エスカレーション・メールを送信するためにヒューマン・タスク・コンテナーが使用するメール・セッション・リソースの Java Naming and Directory Interface (JNDI) 名。

タイプ	**値**
データ型	読み取り専用ストリング
サーバーでの構成時デフォルト	mail/HTMNotification_*nodeName_serverName*
クラスターでの構成時デフォルト	mail/HTMNotification_*clusterName*

Common Event Infrastructure ロギングの使用可能化

Common Event Infrastructure (CEI) ロギングを使用可能または使用不可にすることができます。

タイプ	**値**
データ型	チェック・ボックス
デフォルト	非選択状態

監査ログを使用可能にする

監査ログを、使用可能または使用不可にすることができます。

タイプ	値
データ型	チェック・ボックス
デフォルト	非選択状態

送信者の E メール・アドレス

エスカレーション E メールの送信者として表示されるアドレス。

タイプ	値
データ型	ストリング
デフォルト	taskmanager.emailservice @htm.companydomain

エスカレーション URL プレフィックス

このプレフィックスは、エスカレーションの詳細を知らせるエスカレーション E メールにリンクを提供する際に使用されます。

タイプ	値
データ型	ストリング
デフォルト	なし

タスク URL プレフィックス

タスクの詳細を知らせる E メールに組み込まれる URL のプレフィックス。

タイプ	値
データ型	ストリング
デフォルト	なし

管理者 URL プレフィックス

タスク管理者役割を実行する受信者が受け取るエスカレーション E メールに組み込まれる URL のプレフィックス。

タイプ	値
データ型	ストリング
デフォルト	なし

Process Explorer URL プレフィックス

Business Process Choreographer Explorer の URL のプレフィックス。

タイプ	値
データ型	ストリング
デフォルト	なし

スタッフ照会リフレッシュ・スケジュール

スタッフ照会をリフレッシュするスケジュール。

タイプ	値
データ型	ストリング
形式	このフィールドは、com.ibm.websphere.scheduler.UserCalendar に基づく crontab 形式 <Minute> <Hour> <Day_of_the_Month> <Month_of_the_Year> <Day_of_the_Week> <Command> を使用します
例	スタッフ照会を毎週木曜日に 1 時間ごとにリフレッシュする場合は、0 * * * 4 ? というスケジュールを使用します
デフォルト	0 0 1 * * ? (毎月の最初の日の深夜 12 時)

スタッフ照会結果のタイムアウト

スタッフ照会の結果を有効と見なす期間 (秒単位)。この期間を過ぎると、スタッフ照会の結果は有効期限が切れます。

タイプ	値
データ型	整数
デフォルト	3600
単位	秒

グループ作業項目の使用可能化

グループ作業項目を使用可能にする場合は、このチェック・ボックスを選択します。バージョン 6.0.2 よりも前のバージョンに合わせて作成したアプリケーションで、com.ibm.bpe.api.StaffResultSet インターフェース、com.ibm.task.api.StaffResultSet インターフェースのいずれかまたは両方を使用している場合は、グループ作業項目を使用可能にする前に、グループ作業項目を処理できるようにアプリケーションを変更する必要があります。

タイプ	値
データ型	チェック・ボックス
デフォルト	選択されません (バージョン 6.0.2 よりも前に作成されたアプリケーションとの互換性を確保するために、グループ作業項目は使用可能になりません)

ヒューマン・タスク・コンテナーのカスタム・プロパティー

このパネルを使用して、ヒューマン・タスク・コンテナーのカスタム・プロパティーを管理します。

カスタム・プロパティーを使用して、ヒューマン・タスク・コンテナーの追加構成パラメーター (エスカレーション・イベントの発生時に E メールを送信するためのパラメーターなど) を設定します。この管理コンソール・ページを表示するには、**「サーバー」** → **「アプリケーション・サーバー」** → ***server_name*** → **「ヒューマン・タスク・コンテナー設定」** → **「ヒューマン・タスク・コンテナー」** → **「カスタム・プロパティー」**をクリックします。

EscalationEmail.ClientDetailURL:

Business Process Choreographer Explorer またはカスタム Web クライアント・アプリケーションのエスカレーション詳細ビューの URL。*htm:task.ClientDetailURL* 変数を使用して、このプロパティーの値を照会できます。このプロパティーは、バージョン 6.0.1 のタスクとの互換性のためにだけ使用されます。バージョン 6.0.2 以降で作成したタスクには、このプロパティーは影響を与えません。

タイプ	**値**
データ型	ストリング
デフォルト	http://clientapphost[:port]/bpc/DetailsView.jsp?id=

EscalationEmail.Subject:

エスカレーションの発生時に送信される E メールの件名を指定します。件名のストリングには、タスクまたはプロセス変数を使用できます。このプロパティーは、バージョン 6.0.1 のタスクとの互換性のためにだけ使用されます。バージョン 6.0.2 以降で作成したタスクには、このプロパティーは影響を与えません。

タイプ	**値**
データ型	ストリング
デフォルト	タスク '%htm:task.displayName%' がエスカレートされました (The task '%htm:task.displayName%' has been escalated)

EscalationEmail.Template:

エスカレーション E メールの本文に使用されるテンプレート・ファイルの URL。このファイルは、HTML ファイルまたはテキスト・ファイルで、E メールの作成時に解決されるタスクおよびプロセス変数を使用できます。このプロパティーは、バージョン 6.0.1 のタスクとの互換性のためにだけ使用されます。バージョン 6.0.2 以降で作成したタスクには、このプロパティーは影響を与えません。

タイプ	**値**
データ型	ストリング
デフォルト	`file:${WAS_INSTALL_ROOT}/ProcessChoreographer/sample/emailNotification.html`

LDAP スタッフ・プラグイン・プロバイダーの構成

このタスクを使用すると、Business Process Choreographer がプロセスを開始、またはアクティビティーやタスクを要求できる対象を決定するために使用する LDAP スタッフ・プラグイン・プロバイダーを構成できます。

サポートされているユーザー・ディレクトリー・サービスの各タイプに、対応するスタッフ・プラグインが必要です。以下のスタッフ・プラグインがサポートされています。

表 1. サポートされているスタッフ・プラグイン・プロバイダー

ユーザー・ディレクトリー・サービス	プラグイン・プロバイダー
Lightweight Directory Access Protocol (LDAP)	LDAP スタッフ・プラグイン・プロバイダー

表 1. サポートされているスタッフ・プラグイン・プロバイダー (続き)

ユーザー・ディレクトリー・サービス	プラグイン・プロバイダー
ローカル・オペレーティング・システムのユーザー・レジストリー	システム・スタッフ・プラグイン・プロバイダー
WebSphere Application Server のユーザー・レジストリー	ユーザー・レジストリー・スタッフ・プラグイン・プロバイダー

既にこれらのプラグインがすべてインストールされています。ユーザー・レジストリーおよびシステム・プラグインは、構成せずに使用できます。

LDAP スタッフ・プラグインは、匿名アクセスを使用して LDAP サーバー用に構成されています。LDAP サーバーは、インストール済みアプリケーション・サーバーに対してはローカルです。LDAP プラグインの構成は、ユーザーが変更できます。

1. 管理コンソールで、「**リソース**」 → 「**スタッフ・プラグイン・プロバイダー**」をクリックします。
2. 有効範囲を「ノード」に設定していない場合は、「**ノード**」を選択して「**適用**」をクリックします。
3. 新規 LDAP 構成を作成するには、以下のようにします。
   a. LDAP スタッフ・プラグイン・プロバイダーの名前をクリックします。
   b. 「**スタッフ・プラグイン構成**」を選択します。
   c. 「**新規作成**」 → 「**参照**」をクリックし、使用するサンプル Extensible Stylesheet Language (XSL) 変換ファイルを選択します。 LDAP 用の標準 XSL 変換は、*install_root*/ProcessChoreographer/Staff/LDAPTransformation.xslにあります。この変換ファイルは変更しないでください。組織の LDAP スキーマに合うよう変換をカスタマイズする必要がある場合は、別のファイル名を付けたコピーを変更します。
   d. 「**次へ**」をクリックします。
   e. スタッフ・プラグイン・プロバイダーの管理名を入力します。
   f. 説明を入力します。
   g. 例えば bpe/staff/ldapserver1 など、ビジネス・プロセスがこのプラグインの参照で使用する Java Naming and Directory Interface (JNDI) 名を入力します。
   h. 「**適用**」をクリックします。
   i. 「**カスタム・プロパティー**」をクリックします。
   j. 設定する各必須プロパティーおよびオプション・プロパティーごとに、プロパティーの名前をクリックして値を入力し、「**OK**」をクリックします。
   k. 変更を適用するには、「**保管**」をクリックします。 このテーブルでは、LDAP プラグインの各プロパティーについて説明します。

LDAP プラグイン・プロパティー	必須またはオプション	コメント
AuthenticationAlias	オプション	例えば mycomputer/My LDAP Alias など、LDAP への接続に使用される認証別名。「**セキュリティー**」 → 「**グローバル・セキュリティー**」 → 「**JAAS 構成**」 → 「**J2C 認証データ**」をクリックして、管理コンソールでこの別名を定義する必要があります。この別名が設定されていない場合は、LDAP サーバーへの匿名ログオンが使用されます。
AuthenticationType	オプション	AuthenticationType プロパティーが設定されていない場合、デフォルト・ログオンは匿名認証になります。これ以外の場合はすべて、デフォルトは単純な認証です。
BaseDN	必須	例えば "o=mycompany, c=us" など、すべての LDAP 検索操作用の基本識別名 (DN)。
CasesentivenessForObjectclasses	オプション	LDAP オブジェクト・クラスの名前がケース・センシティブかどうかを決定します。
ContextFactory	必須	例えば com.sun.jndi.ldap.LdapCtxFactory など、Java Naming and Directory Interface (JNDI) コンテキスト・ファクトリーを設定します。
ProviderURL	必須	この Web アドレスは、LDAP JNDI ディレクトリー・サーバーおよびポートを指す必要があります。フォーマットは、例えば ldap://localhost:389 など、通常の JNDI 構文である必要があります。
SearchScope	必須	すべての検索操作のデフォルト検索スコープ。 baseDN プロパティーの下の検索の深さを決定します。objectScope、onelevelScope、または subtreeScope のいずれかの値を指定します。
additionalParameterName1-5 and additionalParameterValue1-5	オプション	これらの名前と値のペアを使用し、最大 5 つの任意の接続用 JNDI プロパティーを LDAP サーバーに対してセットアップします。

4. プラグインを活動化するには、サーバーを停止および開始します。
5. これらのいずれかのステップで問題が発生した場合は、「*WebSphere Process Server のトラブルシューティング*」PDF を参照してください。

プロセスは、これで、スタッフ・サポート・サービスを使用してスタッフ照会を解決し、特定のユーザーがどのアクティビティーを実行できるかを決定できるようになりました。

ステップ で構成を続行します。

スタッフ・サービス設定

このページを使用して、スタッフ・サービスを使用可能または使用不可にして、サーバーによって使用されるスタッフ・プラグイン・リソースを管理します。

この管理コンソールのページを表示するには、「**サーバー**」 → 「**アプリケーション・サーバー**」 → ***server_name*** をクリックします。次に、「**ビジネス・インテグレーション**」セクションで、「**スタッフ・サービス**」をクリックします。

サーバー始動時にサービスを使用可能にする

サーバーが、スタッフ・サービスの開始を試みるどうかを指定します。

タイプ
値

デフォルト
選択

範囲
選択 アプリケーション・サーバーが開始されると、自動的にスタッフ・サービスの始動を試みます。
クリア サーバーは、スタッフ・サービスの開始を試みません。スタッフ・プラグイン・リソースがこのサーバーで使用されている場合、システム管理者は、手動でスタッフ・サービスを開始するか、この始動プロパティーを選択して、サーバーを再始動する必要があります。

スタッフ・プラグイン・プロバイダー・コレクション

スタッフ・プラグインは、ユーザー情報の検索を行います。このパネルを使用して、スタッフ・プラグイン・プロバイダーを管理します。

この管理コンソール・パネルを表示するには、「**リソース**」 → 「**スタッフ・プラグイン・プロバイダー**」をクリックします。既存のプラグイン・プロバイダーが表示されます。

通常、スタッフ・プラグイン・プロバイダーは、ノードの有効範囲でのみ定義されます。テーブルに表示されるスタッフ・プラグイン・プロバイダーの有効範囲は、セル、ノード、サーバーの各有効範囲に対応する選択済みのラジオ・ボタンへの矢印で示されます。別の有効範囲のスタッフ・プラグイン・プロバイダーを表示するには、対象の有効範囲のラジオ・ボタンを選択して、「**適用**」をクリックします。

既存のプロバイダーのプロパティーを表示したり変更したりするには、そのプロバイダーの名前をクリックします。新しいプロバイダーを構成するには、「**新規作成**」をクリックします。

名前

管理目的のために、スタッフ・プラグイン・プロバイダーが認識される名前。

タイプ
値

データ型
ストリング

説明

スタッフ・プラグイン・プロバイダーの説明。

タイプ
値

データ型
ストリング

スタッフ・プラグイン・プロバイダー設定

このパネルを使用して、スタッフ・プラグイン・プロバイダーの設定を変更します。

スタッフ・プラグインは、ユーザーのディレクトリーから情報を取得するために使用されます。各スタッフ・プラグイン・プロバイダーは、プラグインを含む名前と Java アーカイブ (JAR) ファイルを指定することにより、ランタイム環境に登録されます。 JAR ファイルの構成ファイルは、プラグインのプロパティーに加え、そのプラグインを表すクラス名を定義します。

この管理コンソール・ページを表示するには、まず「**リソース**」 → 「**スタッフ・プラグイン・プロバイダー**」をクリックします。すべてのスタッフ・プラグイン・プロバイダーと各プロバイダーの有効範囲をリストしたテーブルが表示されます。

通常、スタッフ・プラグイン・プロバイダーは、ノードの有効範囲でのみ定義されます。テーブルに表示されるスタッフ・プラグイン・プロバイダーの有効範囲は、セル、ノード、サーバーの各有効範囲に対応する選択済みのラジオ・ボタンへの矢印で示されます。別の有効範囲のスタッフ・プラグイン・プロバイダーを表示するには、対象の有効範囲のラジオ・ボタンを選択して、「**適用**」をクリックします。

スタッフ・プラグインの構成やカスタム・プロパティーを表示したり変更したりするには、そのプラグインの名前「***staffpluginprovider_name***」をクリックします。

有効範囲

このスタッフ・プラグイン・プロバイダーの有効範囲。

タイプ	**値**
データ型	読み取り専用ストリング
有効な値	セル、ノード、またはサーバーの名前。例: `cells:viennaNode02Cell:nodes:viennaNode02`
説明	有効範囲は、リソース定義が可視になるレベルを決定します。ノードに固有の設定が存在するため、スタッフ・プラグイン構成にはノードの有効範囲を使用します。

名前

管理目的のために、スタッフ・プラグイン・プロバイダーが認識される名前。

タイプ	**値**
データ型	ストリング

説明

スタッフ・プラグイン・プロバイダーの説明。

タイプ	**値**
データ型	ストリング

JAR ファイル

プラグインを含む JAR ファイルのファイル名 (絶対パスを含む)。

タイプ	値
データ型	読み取り専用ストリング

スタッフ・プラグイン構成コレクション

このページを使用して、スタッフ・プラグイン構成を管理します。

スタッフ・プラグイン構成は、スタッフ・プラグイン・プロバイダーについて定義されます。スタッフ・プラグイン構成は、スタッフ・プラグイン・プロバイダーによって指定される、任意のカスタム・プロパティーを定義できます。各スタッフ・プラグイン・プロバイダーは、複数のスタッフ・プラグイン構成を持つことができます。「**新規**」 をクリックして新規構成を作成するか、既存の構成の名前をクリックしてそのプロパティーを表示または変更します。

この管理コンソール・ページを表示するには、「**リソース**」 → 「**スタッフ・プラグイン・プロバイダー**」 → 「***staffpluginprovider_name***」 → 「**スタッフ・プラグイン構成**」をクリックします。

名前

管理目的で使用されるスタッフ・プラグイン構成の名前。名前をクリックすると、構成の設定を表示または変更できます。

タイプ	値
データ型	ストリング

説明

スタッフ・プラグイン構成の説明。

タイプ	値
データ型	ストリング

JNDI 名

ネーム・スペースでスタッフ・プラグイン構成を検索するのに使用される Java Naming and Directory Interface (JNDI) 名。

タイプ	値
データ型	ストリング

XSL 変換ファイル

Extensible Style Language (XSL) 変換ファイルのファイル名 (絶対パスを含む)。

タイプ	値
データ型	ストリング

スタッフ・プラグイン構成の設定

このページを使用して、スタッフ・プラグイン構成の設定を表示または変更します。

この管理コンソール・ページを表示するには、「**リソース**」 → 「**スタッフ・プラグイン・プロバイダー**」 → 「***staffpluginprovider_name***」 → 「**スタッフ・プラグイン構成**」 → 「***staffpluginconfiguration_name***」をクリックします。

有効範囲

このスタッフ・プラグイン・プロバイダーの有効範囲。有効範囲は、リソース定義が可視になるレベルを決定します。

タイプ	**値**
データ型	読み取り専用ストリング
有効な値	セル、ノード、またはサーバーの名前。例: `cells:viennaNode02Cell:nodes:viennaNode02`

名前

管理目的で使用されるスタッフ・プラグイン構成の名前。

タイプ	**値**
データ型	ストリング

説明

スタッフ・プラグイン構成の説明。

タイプ	**値**
データ型	ストリング

JNDI 名

ネーム・スペースでスタッフ・プラグイン構成を検索するのに使用される Java Naming and Directory Interface (JNDI) 名。

タイプ	**値**
データ型	ストリング

XSL 変換ファイル

Extensible Style Language (XSL) 変換ファイルのファイル名 (絶対パスを含む)。サンプル・プラグイン用にデフォルトの XSL 変換ファイルが提供されています。変換ファイルをカスタマイズした場合は、ファイルへのパスを指定します。パス名には WebSphere 環境変数を含めることができます。

タイプ	**値**
データ型	ストリング

スタッフ・サービスについて

Business Process Choreographer では、ビジネス・プロセスとヒューマン・タスクのロジックをスタッフの解決から分離することができます。スタッフ照会は、ディレクトリー・サービスに固有のプラグインを使用して解決されます。スタッフ・サービス使用の基本的な側面については、以下の項で説明します。

- 『スタッフ照会とスタッフ・サービスの概念』
- 75 ページの『スタッフ照会のインプリメント』
- 75 ページの『スタッフ照会の動詞セット』
- 78 ページの『リポジトリー固有のスタッフ照会』
- 79 ページの『スタッフ動詞の XSL 変換ファイル』
- 81 ページの『スタッフ動詞でのタスク・コンテキスト変数およびプロセス・コンテキスト変数の使用』
- 81 ページの『E メール動詞セット』

スタッフ解決プラグインについて詳しくは、WebSphere Business Process Choreographer の 「*Process Choreographer: Staff Resolution Architecture*」、「*Process Choreographer: Programming Model for Staff Resolution*」、および「*Process Choreographer: Staff Resolution Parameter Reference*」のホワイト・ペーパーを参照してください。

スタッフ照会とスタッフ・サービスの概念

スタッフ・サポート・サービスのスタッフ照会を定義するには、WebSphere Integration Developer を使用します。スタッフ照会はスタッフ照会テンプレート、スタッフ動詞が基本になっており、ヒューマン・タスクおよびビジネス・プロセス (ProcessStarter や PotentialOwners など) の予測されるロールと関連しています。

スタッフ動詞は固有の名前で識別しますが、ここには一連の照会パラメーターが含まれます。パラメーター化されたスタッフ動詞はアプリケーションのデプロイメント時に変換され、リポジトリー固有のスタッフ照会を判別します。これを使用するのは、ユーザー・リポジトリーからユーザーの ID を検索するためにビジネス・プロセスまたはヒューマン・タスクを実行するときです。

各ビジネス・プロセスまたはヒューマン・タスクは、その JNDI 名によって特定のスタッフ・プラグイン構成と関連付けられます。この構成はデプロイメント時にプロセス定義またはタスク定義から抽出され、検出された各スタッフ動詞をリポジトリー固有のスタッフ照会にマップするときに使用されます。このマッピングは XSL 変換ファイルによって制御されますが、このファイルはスタッフ動詞を入力として使用し、対応するリポジトリー固有の照会を出力します。

デフォルトでは、異なるユーザー・リポジトリー・オプションを表す次の 3 つのスタッフ・プラグイン・プロバイダーがあります。

- LDAP スタッフ・プラグイン・プロバイダーは、LDAP サーバーに対して実行できるスタッフ照会を生成するときに使用します。
- ユーザー・レジストリー・スタッフ・プラグイン・プロバイダーは、WebSphere Application Server のユーザー・レジストリーに対して実行できるスタッフ照会を生成するときに使用します。
- システム・スタッフ・プラグイン・プロバイダーは、ユーザー・リポジトリーとは関連していません。その代わりに、スタッフ動詞パラメーターから直接得られるユーザー ID を戻します。このプラグイン・プロバイダーの目的は、テストとプロトタイピングです。

前述のスタッフ・プラグイン・プロバイダーは、それぞれ 1 つ以上の構成と関連付けられます。特に、1 つの構成が 1 つの XSL 変換ファイルを指定し、このファイ

ルがスタッフ動詞とリポジトリーに固有のスタッフ照会との間のマッピングを実行します。デフォルトでは、以下の変換ファイルが用意されています。

- LDAPTransformation.xsl ファイルは、JNDI インターフェースを介して実行できる LDAP 固有のスタッフ照会にスタッフ動詞をマップします。
- UserRegistryTransformation.xsl ファイルは、WebSphere ユーザー・レジストリーに固有のスタッフ照会にスタッフ動詞をマップします。
- SystemTransformation.xsl ファイルは、スタッフ動詞内に指定されている実際のユーザー ID にスタッフ動詞をマップします。実ユーザー・リポジトリーは必要ありません。
- EverybodyTransformation.xsl ファイルは、すべてのスタッフ動詞をデフォルトの結果である「everybody」にマップします。実ユーザー・リポジトリーは必要ありません。

スタッフ照会のインプリメント

次の例では、スタッフ照会をインプリメントするときに必要な手順を要約して示します。

1. WebSphere Integration Developer を使用することにより、モデラーは新規作成タスクとスタッフ・プラグイン構成 bpe/staff/sampleldapconfiguration を関連付けます。
2. WebSphere Integration Developer を使用することにより、モデラーはタスクのロールと対応するスタッフ動詞を関連付けます。例えば、PotentialOwners は以下のパラメーターを含むスタッフ動詞「Group Members」と関連付けられます。
   - 値「cn=group1,dc=mycomp,dc=com」に設定された「GroupName」
   - 値「true」に設定された「IncludeSubgroups」
3. タスクのコンテキストで、WebSphere Integration Developer は、動詞の定義を次のように XML 断片として保管します。

```
<verb>
      <name>Group Members</name>
      <parameter id="GroupName">cn=group1,dc=mycomp,dc=com</parameter>
      <parameter id="IncludeSubgroups">true</parameter>
</verb>
```

4. タスクを WebSphere Application Server に配置すると、スタッフ・サポート・サービスが LDAP スタッフ・プラグイン・プロバイダー `bpe/staff/sampleldapconfiguration` を使用することを設定します。関連の LDAPTransformation.xsl ファイルは、スタッフ動詞を LDAP 固有の照会に変換するときに使用します。この照会は、内部で格納されます。

スタッフ照会の動詞セット

スタッフ・サポート・サービスは、ユーザー・リポジトリー・インフラストラクチャーから独立した、抽象的な形式の照会を受け入れます。プロセス・エディターおよびタスク・エディターの両方に、プロセスおよびタスクをモデル化時に使用可能な、定義済みのスタッフ動詞のセットがあります。これらの動詞は、`VerbSet.xml` ファイルに含まれています。このファイルは WebSphere Integration Developer と共にインストールされます。

個々のスタッフ解決プラグインおよび XSLT マッピング・ファイルで、すべての動詞がサポートされるわけではありません。例えば、*Manager of Employee* という動詞は、ユーザー・レジストリーやシステム・プラグインを使用する場合は使用できません。スタッフ照会動詞セットは変更することができます。ファイルのコピーに対して変更を加えてください。コピーされたファイルは必ず、別のファイル名にしてください。

次の定義済み動詞セットを使用することができます。これらの各動詞と組み合わせて使用できるパラメーターについては、『定義済みスタッフ動詞とそのパラメーター』を参照してください。

Department Members
この動詞は、部門のメンバーを検索する照会を定義するときに使用します。検索されるユーザーは、特定の部門 (DepartmentName、AlternativeDepartmentName1、または AlternativeDepartmentName2) のいずれかに属します。この動詞は、LDAP プラグインによってサポートされます。デフォルトのマッピング XSLT ファイルを、自分の組織の LDAP スキーマに合うようにカスタマイズする必要がある場合があります。

Group この動詞は、groupName パラメーターと一致するグループの名前を検索するときに使用します。これはグループ作業項目と共に使用されます。この動詞は、すべてのプラグインによってサポートされます。

Everybody
この動詞は、WebSphere Process Server によって認証されたすべてのユーザーに作業項目を割り当てるときに使用します。この動詞は、システム、ユーザー・レジストリー、LDAP の各プラグインによってサポートされます。

Group Members
この動詞は、3 つまでのグループのメンバーを検索する照会を定義するときに使用します。検索されるユーザーは、特定のグループ (GroupName、AlternativeGroupName1、または AlternativeGroupName2) のいずれかに属します。この動詞は、ユーザー・レジストリーおよび LDAP の各プラグインによってサポートされます。デフォルトのマッピング XSLT ファイルを、自分の組織の LDAP スキーマに合うようにカスタマイズする必要がある場合があります。

Group Members without Named Users
この動詞は、グループのメンバーのうち、明示的に指定されたそのグループのユーザーを除くメンバー を検索する照会を定義するときに使用します。除外するものとして、1 人以上のメンバーをコンマで区切ったリストの形で指定することができます。この動詞は、ユーザー・レジストリーおよび LDAP の各プラグインによってサポートされます。デフォルトのマッピング XSLT ファイルを、自分の組織の LDAP スキーマに合うようにカスタマイズする必要がある場合があります。

Group Members without Filtered Users
この動詞は、グループのメンバーのうち、LDAP 検索フィルターで定義された一連のユーザーを除くメンバーを検索する照会を定義するときに使用します。この動詞は、LDAP プラグインによってサポートされます。デフォルトのマッピング XSLT ファイルを、自分の組織の LDAP スキーマに合うようにカスタマイズする必要がある場合があります。

Group Search
この動詞は、属性の一致に基づいてグループを検索し、そのグループのメン

バーを取得するときに使用します。この動詞は、ユーザー・レジストリーおよび LDAP の各プラグインによってサポートされます。デフォルトのマッピング XSLT ファイルを、自分の組織の LDAP スキーマに合うようにカスタマイズする必要がある場合があります。

Manager of Employee
この動詞は、人の名前を使用してその人の管理者を検索するときに使用します。この動詞は、LDAP プラグインによってサポートされます。デフォルトのマッピング XSLT ファイルを、自分の組織の LDAP スキーマに合うようにカスタマイズする必要がある場合があります。

Manager of Employee by user ID
この動詞は、人のユーザー ID を使用してその人の管理者を検索するときに使用します。この動詞は、コンテキスト照会と組み合わせて使用すると便利です。この動詞は、LDAP プラグインによってサポートされます。デフォルトのマッピング XSLT ファイルを、自分の組織の LDAP スキーマに合うようにカスタマイズする必要がある場合があります。

Native Query
この動詞は、ディレクトリー固有のパラメーターに基づいた固有の照会を定義するときに使用します。この動詞は、ユーザー・レジストリーおよび LDAP の各プラグインによってサポートされます。デフォルトのマッピング XSLT ファイルを、自分の組織の LDAP スキーマに合うようにカスタマイズする必要がある場合があります。

Nobody
この動詞は、標準のユーザーが作業項目にアクセスするのを拒否するときに使用します。インライン・タスクの場合は、プロセス管理者とビジネス・プロセスのシステム管理者にのみアクセス権があります。スタンドアロン・タスクの場合は、ヒューマン・タスク管理者とヒューマン・タスクのシステム管理者のみにアクセス権があります。使用する API により、許可される J2EE 管理者は異なります。ビジネス・プロセス API の場合は BPESystemAdministrator ユーザーになり、ヒューマン・タスク API の場合は TaskSystemAdministrator ユーザーになります。この動詞は、システム、ユーザー・レジストリー、LDAP の各プラグインによってサポートされます。

Person Search
この動詞は、属性の一致に基づいて人を検索するときに使用します。この動詞は、ユーザー・レジストリーおよび LDAP の各プラグインによってサポートされます。デフォルトのマッピング XSLT ファイルを、自分の組織の LDAP スキーマに合うようにカスタマイズする必要がある場合があります。

Role Members
この動詞は、スタッフ・リポジトリーのロールに関連付けられているユーザーを検索するときに使用します。検索されるユーザーは、指定のロール (RoleName、AlternativeRoleName1、または AlternativeRoleName2) のいずれかに属します。この動詞は、LDAP プラグインによってサポートされます。デフォルトのマッピング XSLT ファイルを、自分の組織の LDAP スキーマに合うようにカスタマイズする必要がある場合があります。

Users
この動詞は、名前で認識されているユーザーについてのスタッフ照会を定義するときに使用します。プロセス・テンプレートにユーザー名をハードコーディングすることはお勧めできません。この動詞は、テストを目的とする場合に役立ちます。この動詞は、システム、ユーザー・レジストリー、LDAP

の各プラグインによってサポートされます。デフォルトのマッピング XSLT ファイルを、自分の組織の LDAP スキーマに合うようにカスタマイズする必要がある場合があります。

Users by user ID

この動詞は、ユーザー ID が認識されているユーザーについてのスタッフ照会を定義するときに使用します。プロセス・テンプレートとタスク・テンプレートにユーザー ID をハードコーディングすることはお勧めできませんが、この動詞はコンテキスト照会と組み合わせて使用すると便利です。例えば、次のように指定します。

```
User [username='%wf:process.starter%']
```

この動詞は、テストを目的とする場合に役立ちます。この動詞は、システム、ユーザー・レジストリー、LDAP の各プラグインによってサポートされます。

Users by user ID without Named Users

この動詞は、ユーザー ID が認識されているユーザー (明示的に指定されたユーザー ID を持つユーザーを除く) についてのスタッフ照会を定義するときに使用します。プロセス・テンプレートとタスク・テンプレートにユーザー ID をハードコーディングすることはお勧めできませんが、この動詞はコンテキスト照会と組み合わせて使用すると便利です。例えば、次のように指定します。

```
User [userID='%htm:task.potentialStarters%', NamedUsers='%wf:activity(...).owner%']
```

リポジトリー固有のスタッフ照会

スタッフ・プラグイン構成に関連している XSL 変換ファイルは、特定のリポジトリーに固有のスタッフ照会を生成するときに使用します。各照会はそれぞれのスタッフ・プラグインによって実行され、これによってユーザー ID のリストを取得することができます。スタッフ・プラグインで使用できる定義済み照会は、プラグインで実行できる呼び出しに対応しているため、固定されています。

定義済み照会を基本にすると、以下の仕組みを使用してより複雑な照会を作成できます。

- 照会結果を結合することは、個々の照会で戻されたユーザー ID をユーザー ID の現在の結果リストに追加することを意味します。例えば、LDAP スタッフ・プラグインでは、特に以下のタイプの定義済み照会を使用できます。

  指定したグループのグループ・メンバーのユーザー ID リスト:

  ```
  <sldap:usersOfGroup groupDN="cn=group1,dc=mycomp" recursive="yes">
  ...
  </sldap:usersOfGroup>
  ```

  指定したユーザーのユーザー ID:

  ```
  <sldap:user dn="uid=user1,dc=mycomp" .../>
  ```

  指定したグループのメンバーのユーザー ID リストに加えて指定したユーザーの ID という複雑な照会を構成できます。

```
<sldap:staffQueries>
     <sldap:usersOfGroup groupDN="cn=group1,dc=mycomp" recursive="yes">
    ...
    </sldap:usersOfGroup>
    <sldap:user dn="uid=user1,dc=mycomp" .../>
</sldap:staffQueries>
```

- 照会結果の差をとることは、<remove> 照会で戻されたユーザーID を現在の結果リストから削除することを意味します。例えば、指定したグループ・メンバーを対象として検索した ID のリストから ″user1″ を削除する場合は、次のように記述します。

```
<sldap:staffQueries>
     <sldap:usersOfGroup groupDN="cn=group1,dc=mycomp" recursive="yes">
    ...
    </sldap:usersOfGroup>
    <sldap:remove value="user1"/>
</sldap:staffQueries>
```

- 照会結果を参照することは、1 つの照会から得られた結果を使用して後続の照会の振る舞いに影響を及ぼすことを意味します。例えば、次の断片では、照会を 2 回実行しています。まず、ユーザー ″uid=user1,...″ の LDAP 項目にある ″manager″ 属性の値を取得して中間の変数 ″supervisor″ に保管します。次に、この変数を使用してマネージャーの LDAP 項目を検索し、関連のユーザー ID を取得します。

```
<sldap:staffQueries>
       <sldap:intermediateResult name="supervisor">
         <sldap:user dn="uid=user1,dc=mycomp" attribute="manager" ... />
       </sldap:intermediateResult>
      <sldap:user dn="%supervisor% .../>
    </sldap:staffQueries>
```

前述した 3 つの組み合わせ規則に従って構成したスタッフ照会は、スタッフ・プラグインで実行できます。サポートされている各スタッフ・プラグインのすべての定義済みスタッフ照会の詳細と、照会のその他の組み合わせ例については、WebSphere Business Process Choreographer のホワイト・ペーパー「*Process Choreographer: Staff Resolution Parameter Reference*」を参照してください。

スタッフ動詞の XSL 変換ファイル

スタッフ・プラグイン構成用に指定された XSL 変換ファイルは、スタッフ動詞とリポジトリー固有のスタッフ照会間のマッピングを定義します。各スタッフ・プラグイン構成には、それ専用の XSL 変換ファイルがあることが予想されています。

デフォルトの変換ファイルは、次のとおりです。

- LDAP スタッフ・プロバイダー・プラグイン用の LDAPTransformation.xsl
- ユーザー・レジストリー・スタッフ・プロバイダー・プラグイン用の UserRegistryTransformation.xsl
- システム・スタッフ・プロバイダー・プラグイン用の SystemTransformation.xsl および EverybodyTransformation.xsl

これらの変換ファイルは、一連の定義済みスタッフ動詞を、対応するリポジトリー固有の照会 (単純および複合) にマップします。これらのファイルは、*install_root*/ProcessChoreographer/Staff ディレクトリーにあります。

変換ファイルでは、生成されたリポジトリー固有照会を使用する場合に、スタッフ動詞とその実行について特定のセマンティクスを想定しています。他のセマンティクスが必要な場合は、それに応じて変換ファイルでのマッピングも変更する必要があります。

例えば、LDAP スタッフ・プラグインは次のような定義済みのスタッフ動詞と合わせて提供されます。

```
<staff:verb>
  <staff:name>Manager of Employee</staff:name>
   <staff:parameter id="EmployeeName">
          uid=anEmployeeName,cn=users,dc=ibm,dc=com
   </staff:parameter>
</staff:verb>
```

この動詞は LDAPTransformation.xsl ファイルによって、次のように LDAP 照会にマップされます。

```
<sldap:staffQueries>
     <sldap:intermediateResult name="supervisor">
     <sldap:user dn="anEmployeeName" attribute="manager"
                                objectclass="inetOrgPerson"/>
     </sldap:intermediateResult>
     <sldap:user dn="%supervisor%" attribute="uid" objectclass="inetOrgPerson"/>
</sldap:staffQueries>
```

ここでは、スーパーバイザー (supervisor) の LDAP DN が従業員 (employee) の属性 ″manager″ に格納されていることを明示的に想定しています。この動詞のセマンティクスが異なる場合、例えばスーパーバイザーの元の LDAP 属性が ″teacher″ である場合を考えます。この場合は、属性の違いに応じて LDAP 固有の照会を次のように変更する必要があります。

```
<sldap:staffQueries>
     <sldap:intermediateResult name="supervisor">
     <sldap:user dn="anEmployeeName" attribute="teacher"
                         objectclass="inetOrgPerson"/>
     </sldap:intermediateResult>
     <sldap:user dn="%supervisor%" attribute="uid" objectclass="inetOrgPerson"/>
</sldap:staffQueries>
```

これを実現するには、LDAPTransformation.xsl ファイルをしかるべく改変します。

```
  <xsl:template name="ManagerOfEmployee">
      <sldap:staffQueries>...
      <sldap:intermediateResult>
           <xsl:attribute name="name">supervisor</xsl:attribute>
           <sldap:user>
                 <xsl:attribute name="dn">
                     <xsl:value-of select="staff:parameter[@id='EmployeeName']"/>
                 </xsl:attribute>
                 <xsl:attribute name="attribute">teacher</xsl:attribute>
                 ...
           </sldap:user>
      </sldap:intermediateResult>
      <sldap:user>
            <xsl:attribute name="dn">%supervisor%</xsl:attribute>
            ...
      </sldap:user>
      </sldap:staffQueries>
</xsl:template>
```

デフォルトの変換ファイルを表示すると、マッピング動作の理解を深めることができます。デフォルトの変換のセマンティクスについては、75 ページの『スタッフ照会の動詞セット』を参照してください。

スタッフ動詞でのタスク・コンテキスト変数およびプロセス・コンテキスト変数の使用

特定のスタッフ動詞では、ビジネス・プロセス・コンテキスト変数とヒューマン・タスク・コンテキスト変数をパラメーター値として使用できます。これにより、コンテキストによって提供される情報に基づいて、スタッフ・サポート・サービスが実行時にスタッフ動詞を解決できるようになります。スタッフ動詞の例を次に示します。

```
<verb>
<name>Users by user ID</staff:name>
 <parameter id="UserID">%htm:input.¥name%</staff:parameter>
</verb>
```

このスタッフ動詞は、タスク・コンテキスト変数 htm:input.¥name をパラメーターとして指定します。この変数は、タスクの開始時にタスクが受け取る入力メッセージの名前の部分を表します。スタッフ・サポート・サービスは、コンテキスト変数を実際のタスク・コンテキスト値に動的に置き換えます。

コンテキスト変数を使用できる動詞とパラメーターの詳細については、82 ページの『定義済みスタッフ動詞』を参照してください。

E メール動詞セット

WebSphere Integration Developer の E メール動詞セットの目的は、タスク・エスカレーションの E メール通知です。これらの E メール動詞は、モデル化およびデプロイメント中にスタッフ・リポジトリーについて実行できる一連の照会に変換されます。E メール動詞は、LDAP プラグインがサポートする最も一般的なスタッフ動詞に対して定義されます。以下の E メール動詞が使用可能です。

- Email Address for Department Members
- Email Address for Group Members
- Email Address for Group Members without Names Users
- Email Address for Group Members without Filtered Users
- Email Address for Group Search
- Email Address for Role Members
- Email Address for Users
- Email Address for Users by User ID

その他の LDAP スタッフ動詞の場合は、スタッフ動詞が取得したユーザー ID が、Email Address for Users by User ID 動詞への入力として使用されます。

E メール動詞を特定のスタッフ・リポジトリーについての照会として実行するには、これらの動詞を、LDAP XSL 変換を使用して実行可能な照会に変換する必要があります。変換の結果 (マッピング) は、LDAP スタッフ解決プラグインによって実行することができます。実行時に、user1@mycomp.com、user2@mycomp.com などの E メール・アドレスのセットが照会によって戻されます。

定義済みスタッフ動詞

定義済みスタッフ動詞は、スタッフ・リポジトリーに対する照会を作成するために用意されています。

WebSphere Integration Developer でスタッフ動詞を使用して、ビジネス・プロセス内またはヒューマン・タスク内のスタッフの割り当てをモデル化することができます。スタッフ・リポジトリー・タイプに使用可能なスタッフ動詞のみ使用できます。スタッフ動詞は、モデル化およびデプロイメント中にスタッフ・リポジトリーについて実行できる一連の照会に変換されます。

LDAP の定義済みスタッフ動詞:

Business Process Choreographer の LDAP スタッフ・プラグインで使用する定義済みスタッフ動詞およびパラメーターについて説明します。

WebSphere Integration Developer でスタッフ動詞を使用して、ビジネス・プロセス内またはヒューマン・タスク内のスタッフの割り当てをモデル化することができます。このスタッフ動詞は、モデル化およびデプロイメント中に LDAP スタッフ・リポジトリーについて実行できる一連の照会に変換されます。このセクションでは、次の定義済みスタッフ動詞用のパラメーターを示します。

- Department Members
- Group
- Everybody
- Group Members
- Group Members without Named Users
- Group Members without Filtered Users
- Group Search
- Manager of Employee
- Manager of Employee by user ID
- Native Query
- Nobody
- Person Search
- Role Members
- Users
- Users by user ID
- Users by user ID without Named Users

Department Members

この動詞は、部門のメンバーを検索する照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
DepartmentName	必須	ストリング	検索するユーザーの部門名。
IncludeNestedDepartments	必須	ブール	ネストされた部門を照会で考慮するかどうかを指定します。
Domain	オプション	ストリング	その部門が属するドメイン。このパラメーターは、照会をディレクトリーのサブセットに限定する場合に使用します。
AlternativeDepartmentName1	オプション	ストリング	ユーザーが属することのできる追加の部門。

パラメーター	使用方法	タイプ	説明
AlternativeDepartmentName2	オプション	ストリング	ユーザーが属することのできる追加の部門。

Group

この動詞は、グループのメンバーを許可する照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
GroupId	必須	ストリング	許可するユーザー・グループの名前。

Everybody

この動詞は、WebSphere Process Server によって認証されたすべてのユーザーに作業項目を割り当てるときに使用します。この動詞はパラメーターを取りません。

Group Members

この動詞は、グループのメンバーを検索する照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
GroupName	必須	ストリング	検索するユーザーのグループ名。
IncludeSubgroups	必須	ブール	ネストされたサブグループを照会で考慮するかどうかを指定します。
Domain	オプション	ストリング	そのグループが属するドメイン。このパラメーターは、照会をディレクトリーのサブセットに限定する場合に使用します。
AlternativeGroupName1	オプション	ストリング	ユーザーが属することのできる追加のグループ。
AlternativeGroupName2	オプション	ストリング	ユーザーが属することのできる追加のグループ。

Group Members without Named Users

この動詞は、グループのメンバーのうち、明示的に指定されたユーザーを除くすべてのメンバーを検索する照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
GroupName	必須	ストリング	検索するユーザーのグループ名。実行時に評価されるカスタム・プロパティーをサポートします。
IncludeSubgroups	必須	ブール	ネストされたサブグループを照会で考慮するかどうかを指定します。
NamedUsers	必須	ストリング	検索されたグループ・メンバーのリストから除外するユーザーのユーザー ID。コンテキスト変数およびカスタム・プロパティー (例: `%htm:task.originator%`) をサポートします。

Group Members without Filtered Users

この動詞は、グループのメンバーのうち、LDAP 検索フィルターで定義された一連のユーザーを除くすべてのメンバーを検索する照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
GroupName	必須	ストリング	検索するユーザーのグループ名。
IncludeSubgroups	必須	ブール	ネストされたサブグループを照会で考慮するかどうかを指定します。
FilterAttribute	必須	ストリング	LDAP フィルターで使用する属性の名前。
FilterValue	必須	ストリング	LDAP フィルターで使用するフィルター値。

Group Search

この動詞は、属性の一致に基づいてグループを検索し、そのグループのメンバーを取得するときに使用します。属性を 1 つ設定する必要があります。複数の属性を設定すると、最初の属性のみが評価されます。

パラメーター	使用方法	タイプ	説明
GroupID	オプション	ストリング	検索するユーザーのグループ ID。
Type	オプション	ストリング	検索するユーザーのグループ・タイプ。
IndustryType	オプション	ストリング	ユーザーが属するグループの業界のタイプ。
BusinessType	オプション	ストリング	ユーザーが属するグループの業種。
GeographicLocation	オプション	ストリング	ユーザーが配置されている場所の標識。
Affiliates	オプション	ストリング	ユーザーの関連会社。
DisplayName	オプション	ストリング	グループの表示名。
Secretary	オプション	ストリング	ユーザーの秘書。
Assistant	オプション	ストリング	ユーザーのアシスタント。
Manager	オプション	ストリング	ユーザーの管理者。
BusinessCategory	オプション	ストリング	ユーザーが属するグループの業種。
ParentCompany	オプション	ストリング	ユーザーの親会社。

Manager of Employee

この動詞は、人の名前を使用してその人の管理者を検索するときに使用します。

パラメーター	使用方法	タイプ	説明
EmployeeName	必須	ストリング	管理者が検索対象となる従業員の名前。
Domain	オプション	ストリング	従業員が属するドメイン。このパラメーターは、照会をディレクトリーのサブセットに限定する場合に使用します。

Manager of Employee by user ID

この動詞は、人のユーザー ID を使用してその人の管理者を検索するときに使用します。

パラメーター	使用方法	タイプ	説明
EmployeeUserID	必須	ストリング	自分の管理者が検索対象となる従業員のユーザー ID。%wf:process.starter% などのコンテキスト変数およびカスタム・プロパティーをサポートします。
Domain	オプション	ストリング	従業員が属するドメイン。このパラメーターは、照会をディレクトリーのサブセットに限定する場合に使用します。

Native Query

この動詞は、ディレクトリー固有のパラメーターに基づいた固有の照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
QueryTemplate	必須	ストリング	照会に使用する照会テンプレート。ユーザー・レジストリーおよび LDAP プラグインのデフォルトのマッピング・ファイルは、テンプレートの search、user、および usersOfGroup をサポートします。
Query	必須	ストリング	照会を指定します。%wf:process.starter% などのコンテキスト変数およびカスタム・プロパティーを使用できます。照会のタイプは、照会テンプレートによって異なります。 • 検索テンプレート: 検索フィルター • ユーザー・テンプレート: ユーザー dn • usersOfGroup: グループ dn
AdditionalParameter1	オプション	ストリング	照会を指定します。コンテキスト変数 (例: %wf:process.starter%) を使用できます。パラメーターのタイプは、照会テンプレートによって異なります。 • 検索テンプレート。再帰的検索を行うかどうかの指定に使用します。サポートされる値 : yes および no • ユーザー・テンプレート。非サポート • usersOfGroup。再帰的検索を行うかどうかの指定に使用します。サポートされる値 : yes および no
AdditionalParameter2	オプション	ストリング	この動詞は、追加パラメーターを指定するときに使用します。
AdditionalParameter3	オプション	ストリング	この動詞は、追加パラメーターを指定するときに使用します。 デフォルトのマッピング XSLT ファイルを使用する場合、このパラメーターはサポートされません。
AdditionalParameter4	オプション	ストリング	この動詞は、追加パラメーターを指定するときに使用します。 デフォルトのマッピング XSLT ファイルを使用する場合、このパラメーターはサポートされません。

パラメーター	使用方法	タイプ	説明
AdditionalParameter5	オプション	ストリング	この動詞は、追加パラメーターを指定するときに使用します。 デフォルトのマッピング XSLT ファイルを使用する場合、このパラメーターはサポートされません。

Nobody

インライン・タスクの場合は、ビジネス・プロセス管理者のみにアクセス権があります。 スタンドアロン・タスクの場合は、ヒューマン・タスク管理者のみにアクセス権があります。 さらに、Business Flow Manager API を使用する場合は、BPESystemAdministrator のロール・メンバーにアクセス権があり、 Human Task Manager API の場合は、TaskSystemAdministrator のロール・メンバーにアクセス権があります。この動詞はパラメーターを取りません。

Person Search

この動詞は、属性の一致に基づいて人を検索するときに使用します。属性を 1 つ設定する必要があります。複数の属性を設定すると、最初の属性のみが評価されます。

パラメーター	使用方法	タイプ	説明
UserID	オプション	ストリング	検索するユーザーのユーザー ID。
Profile	オプション	ストリング	検索するユーザーのプロファイル。
LastName	オプション	ストリング	検索するユーザーのラストネーム。
FirstName	オプション	ストリング	検索するユーザーのファーストネーム。
MiddleName	オプション	ストリング	検索するユーザーのミドルネーム。
Email	オプション	ストリング	ユーザーの電子メール・アドレス。
Company	オプション	ストリング	ユーザーが属する会社。
DisplayName	オプション	ストリング	ユーザーの表示名。
Secretary	オプション	ストリング	ユーザーの秘書。
Assistant	オプション	ストリング	ユーザーのアシスタント。
Manager	オプション	ストリング	ユーザーの管理者。
Department	オプション	ストリング	ユーザーが属する部門。
Phone	オプション	ストリング	ユーザーの電話番号。
Fax	オプション	ストリング	ユーザーの FAX 番号番号。
Gender	オプション	ストリング	ユーザーが男性であるか、女性であるか。
Timezone	オプション	ストリング	ユーザーが配置されている場所の時間帯。
PreferredLanguage	オプション	ストリング	ユーザーの希望する言語。

Role Members

この動詞は、ビジネス・プロセスのロールに関連付けられているユーザーを検索するときに使用します。

パラメーター	使用方法	タイプ	説明
RoleName	必須	ストリング	検索するユーザーのロール名。
IncludeNestedRoles	必須	ブール	ネストされたロールを照会で考慮するかどうかを指定します。
Domain	オプション	ストリング	ロールが属するドメイン。このパラメーターは、照会をディレクトリーのサブセットに限定する場合に使用します。
AlternativeRoleName1	オプション	ストリング	ユーザーの追加のロール名。
AlternativeRoleName2	オプション	ストリング	ユーザーの追加のロール名。

Users

この動詞は、名前で認識されているユーザーについてのスタッフ照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
Name	必須	ストリング	検索するユーザーの名前。
AlternativeName1	オプション	ストリング	追加のユーザー名。このパラメーターは、複数のユーザーを検索するときに使用します。
AlternativeName2	オプション	ストリング	追加のユーザー名。このパラメーターは、複数のユーザーを検索するときに使用します。

Users by user ID

この動詞は、ユーザー ID が認識されているユーザーについてのスタッフ照会を定義するときに使用します。ショート・ネームを使用して、「wpsadmin」などの値を指定します。この動詞には、スタッフ・リポジトリーにアクセスするという意味はありません。

パラメーター	使用方法	タイプ	説明
UserID	必須	ストリング	検索するユーザーのユーザー ID。コンテキスト変数およびカスタム・プロパティー (例: %htm:task.potentialStarters%) をサポートします。
AlternativeID1	オプション	ストリング	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
AlternativeID2	オプション	ストリング	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。

Users by user ID without Named Users

この動詞は、ユーザー ID が認識されているユーザー (明示的に指定されたユーザー ID を持つユーザーを除く) についてのスタッフ照会を定義するときに使用します。ショート・ネームを使用して、「wpsadmin」などの値を指定します。この動詞には、スタッフ・リポジトリーにアクセスするという意味はありません。

パラメーター	使用方法	タイプ	説明
UserID	必須	ストリング	検索するユーザーのユーザー ID。コンテキスト変数およびカスタム・プロパティー (例: %htm:task.potentialStarters%) をサポートします。
AlternativeID1	オプション	ストリング	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
AlternativeID2	オプション	ストリング	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
NamedUsers	必須	ストリング	ユーザー ID のリストから除外するユーザーのユーザー ID。コンテキスト変数およびカスタム・プロパティー (例: %wf:activity(...).owner%) をサポートします。

システム・ユーザー・リポジトリーの定義済みスタッフ動詞:

Business Process Choreographer のシステム・ユーザー・リポジトリー・スタッフ・プラグインで使用する定義済みスタッフ動詞およびパラメーターについて説明します。

WebSphere Integration Developer でスタッフ動詞を使用して、ビジネス・プロセス内またはヒューマン・タスク内のスタッフの割り当てをモデル化することができます。このスタッフ動詞は、モデル化およびデプロイメント中にスタッフ・リポジトリーについて実行できる一連の照会に変換されます。このセクションでは、次の定義済みスタッフ動詞用のパラメーターを示します。

- Group
- Everybody
- Nobody
- Users
- Users by user ID
- Users by user ID without Named Users

Group

この動詞は、グループのメンバーを許可する照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
GroupId	必須	ストリング	許可するユーザー・グループの名前。

Everybody

この動詞は、WebSphere Process Server によって認証されたすべてのユーザーに作業項目を割り当てるときに使用します。この動詞はパラメーターを取りません。

Nobody

インライン・タスクの場合は、ビジネス・プロセス管理者のみにアクセス権があります。 スタンドアロン・タスクの場合は、ヒューマン・タスク管理者のみにアクセス権があります。 さらに、Business Flow Manager API を使用する場合は、BPESystemAdministrator のロール・メンバーにアクセス権があり、 Human Task

Manager API の場合は、TaskSystemAdministrator のロール・メンバーにアクセス権があります。 この動詞はパラメーターを取りません。

Users

この動詞は、名前で認識されているユーザーについてのスタッフ照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
Name	必須	ストリング	検索するユーザーの名前。
AlternativeName1	オプション	ストリング	追加のユーザー名。このパラメーターは、複数のユーザーを検索するときに使用します。
AlternativeName2	オプション	ストリング	追加のユーザー名。このパラメーターは、複数のユーザーを検索するときに使用します。

Users by user ID

この動詞は、ユーザー ID が認識されているユーザーについてのスタッフ照会を定義するときに使用します。ショート・ネームを使用して、「wpsadmin」などの値を指定します。この動詞には、スタッフ・リポジトリーにアクセスするという意味はありません。

パラメーター	使用方法	タイプ	説明
UserID	必須	ストリング	検索するユーザーのユーザー ID。コンテキスト変数およびカスタム・プロパティー (例: %htm:task.potentialStarters%) をサポートします。
AlternativeID1	オプション	ストリング	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
AlternativeID2	オプション	ストリング	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。

Users by user ID without Named Users

この動詞は、ユーザー ID が認識されているユーザー (明示的に指定されたユーザー ID を持つユーザーを除く) についてのスタッフ照会を定義するときに使用します。ショート・ネームを使用して、「wpsadmin」などの値を指定します。この動詞には、スタッフ・リポジトリーにアクセスするという意味はありません。

パラメーター	使用方法	タイプ	説明
UserID	必須	ストリング	検索するユーザーのユーザー ID。コンテキスト変数およびカスタム・プロパティー (例: %htm:task.potentialStarters%) をサポートします。
AlternativeID1	オプション	ストリング	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。

パラメーター	使用方法	タイプ	説明
AlternativeID2	オプション	ストリング	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
NamedUsers	必須	ストリング	ユーザー ID のリストから除外するユーザーのユーザー ID。コンテキスト変数およびカスタム・プロパティー (例: %wf:activity(...).owner%) をサポートします。

ユーザー・レジストリーの定義済みスタッフ動詞:

Business Process Choreographer のユーザー・レジストリー・スタッフ・プラグインで使用する定義済みスタッフ動詞およびパラメーターについて説明します。

WebSphere Integration Developer でスタッフ動詞を使用して、ビジネス・プロセス内またはヒューマン・タスク内のスタッフの割り当てをモデル化することができます。このスタッフ動詞は、モデル化およびデプロイメント中にスタッフ・リポジトリーについて実行できる一連の照会に変換されます。このセクションでは、次の定義済みスタッフ動詞用のパラメーターを示します。

- Group
- Everybody
- Group Members
- Group Members without Named Users
- Group Search
- Native Query
- Nobody
- Person Search
- Users
- Users by user ID
- Users by user ID without Named Users

Group

この動詞は、グループ作業項目で使用するグループの名前を検索する照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
GroupName	必須	ストリング	検索するグループの名前。

Everybody

この動詞は、WebSphere Process Server によって認証されたすべてのユーザーに作業項目を割り当てるときに使用します。この動詞はパラメーターを取りません。

Group Members

この動詞は、グループのメンバーを検索する照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
GroupName	必須	ストリング	検索するユーザーのグループ名。

パラメーター	使用方法	タイプ	説明
AlternativeGroupName1	オプション	ストリング	ユーザーが属することのできる追加のグループ。
AlternativeGroupName2	オプション	ストリング	ユーザーが属することのできる追加のグループ。

Group Members without Named Users

この動詞は、グループのメンバーのうち、明示的に指定されたユーザーを除くすべてのメンバーを検索する照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
GroupName	必須	ストリング	検索するユーザーのグループ名。実行時に評価されるカスタム・プロパティーをサポートします。
NamedUsers	必須	ストリング	検索されたグループ・メンバーのリストから除外するユーザーのユーザー ID。コンテキスト変数およびカスタム・プロパティー (例: `%htm:task.originator%`) をサポートします。

Group Search

この動詞は、属性の一致に基づいてグループを検索し、そのグループのメンバーを取得するときに使用します。属性を 1 つ設定する必要があります。複数の属性を設定すると、最初の属性のみが評価されます。

パラメーター	使用方法	タイプ	説明
GroupID	オプション	ストリング	検索するユーザーのグループ ID。

Native Query

この動詞は、ディレクトリー固有のパラメーターに基づいた固有の照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
QueryTemplate	必須	ストリング	照会に使用する照会テンプレート。デフォルトのマッピング・ファイルは、テンプレートの search、user、および usersOfGroup をサポートします。
Query	必須	ストリング	照会を指定します。`%wf:process.starter%` などのコンテキスト変数およびカスタム・プロパティーを使用できます。照会のタイプは、照会テンプレートによって異なります。 • 検索テンプレート: 検索パターン • ユーザー・テンプレート: ユーザー名 • usersOfGroup: グループ名

パラメーター	使用方法	タイプ	説明
AdditionalParameter1	オプション	ストリング	照会を指定します。コンテキスト変数 (例: `%wf:process.starter%`) を使用できます。サポートされる値は、`group` および `user` です。
AdditionalParameter2	オプション	ストリング	この動詞は、追加パラメーターを指定するときに使用します。
AdditionalParameter3	オプション	ストリング	この動詞は、追加パラメーターを指定するときに使用します。 デフォルトのマッピング XSLT ファイルを使用する場合、このパラメーターはサポートされません。
AdditionalParameter4	オプション	ストリング	この動詞は、追加パラメーターを指定するときに使用します。 デフォルトのマッピング XSLT ファイルを使用する場合、このパラメーターはサポートされません。
AdditionalParameter5	オプション	ストリング	この動詞は、追加パラメーターを指定するときに使用します。 デフォルトのマッピング XSLT ファイルを使用する場合、このパラメーターはサポートされません。

Nobody

インライン・タスクの場合は、ビジネス・プロセス管理者のみにアクセス権があります。 スタンドアロン・タスクの場合は、ヒューマン・タスク管理者のみにアクセス権があります。 さらに、Business Flow Manager API を使用する場合は、BPESystemAdministrator のロール・メンバーにアクセス権があり、 Human Task Manager API の場合は、TaskSystemAdministrator のロール・メンバーにアクセス権があります。 この動詞はパラメーターを取りません。

Person Search

この動詞は、属性の一致に基づいて人を検索するときに使用します。属性を 1 つ設定する必要があります。複数の属性を設定すると、最初の属性のみが評価されます。

パラメーター	使用方法	タイプ	説明
UserID	オプション	ストリング	検索するユーザーのユーザー ID。

Users

この動詞は、名前で認識されているユーザーについてのスタッフ照会を定義するときに使用します。

パラメーター	使用方法	タイプ	説明
Name	必須	ストリング	検索するユーザーの名前。
AlternativeName1	オプション	ストリング	追加のユーザー名。このパラメーターは、複数のユーザーを検索するときに使用します。
AlternativeName2	オプション	ストリング	追加のユーザー名。このパラメーターは、複数のユーザーを検索するときに使用します。

Users by user ID

この動詞は、ユーザー ID が認識されているユーザーについてのスタッフ照会を定義するときに使用します。ショート・ネームを使用して、「wpsadmin」などの値を指定します。この動詞には、スタッフ・リポジトリーにアクセスするという意味はありません。

パラメーター	使用方法	タイプ	説明
UserID	必須	ストリング	検索するユーザーのユーザー ID。コンテキスト変数およびカスタム・プロパティー (例: %htm:task.potentialStarters%) をサポートします。
AlternativeID1	オプション	ストリング	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
AlternativeID2	オプション	ストリング	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。

Users by user ID without Named Users

この動詞は、ユーザー ID が認識されているユーザー (明示的に指定されたユーザー ID を持つユーザーを除く) についてのスタッフ照会を定義するときに使用します。ショート・ネームを使用して、「wpsadmin」などの値を指定します。この動詞には、スタッフ・リポジトリーにアクセスするという意味はありません。

パラメーター	使用方法	タイプ	説明
UserID	必須	ストリング	検索するユーザーのユーザー ID。コンテキスト変数およびカスタム・プロパティー (例: %htm:task.potentialStarters%) をサポートします。
AlternativeID1	オプション	ストリング	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
AlternativeID2	オプション	ストリング	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
NamedUsers	必須	ストリング	ユーザー ID のリストから除外するユーザーのユーザー ID。コンテキスト変数およびカスタム・プロパティー (例: %wf:activity(...).owner%) をサポートします。

新規カスタム動詞のインプリメント

ここでは、新規のスタッフ動詞をスタッフ・サポート・サービス・インフラストラクチャーに追加し、ビジネス・プロセスやヒューマン・タスクのモデリング時にこれらの動詞を WebSphere Integration Developer で使用できるようにする方法について説明します。

新規のスタッフ動詞の指定内容は、WebSphere Integration Developer インストール環境の一部である VerbSet.xml ファイルに追加する必要があります。 例えば、新規の動詞 Mentor of Employee の場合は、以下のようになります。

```
  <vs:DefineVerb name='Mentor of Employee'>
    <vs:Description>Assigns the mentor of an employee.
Supported by sample XSLT files for:
 - LDAP
    </vs:Description>
    <vs:Mandatory>
      <vs:Parameter>
        <vs:Name>EmployeeName</vs:Name>
        <vs:Type>xsd:string</vs:Type>
      </vs:Parameter>
    </vs:Mandatory>
    <vs:Optional>
      <vs:Parameter>
        <vs:Name>Domain</vs:Name>
        <vs:Type>xsd:string</vs:Type>
      </vs:Parameter>
    </vs:Optional>
  </vs:DefineVerb>
```

LDAP 変換ファイルのディスパッチャー・セクションに新規の動詞を追加する必要があります。以下に例を示します。

```
    <xsl:choose>
      ...
      <xsl:when test="$verb='Mentor of Employee'">
        <xsl:call-template name="MentorOfEmployee"/>
      ...
   </xsl:choose>
```

LDAP 変換ファイルには、マッピングをインプリメントするテンプレートも追加する必要があります。以下に例を示します。

```
<!-- Begin template MentorOfEmployee -->
  <xsl:template name="MentorOfEmployee">
    <sldap:staffQueries>
      <xsl:attribute name="threshold">
        <xsl:value-of select="$Threshold"/>
      </xsl:attribute>

      <sldap:intermediateResult>
        <xsl:attribute name="name">mentorvariable</xsl:attribute>
        <sldap:user>
          <xsl:attribute name="dn">
            <xsl:value-of select="staff:parameter[@id='EmployeeName']"/>
          </xsl:attribute>
          <xsl:attribute name="attribute">mentor</xsl:attribute>
          <xsl:attribute name="objectclass">inetOrgPerson</xsl:attribute>
        </sldap:user>
      </sldap:intermediateResult>

      <sldap:user>
        <xsl:attribute name="dn">%mentorvariable%</xsl:attribute>
        <xsl:attribute name="attribute">uid</xsl:attribute>
```

```
        <xsl:attribute name="objectclass">inetOrgPerson</xsl:attribute>
      </sldap:user>
    </sldap:staffQueries>
  </xsl:template>
<!-- End template MentorOfEmployee -->
```

マッピングによって、LDAP 固有の有効な照会が生成されること検証します。

LDAP 変換ファイルの適合

LDAP 変換 XSL ファイルを LDAP スキーマに適合させる方法について説明します。

デフォルトの LDAPTransformation.xsl ファイルは、WebSphere が前提とするデフォルトの LDAP スキーマのエレメントを使用する LDAP 照会に、事前定義のスタッフ動詞をマップします。このスキーマでは、以下のことを前提にしています。

- グループ項目の LDAP オブジェクト・クラスは groupOfName である。
- グループのメンバー DN を含むグループ項目属性は member である。
- 個人項目の LDAP オブジェクト・クラスは inetOrgPerson である。
- 個人項目のログイン ID を含む属性は uid である。
- 個人の E メール・アドレスを含む個人項目属性は mail である。
- 個人の管理者の識別名を含む個人項目属性は manager である。

ご使用の LDAP スキーマに、異なるオブジェクト・クラスや属性名がある場合は、LDAP 変換ファイル内でこれらの設定を変更する必要があります。デフォルトの LDAPTransformation.xsl ファイルの場合は、設定の変更を以下に示すファイルの変数宣言部分で実行できます。

```
  <xsl:variable name="DefaultGroupClass">groupOfNames</xsl:variable>
  <xsl:variable name="DefaultGroupClassMemberAttribute">member</xsl:variable>

  <xsl:variable name="DefaultPersonClass">inetOrgPerson</xsl:variable>
  <xsl:variable name="DefaultUserIDAttribute">uid</xsl:variable>
  <xsl:variable name="DefaultMailAttribute">mail</xsl:variable>
  <xsl:variable name="DefaultManagerAttribute">manager</xsl:variable>
```

以下の例に示すように、個々のスタッフ動詞を変換する XSL テンプレート内で変更を適用できます。

例: DepartmentMembers

個人項目のオブジェクト・クラスを ePerson に変更、ログイン ID 属性を cn に変更

```
<sldap:StaffQueries>
  <xsl:attribute name="threshold">
    <xsl:value-of select="$Threshold">
  </xsl:attribute>

  <sldap:search>
...
    <sldap:attribute>
      <xsl:attribute name="name">cn</xsl:attribute>
      <xsl:attribute name="objectclass">ePerson</xsl:attribute>
      <xsl:attribute name="usage">simple</xsl:attribute>
```

```
    </sldap:attribute>

  </sldap:search>
</sldap:StaffQueries>
```

例: GroupMembers

グループ項目のオブジェクト・クラスを groupOfUniqueNames に変更、メンバーの DN リストを含むグループ項目属性を uniqueMember に変更、ログイン ID を含む個人項目属性を cn に変更

```
<sldap:usersOfGroup>
...

  <sldap:attribute>
    <xsl:attribute name="name">uniqueMember</xsl:attribute>
    <xsl:attribute name="objectclass">groupOfUniqueNames</xsl:attribute>
    <xsl:attribute name="usage">recursive</xsl:attribute>
  </sldap:attribute>

  ...
  <sldap:attribute>
    <xsl:attribute name="name">cn</xsl:attribute>
    <xsl:attribute name="objectclass">inetOrgPerson</xsl:attribute>
    <xsl:attribute name="usage">simple</xsl:attribute>
  </sldap:attribute>

</sldap:usersOfGroup>
```

例: GroupMembersWithoutFilteredUsers

LDAP フィルター演算子を >= に変更

```
<sldap:StaffQueries>
  <sldap:usersOfGroup>
    ...
  </sldap:usersOfGroup>

  <sldap:intermediateResult>
    <xsl:attribute name="name">filteredusers</xsl:attribute>
    <sldap:search>
      <xsl:attribute name="filter">
        <xsl:value-of select="staff:parameter[@id='FilterAttribute']"/>
         >=
        <xsl:value-of select="staff:parameter[@id='FilterValue']"/>
      </xsl:attribute>
     ...
    <sldap:search>
     ...

  </sldap:intermediateResult>
   ...
</sldap:StaffQueries>
```

例: GroupSearch

検索属性を MyType に変更、オブジェクト・クラスを mypersonclass に変更、ログイン ID を含む属性を myuid に変更

```
<sldap:StaffQueries>
 ...
  <sldap:search>
    <xsl:attribute name="filter">
      (&amp;
       ...
```

```
      <xsl:if test="staff:parameter[@id='MyType']!="">
       (<xsl:value-of select="$GS_Type"/>=
        <xsl:value-of select=staff:parameter[@id='Type']"/>)
      </xsl:if>
      )
       ...
    </xsl:attribute>

    <sldap:attribute>
      <xsl:attribute name="name">myuid</xsl:attribute>
      <xsl:attribute name="objectclass">mypersonclass</xsl:attribute>
      <xsl:attribute name="usage">simple</xsl:attribute>
    </sldap:attribute>
     ...
  <sldap:search>
</sldap:StaffQueries>
```

例: 従業員の管理者

管理者 DN を含む属性を managerentry に変更、管理者ログイン ID 属性のソースを name に変更

```
<sldap:StaffQueries>

  <sldap:intermediateResult>
   ...
    <sldap:user>
     ...
      <xsl:attribute name="name">managerentry</xsl:attribute>
       ...
     </sldap:user>
   </sldap:intermediateResult>

  <sldap:user>
   ...
    <xsl:attribute name="name">name</xsl:attribute>
     ...
  </sldap:user>
</sldap:StaffQueries>
```

例: PersonSearch

検索属性を MyAttribute に変更、オブジェクト・クラスを mypersonclass に変更、戻り属性のソースを myuid に変更

```
<sldap:StaffQueries>
 ...
  <sldap:search>
    <xsl:attribute name="filter">
      (&amp;
       ...
      <xsl:if test="staff:parameter[@id='MyAttribute']!="">
       (<xsl:value-of select="$PS_UserID"/>=
        <xsl:value-of select=staff:parameter[@id='UserID']"/>)
      )
      </xsl:if>
       ...
    </xsl:attribute>

    <sldap:attribute>
      <xsl:attribute name="name">myuid</xsl:attribute>
      <xsl:attribute name="objectclass">mypersonclass</xsl:attribute>
      <xsl:attribute name="usage">simple</xsl:attribute>
```

```
    </sldap:attribute>
     ...
  </sldap:search>
</sldap:StaffQueries>
```

例: Users

戻り属性のソースを myuid に変更、オブジェクト・クラスを mypersonclass に変更

```
<sldap:user>
 ...
  <xsl:attribute name="attribute">myuid</xsl:attribute>
  <xsl:attribute name="objectclass">mypersonclass</xsl:attribute>
</sldap:user>
```

概要: Business Process Choreographer Explorer の構成

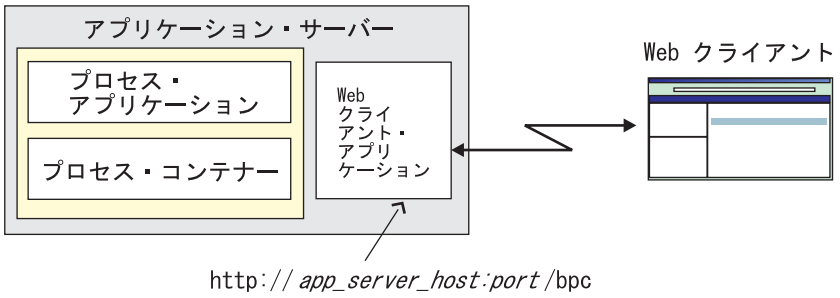
Business Process Choreographer Explorer は、プロセス管理およびタスク処理用のユーザー・インターフェースを提供します。これは、JavaServer Faces (JSF) テクノロジーおよび Business Process Choreographer Explorer コンポーネントに基づく Java 2 Enterprise Edition (J2EE) Web アプリケーションです。

- 『Business Process Choreographer Explorer について』
- 99 ページの『Business Process Choreographer Explorer の構成』

Business Process Choreographer Explorer について

Business Process Choreographer Explorer は、ビジネス・プロセスおよびヒューマン・タスクとの対話を目的として汎用の Web ユーザー・インターフェースをインプリメントする Web アプリケーションです。

Business Process Choreographer Explorer は、ビジネス・プロセス・コンテナーとヒューマン・タスク・コンテナーの両方がインストールされているアプリケーション・サーバー上またはクラスター上にインストールできます。複数のアプリケーション・サーバーまたはクラスター上でビジネス・プロセス・アプリケーションまたはヒューマン・タスク・アプリケーションを扱う仕事をする場合は、1 つのサーバーまたはクラスター上に Business Process Choreographer Explorer を構成すれば十分です。Business Process Choreographer が構成されているすべてのアプリケーション・サーバーまたはクラスター上に構成する必要はありません。



単一の Business Process Choreographer Explorer は複数の Business Process Choreographer 構成に接続できません。同じサーバー上に、Business Process Choreographer Explorer の複数のインスタンスを構成できます。

Business Process Choreographer Explorer を開始する場合、ユーザー・インターフェースに表示されるオブジェクト、および実行できるアクションは、所属するユーザー・グループとそのグループに与えられた権限によって異なります。例えば、管理者の場合は、配置されたビジネス・プロセスおよびタスクの運用を平滑化する責任を負います。管理者は、プロセスやタスクのテンプレート、プロセス・インスタンス、タスク・インスタンス、およびそれらの関連オブジェクトに関する情報を表示できます。これらのオブジェクトを操作することもできます。例えば、新規プロセス・インスタンスの開始、タスクの作成と開始、失敗したアクティビティーの修復と再始動、作業項目の管理、完了したプロセス・インスタンスおよびタスク・インスタンスの削除を実行できます。ただし、ユーザーの場合は、割り当てられたタスクのみを表示し、操作することができます。

Business Process Choreographer Explorer の構成

スクリプトを使用して Business Process Choreographer Explorer を構成するには、このタスクを使用します。

ビジネス・プロセス・コンテナーおよびヒューマン・タスク・コンテナーを構成済みです。

以下のいずれかの場合があります。

- Business Process Choreographer Explorer をまだインストールしていません。
- 既存の Business Process Choreographer 構成に追加しようとしています。
- 他のインスタンスがすでに実行しているサーバーに Business Process Choreographer Explorer の別のインスタンスを追加しようとしています。

1. Business Process Choreographer ディレクトリーに移動し、`clientconfig.jacl` スクリプトを起動します。

```
cd install_root/ProcessChoreographer/config
../../bin/wsadmin.sh -f clientconfig.jacl
  ( [-user userName][-password password]|[-conntype NONE])
    [-profileName profileName]
    [-hostName explorerVirtualHostname]
    [-explorerHost explorerURL]
    [-precompileJSPs { yes | no }]
    [-remoteNodeName nodeName]
    [-remoteServerName serverName]
    [-remoteClusterName clusterName]
    [-contextRootExplorer explorerContextRoot]
```

z/OS では、次のコマンドを実行します。

```
cd install_root/ProcessChoreographer/config
../../bin/wsadmin.sh -f clientconfig.jacl
  ( [-user userName][-password password]|[-conntype NONE])
    [-profileName profileName]
    [-hostName explorerVirtualHostname]
    [-explorerHost explorerURL]
    [-precompileJSPs { yes | no }]
    [-remoteNodeName nodeName]
    [-remoteServerName serverName]
    [-remoteClusterName clusterName]
    [-contextRootExplorer explorerContextRoot]
```

スタンドアロン・サーバー環境の場合:

- `-conntype NONE` オプションを指定するのはアプリケーション・サーバーが稼働していない場合に限定します。
- サーバーが稼働していてグローバル・セキュリティーが使用可能な場合は、`-user` および `-password` オプションを指定します。
- デフォルト・プロファイルを構成していない場合は、`-profileName` オプションを追加します。

Network Deployment 環境の場合:

- スクリプトをデプロイメント・マネージャー・ノードで実行します。
- `-conntype NONE` オプションを指定するのはデプロイメント・マネージャーが稼働していない場合に限定します。
- グローバル・セキュリティーが使用可能な場合、`-user` および `-password` オプションを組み込んでください。
- デフォルト・プロファイルを構成していない場合は、`-profileName` オプションを追加します。

以下のパラメーターを指定することもできます。

node *nodeName*

nodeName は、Business Process Choreographer を構成するノードの名前です。ノードが 1 つだけでサーバーが 1 つしかない場合、このパラメーターはオプションです。

server *serverName*

serverName は、Business Process Choreographer を構成するサーバーの名前です。ノードが 1 つだけでサーバーが 1 つしかない場合、このパラメーターはオプションです。

cluster *clusterName*

clusterName は、Business Process Choreographer を構成するクラスターの名前です。スタンドアロン・サーバー環境の場合、およびノードとサーバーを指定した場合は、このオプションを指定しないでください。このオプションは、非対話式に使用することはできません。

hostName *explorerVirtualHostname*

ここで *explorerVirtualHostname* は、Business Process Choreographer Explorer を実行する仮想ホストです。デフォルト値は `default_host` です。

explorerHost *explorerURL*

ここで *explorerURL* は、Business Process Choreographer Explorer の URL です。このパラメーターを非クラスター環境で指定しない場合は、デフォルト値が計算されます (例えば、http://localhost:9080)。このパラメーターの値は、Human Task Manager の EscalationMail.ClientDetailURL カスタム・プロパティーに使用されます。

precompileJSPs { yes | no }

Java Server Pages (JSPs) をプリコンパイルするかどうかを識別します。

remoteNodeName *nodeName*

ローカル側の Business Process Choreographer Explorer に接続しない場合は、このパラメーターと remoteServerName を使用します。node と server

パラメーターまたは cluster パラメーターによって指定される Business Process Choreographer サーバーに接続する場合は、このパラメーターを使用しないでください。

remoteServerName *serverName*
ローカル側の Business Process Choreographer Explorer に接続しない場合は、このパラメーターと remoteNodeName を使用します。node と server パラメーターまたは cluster パラメーターによって指定される Business Process Choreographer サーバーに接続する場合は、このパラメーターを使用しないでください。

remoteClusterName *clusterName*
ローカル側の Business Process Choreographer Explorer に接続せず、remoteNodeName および remoteServerName を指定しない場合は、このパラメーターを使用します。node と server パラメーターまたは cluster パラメーターによって指定される Business Process Choreographer サーバーに接続する場合は、このパラメーターを使用しないでください。

contextRootExplorer *contextRootExplorer*
ここで *contextRootExplorer* は、Business Process Choreographer Explorer のコンテキスト・ルートです。デフォルト値は `/bpc` であり、`http://host:port/contextRootExplorer` のデフォルト URL になります。コンテキスト・ルートは WebSphere セル内で固有のものでなければなりません。

2. `clientconfig.jacl` スクリプトにより、パラメーターとして指定されなかった必要な情報の入力が促されます。
3. **オプション:** 構成に問題がある場合、`clientconfig.jacl` スクリプトによって書き込まれたログ・ファイルを確認します。このログは、`profiles/profileName/logs/clientconfig.log` ファイルにあります。このディレクトリーには、問題に関する詳細な情報が含まれる可能性のある `wsadmin.traceout` ファイルも格納されています。

Business Process Choreographer Explorer が構成されて、使用する準備が完了しました。

Business Process Choreographer Explorer を開始します。

Business Process Choreographer Observer インフラストラクチャーの構成

Business Process Choreographer Observer および Event Collector を構成する方法について説明します。これにより、ビジネス・プロセスおよびヒューマン・タスクの実行状態を監視できます。

Business Process Choreographer は、管理コンソールまたは bpeconfig スクリプトを使用して構成済みですが、Business Process Choreographer Observer は未構成です。WAS_HOME 環境変数は、アプリケーション・サーバーのインストール・ディレクトリーに設定する必要があります。管理権限を持つユーザー ID (例えば `root`) を使用してログオンする必要があります。

このトピックでは、ND 環境で Business Process Choreographer Observer を構成する方法、および拡張構成については説明しません。Business Process Choreographer Observer の構成と使用についての詳細は、サポート文書 7008553 を参照してください。

1. Event Collector アプリケーションおよびデータベースを構成します。 103 ページの『Business Process Choreographer Event Collector の構成』を実行します。
2. 監視者アプリケーションを構成します。 107 ページの『Business Process Choreographer Observer の構成』を実行します。

Business Process Choreographer Observer がインストールおよび構成されています。

Business Process Choreographer が作動していることを確認します。

Business Process Choreographer Observer について

Business Process Choreographer Observer を使用して、完了したプロセスおよびタスクに関するレポートを作成できます。また、これを使用して、実行中のプロセスおよびタスクの状況を表示することもできます。

Business Process Choreographer Observer の構成と使用の詳細については (ND 環境におけるものも含めて)、サポート文書 7008553 を参照してください。

Business Process Choreographer Observer は以下に示す複数の方法で構成できます。

- 既存の Business Process Choreographer 構成がある場合は、 101 ページの『Business Process Choreographer Observer インフラストラクチャーの構成』の説明に従って、構成スクリプトを使用できます。
- 「プロファイル作成」ウィザードを使用してサンプルの Business Process Choreographer 構成を作成するオプションを選択する場合、Business Process Choreographer Observer および Event Collector が構成され、 Common Event Infrastructure (CEI) ロギングが Business Process Choreographer 状態監視者で使用可能になり、必要なデータベース・スキーマが、BPEDB という名前の Business Process Choreographer Cloudscape データベース内に作成されます。
- 管理コンソールを使用して Business Process Choreographer インストール・ウィザードを実行する場合は、 Business Process Choreographer Observer を構成するオプションがあります。

Business Process Choreographer Observer は、次の 2 つの J2EE エンタープライズ・アプリケーションに基づいています。

- Business Process Choreographer Event Collector
- Business Process Choreographer Observer

これらは同じデータベースを使用します。 Event Collector は CEI バスからイベント情報を読み取り、Business Process Choreographer Observer データベースに保管します。未加工のイベント・データは、定期的に Business Process Choreographer Observer からの照会に適したフォーマットに変換されます。Business Process Choreographer が Cloudscape、DB2、または Oracle を使用する場合、 Business Process Choreographer Observer は同じデータベース・インスタンスを使用できます。それ以外を使用する場合は、新規データベースを作成する必要があります。

Business Process Choreographer Observer データベースは、イベント・データを保管するデータベース表の集合です。表は、既存データベースまたは Business Process Choreographer Observer 専用の新規データベースの中に作成できます。 Business Process Choreographer Observer をインストールする前に、データベースが使用可能になっており、このデータベースをポイントするように、アプリケーション・サーバーで XA データ・ソースが構成されている必要があります。データ・ソースは手動で作成する必要があります。Observer の表を作成するには、 Observer 構成スクリプトを使用するか、*install_root*¥dbscripts¥ProcessChoreographer¥*database_type* にあるスクリプトを使用して、手動でスキーマを作成できます。

ご使用のデータベース・システムが DB2 または Oracle の場合は、構成ツールを使用する前に、 `createTablespace_Observer.sql` スクリプトを実行する必要があります。

Business Process Choreographer Event Collector の構成

ここでは、スクリプトを使用して、Business Process Choreographer Observer に必要な Event Collector およびデータベース表を構成する方法について説明します。

- データベースが使用可能です。
- CLASSPATH 環境変数には、データベースの JDBC ドライバーが含まれています。
- アプリケーション・サーバーの管理コンソールを使用して、XA データ・ソースを手動で作成する必要があります。データ・ソースはデータベースをポイントする必要があります。

  注: z/OS で、タイプ 2 の XA 以外のプロバイダーを使用する場合は、XA データ・ソースを作成しないでください。 XA データ・ソースは、タイプ 4 の XA プロバイダーのみが対象です。

1. 構成スクリプトがある Business Process Choreographer サブディレクトリーに変更します。

   以下のコマンドを入力します。

   ```
   cd install_root/ProcessChoreographer/config
   ```

2. スクリプトを開始して、Event Collector をセットアップします。

   以下のコマンドを入力します。

   ```
   setupEventCollector.sh [-conntype SOAP | RMI | JMS | NONE]
        ( [-node nodeName] -server serverName ) | ( -cluster clusterName )
        [ -remove [-silent]]
   ```

   各部の意味は、次のとおりです。

   conntype SOAP | RMI | JMS | NONE
   wsadmin ツールが使用する接続モード。

   node *nodeName*
   ノードの名前。このパラメーターはオプションです。デフォルト値はローカル・ノードです。

server *serverName*
サーバーの名前。オプション `-conntype none` を指定しない場合、このパラメーターはオプションです。

cluster *clusterName*
オプション `-conntype none` を指定しない場合、このパラメーターはオプションです。

remove
このオプションを指定して Event Collector を除去します。このオプションを指定しない場合、デフォルトで Event Collector が構成されます。

silent このオプションは、remove オプションと一緒にしか使用できません。これにより、スクリプトはプロンプトを出力しないようになります。このパラメーターはオプションです。

注:

スタンドアロン・サーバー環境の場合:

- `-conntype NONE` オプションを指定するのはアプリケーション・サーバーが稼働していない場合に限定します。
- サーバーが稼働していてグローバル・セキュリティーが使用可能な場合は、`-username` および `-password` オプションを指定します。
- デフォルト・プロファイルを構成していない場合は、`-profileName` オプションを追加します。

「コマンド・メニュー (Commands Menu)」が表示されます。

```
Commands Menu

  1) Prepare a database for the Event Collector
  2) Install the Event Collector application
  3) Remove the Event Collector application and related objects
  4) Change configuration settings of an installed Event Collector

  0) Exit Commands Menu
```

注: ローカル・モードでスクリプトを開始した場合 (`-conntype none` パラメーターを指定し、サーバーとクラスターは指定しない)、0 および 1 のメニュー項目のみが表示されます。

3. オプション 1 を選択して、Event Collector 用にデータベースを準備します。
   a. 「コマンド・メニュー (Commands Menu)」が表示されます。

```
Select the type of your DBMS :
  'd' ... DB2
  'c' ... Cloudscape
  '7' ... DB2 V7 on z/OS
  '8' ... DB2 V8 on z/OS
  'o' ... Oracle
  'x' ... Exit
Your selection : [c]
```

「コマンド・メニュー (Commands Menu)」が表示されます。

4. Select your database type.
   - DB2 の場合、`d` を入力します。

- Cloudscape の場合、c を入力します。
- DB2 V7 on z/OS の場合、7 を入力します。
- DB2 V8 on z/OS の場合、8 を入力します。
- Oracle の場合、o を入力します。

5. データベースの設定を入力します。
   - DB2 の場合は、以下の項目を入力します。
     a. データベース名または別名 (BPEDB など)。
     b. データベースに接続するためのユーザー ID およびパスワード (db2admin および関連するパスワードなど)。
     c. ユーザー ID のパスワード。
     d. データベース・オブジェクトに使用するデータベース・スキーマ。存在しないスキーマを指定した場合は、作成されます。スペース文字を入力するかフィールドを空にしておくと、a. で指定されたユーザー ID のスキーマが使用されます。
   - Cloudscape の場合は、以下の項目を入力します。
     a. d:¥w¥p¥profiles¥Srv01¥databases¥BPEDB などの、データベースへの完全修飾パス。
     b. データベース・オブジェクトに使用するデータベース・スキーマ。存在しないスキーマを指定した場合は、作成されます。スペース文字を入力するかフィールドを空にしておくと、デフォルトのスキーマが使用されます (通常は APP)。
     c. サーバーを停止するように指示される場合は、停止してから、c を押して続行します。
   - Oracle の場合は、以下の項目を入力します。
     a. データベース名 (BPEDB など)。
     b. データベースが存在するホスト名 (localhost など)。
     c. Oracle リスナーが listen しているポート番号 (1521 など)。
     d. データベースに接続するための ユーザー ID (system など)。
     e. ユーザー ID のパスワード。

   接続を検査した後に、データベースが準備されます。
6. エラーがないかどうか検査します。 エラーが発生した場合は、プロファイル・ディレクトリーの logs サブディレクトリーにあるログ・ファイル setupEventCollector.log を検査します。例えば Windows では、プロファイル名が *myServer* で、プロファイルが *install_root*¥profiles に保管されている場合、ログ・ファイルは *install_root*¥profiles¥*myServer*¥logs に置かれます。
7. Event Collector アプリケーションをインストールします。 「コマンド・メニュー (Commands Menu)」が表示されます。

```
Commands Menu

  1) Prepare a database for the Event Collector
  2) Install the Event Collector application
```

```
3) Remove the Event Collector application and related objects
4) Change configuration settings of an installed Event Collector

0) Exit Commands Menu
```

オプション 2 を選択して、Business Process Choreographer Event Collector アプリケーションをインストールします。 JNDI 名プロンプトが表示されます。

```
Specify the JNDI name of the database where the WebSphere BPC Event
Collector should store the collected events.
Enter '?' to get a list.
Your selection : [jdbc/BPEDB]
```

8. データベースへの接続に使用する JNDI 名を入力します。 また、? を入力して、登録済みのすべてのデータ・ソースのリストを入手できます。
9. 収集されたイベントが保管されるデータベース表のためのスキーマの名前を入力します。データ・ソース定義で指定されたスキーマを使用するには、スペース文字を入力するか、フィールドを空にしておきます。
10. Common Event Infrastructure (CEI) バスで認証する JMS ユーザー ID を入力します。 CEI バスのセキュリティーが使用不可の場合、これを空にしておくことができます。ユーザー ID を指定する場合は、次のプロンプトでパスワードも入力します。 必要なすべてのオブジェクトが作成され、エンタープライズ・アプリケーションがインストールされます。成功したことは、次のメッセージによって示されます。

    ```
    WebSphere Business Process Choreographer Event Collector
    installed successfully!
    ```

11. エラー・メッセージがない場合は、y を入力して構成を保管します。エラー・メッセージがある場合は、n を入力して変更を破棄し、元の構成を保持します。 エラーがあった場合、プロファイルの logs ディレクトリーにある setupEventCollector.log という名前のログ・ファイルを検査します。例えば Windows では、プロファイル名が myServer で、プロファイルが *install_root*¥profiles に保管されている場合、ログ・ファイルは *install_root*¥profiles¥myServer¥logs に置かれます。
12. CEI ロギングがサーバー上で有効にされていない場合、以下が表示されます。

    ```
    Checking if CEI event logging is enabled ...

    Warning: The Business process container of server_name has CEI event
    logging disabled.
    To allow the Event Collector to work correctly, CEI event logging is required.
    Do you want to enable the CEI event logging on server_name? (y/n)
    ```

    CEI ロギングを有効にするには「**y**」を入力し、そうでない場合は「**n**」を入力します。
13. アプリケーションを開始するプロンプトが出された場合、y を入力してアプリケーションを開始するか、n を入力して開始しないようにします。
14. すべての設定を有効にするには、サーバーを再始動します。

Business Process Choreographer Event Collector がインストールおよび構成されています。

ステップ 2 (102 ページ) で構成を続行します。 必要に応じて、コマンド・メニューのオプション 4 を使用して、Event Collector の構成パラメーターを変更できます。この説明は、サポート文書 7008553 にあります。

Business Process Choreographer Observer の構成

ここでは、スクリプトを使用して Business Process Choreographer Observer を構成する方法について説明します。これにより、ビジネス・プロセスおよびヒューマン・タスクの実行状態を監視できます。

- 管理コンソールまたは bpeconfig スクリプトを使用して Business Process Choreographer を構成しました。
- Business Process Choreographer Event Collector は構成しましたが、 Business Process Choreographer Observer はまだ構成していません。
- 監視したいビジネス・アプリケーションが実行するコンテナーについて、Common Event Infrastructure (CEI) ロギングが有効になっています。
- サーバーが稼動していなければなりません。

1. 構成スクリプトがある Business Process Choreographer サブディレクトリーに変更します。

   以下のコマンドを入力します。

   ```
   cd install_root/ProcessChoreographer/config
   ```

2. スクリプトを開始して、イベント・コンシューマーをセットアップします。

   以下のコマンドを入力します。

   ```
   setupObserver.sh [-conntype SOAP | RMI | JMS | NONE]
        ( [-node nodeName] -server serverName ) | ( -cluster clusterName )
        [ -remove [-silent]]
   ```

   各部の意味は、次のとおりです。

   conntype SOAP | RMI | JMS | NONE
   wsadmin ツールが使用する接続モード。

   node *nodeName*
   ノードの名前。このパラメーターはオプションです。デフォルト値はローカル・ノードです。

   server *serverName*
   サーバーの名前。オプション `-conntype none` を指定しない場合、このパラメーターはオプションです。

   cluster *clusterName*
   オプション `-conntype none` を指定しない場合、このパラメーターはオプションです。

   remove
   このオプションを指定して Business Process Choreographer Event Collector を除去します。このオプションを指定しない場合、デフォルトで Business Process Choreographer Event Collector が構成されます。

   silent このオプションは、remove オプションと一緒にしか使用できません。これにより、スクリプトはプロンプトを出力しないようになります。このパラメーターはオプションです。

   注:

   スタンドアロン・サーバー環境の場合:

- -conntype NONE オプションを指定するのはアプリケーション・サーバーが稼働していない場合に限定します。
- サーバーが稼働していてグローバル・セキュリティーが使用可能な場合は、-username および -password オプションを指定します。
- デフォルト・プロファイルを構成していない場合は、-profileName オプションを追加します。

「コマンド・メニュー (Commands Menu)」が表示されます。

```
Commands Menu

Working on node 'nodeName', server 'serverName'.

1) Install the Observer application
2) Remove the Observer application and related objects
3) Change configuration settings of an installed Observer

0) Exit Menu
```

3. オプション 1 を選択して、Business Process Choreographer Observer をインストールします。 JNDI 名プロンプトが表示されます。

```
Specify the JNDI name of the database containing the event tables.
Enter '?' to get a list.
Your selection : [jdbc/BPEDB]
```

4. Business Process Choreographer Event Collector が使用するデータ・ソースの JNDI 名を入力します。 また、? を入力して、登録済みのすべてのデータ・ソースのリストを入手できます。
5. 収集されたイベントが保管されるデータベース表のためのスキーマの名前を入力します。このフィールドを空にしておくか、スペース文字を入力する場合、デフォルトのスキーマが使用されます。これは、ステップ 4 で指定したデータ・ソースのプロパティーで指定されたユーザー ID のスキーマです。 必要なすべてのオブジェクトが作成され、エンタープライズ・アプリケーションがインストールされます。成功したことは、次のメッセージによって示されます。

```
WebSphere BPC Observer installed successfully!
```

6. エラー・メッセージがない場合は、y を入力して構成を保管します。エラー・メッセージがある場合は、n を入力して変更を破棄し、元の構成を保持します。エラーがあった場合は、プロファイルの logs ディレクトリーにある setupObserver.log という名前のログ・ファイルを検査します。例えば、*WebSphere/V6R0M0/AppServer*/profiles/default/logs に置かれます。
7. プロンプトが出た場合は、y を入力してアプリケーションを開始するか、 n を入力して開始しないようにします。
8. すべての設定を有効にするには、サーバーを再始動します。

Business Process Choreographer Observer がインストールおよび構成されています。http://*host*:*port*/bpcobserver/ からアクセスできます。ここで、*host* はアプリケーション・サーバーが実行しているホストの名前で、*port* はアプリケーション・サーバーのポート番号 (デフォルトは 9080) です。

ステップ 8 (17 ページ) で構成を続行してください。必要に応じて、コマンド・メニューのオプション 3 を使用して、Business Process Choreographer Observer の構成パラメーターを変更できます。この説明は、サポート文書 7008553 にあります。

Business Process Choreographer の活動化

ビジネス・プロセス・コンテナーおよびヒューマン・タスク・コンテナーを構成したら、これらのコンテナーのインストール先サーバーを再始動する必要があります。

Business Process Choreographer を活動状態にするには、以下の手順を実行します。

1. アプリケーション・サーバーのクラスターにビジネス・プロセス・コンテナーおよびヒューマン・タスク・コンテナーをインストールした場合は、クラスターを再始動します。
2. 1 つのアプリケーション・サーバーにビジネス・プロセス・コンテナーおよびヒューマン・タスク・コンテナーをインストールした場合は、アプリケーション・サーバーを再始動します。
3. ビジネス・プロセス・コンテナーおよびヒューマン・タスク・コンテナー・アプリケーションが正常に始動されたことを確認するには、アプリケーション・サーバーの `SystemOut.log` ファイルにエラー・メッセージが存在しないことを確認します。クラスターでは、クラスター内のすべてのアプリケーション・サーバーのログをチェックします。

Business Process Choreographer が稼働しています。

これで、Business Process Choreographer が作動していることを確認できます。

Business Process Choreographer の作動確認

Business Process Choreographer インストール検査アプリケーションを実行します。

アプリケーション・サーバー、データベース・システム、およびメッセージング・サービスが稼働している必要があります。

1. 管理コンソールまたは wsadmin コマンドを使用して、アプリケーションを *install_root*`/installableApps/bpcivt.ear` にインストールします。 Business Process Choreographer データベースにアクセスできない場合、この段階でエラーが発生します。データベース・システムが稼働していない場合、データベース・クライアントが正しく構成されていない場合、または無効なユーザー ID やパスワードを入力するなどの、データ・ソースの定義中にエラーが発生した場合に、このような問題が起こる可能性があります。 インストール後に、エンタープライズ・アプリケーションは停止状態になり、その中に含まれているプロセス・テンプレートやタスク・テンプレートは開始済みの状態になっています。アプリケーションを始動するまでは、プロセスやタスク・インスタンスを作成できません。
2. アプリケーション BPCIVTApp を開始するには、アプリケーションを選択して「**開始 (Start)**」をクリックします。 この時点で、はじめて入力キューが読み取られます。キュー・マネージャーが稼働中でない場合、あるいは JMS プロバイダーまたは JMS リソースの定義中に誤りがあった場合は、この段階でエラーが発生します。
3. アプリケーションが作動することを確認します。 Web ブラウザーを使用して、次のページを開きます。

   `http://`*app_server_host*`:`*port_no*`/bpcivt`

ここで、*app_server_host* はアプリケーション・サーバーのホストのネットワーク名であり、 *port_no* は、ファイル bpcivt.ear をインストールするときに IVT Web モジュールをマップした仮想ホストが使用するポート番号です。ポート番号はシステム構成によって異なります。 成功したことを示すメッセージが表示されます。

4. **オプション:** BPCIVTApp アプリケーションを停止して除去します。

Business Process Choreographer が作動しています。

ビジネス・プロセス・コンテナーの開始時の振る舞いについて

このトピックでは、すべてのエンタープライズ・アプリケーションが開始されるまでは、ビジネス・プロセス・コンテナーが使用不可となる理由について説明します。

ビジネス・プロセス・コンテナーの開始または再始動時には、すべてのエンタープライズ・アプリケーションが開始されるまで、内部キュー内のメッセージは処理されません。この振る舞いを変更することはできません。再始動時にビジネス・プロセス・コンテナーが使用不可な時間は、すべてのエンタープライズ・アプリケーションが開始されるまでに必要な時間の長さによって異なります。ビジネス・プロセス・エンジンが、稼働中でない関連したエンタープライズ・アプリケーションを使用してプロセスをナビゲートすることがないようにするために、この振る舞いが必要です。

すべてのアプリケーションが開始される前に、内部キュー内のメッセージの処理を開始すると、ClassNotFound 例外が発生します。

LDAP ユーザー・レジストリーを使用するための Business Process Choreographer の構成

ここでは、既存の Business Process Choreographer 構成を変更して、Lightweight Directory Access Protocol (LDAP) ユーザー・レジストリーを使用する方法について説明します。

1. LDAP ユーザー・レジストリーを構成します。
   a. 「**セキュリティー (Security)**」 → 「**グローバル・セキュリティー (Global security)**」と進み、「**グローバル・セキュリティーを使用可能にする (Enable global security)**」が使用可能になっていないことを確認します。
   b. 「**ユーザー・レジストリー (User registries)**」の下で **LDAP** を選択します。
   c. セキュリティー目的のために WebSphere Process Server を実行する時に使用するユーザー名とパスワードを設定します。 「**サーバー・ユーザー ID (Server user ID)**」フィールドにユーザー名を入力し、それに対応するパスワードを 「**サーバー・ユーザー・パスワード (Server user password)**」フィールドに入力します。この ID は、LDAP 管理者ユーザー ID ではありません。このユーザー ID は、LDAP レジストリーになくてはなりません。
   d. ユーザー・レジストリーとして使用する特定の LDAP を「**タイプ**」リストから選択します。
   e. 「**ホスト (Host)**」フィールドに、LDAP サーバーのホスト名を入力します。

f. 「**ポート**」フィールドに、LDAP サーバーがそこで listen を実行するポートの番号を入力します。

g. 「**基本識別名 (Base Distinguished Name)**」を入力します。

この値はディレクトリー・サービスの基本識別名を指定します。この識別名はディレクトリー・サービスの LDAP 検索の開始点を示します。

許可の目的のために、このフィールドでは大/小文字が区別されます。この識別名を指定するということは、 (例えば、他のセルまたは Lotus Domino サーバーなどから) トークンを受信した場合、サーバーの基本識別名 (DN) が他のセルまたは Lotus Domino サーバーの基本 DN と完全に一致しなければならないことを意味します。許可に関して大/小文字を区別する必要がなければ、「**大/小文字を区別しない (Ignore case)**」フィールドを使用可能にします。このフィールドは、 Lotus Domino Directory 以外のすべての LDAP ディレクトリーについて必須指定です。 Lotus Domino Directory については任意で指定します。

h. 「**バインド識別名 (Bind Distinguished Name)**」を入力します。 アプリケーション・サーバーが LDAP サーバーにバインドするために使用するユーザー ID を入力します。例えば、「**サーバー・ユーザー ID (Server user ID)**」に入力した同じユーザー ID を使用できます。

i. 「**バインド・パスワード (Bind Password)**」を入力します。 「**バインド識別名 (Bind Distinguished Name)**」で指定したユーザー ID のパスワードを入力します。

j. 残りのパラメーターはデフォルト値のままにして、変更を確認します。「**OK**」をクリックします。

2. Lightweight Third Party Authentication (LTPA) メカニズムを構成します。 「**認証 (Authentication)**」の下で「**認証メカニズム (Authentication mechanisms)**」を開き、**LTPA** を選択します。「**パスワード**」フィールドおよび「**パスワードの確認 (Confirm password)**」フィールドで、任意のパスワードを入力し「**OK**」をクリックします。

3. グローバル・セキュリティー、Java 2 セキュリティー、LTPA および LDAP を使用可能にします。

a. 「**グローバル・セキュリティーを使用可能にする (Enable global security)**」を選択します。

b. 「**Java 2 セキュリティーを施行する (Enforce Java 2 security)**」を選択します。

c. **アクティブ認証メカニズム**では、**Lightweight Third Party Authentication (LTPA)**を選択します。

d. **アクティブ・アクティブ・ユーザー**では、**Lightweight Directory Access Protocol (LDAP) ユーザー・レジストリー** を選択します。

e. 「**OK**」をクリックして、変更を保管します。

4. WebSphere Process Server を再始動します。

5. ステップ 1c (110 ページ) の「**サーバー・ユーザー ID (Server user ID)**」で指定したユーザー ID を使用して、管理コンソールへログオンします。

6. ビジネス・プロセス・コンテナー・アプリケーションの JMSAPIUser ロールに新規ユーザー・マッピングを追加します。

a. 管理コンソールで、ビジネス・プロセス・コンテナー・アプリケーションを検索します。 「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」 → **BPEContainer_<*your_node*>_<*your_server*>**を順にクリックします。
b. 「**追加プロパティー**」の下で、「**RunAs ロールをユーザーにマップする (Map RunAs roles to users)**」をクリックします。
c. **ユーザー名**には、LDAP ユーザー・レジストリーで定義される正当なユーザー ID を入力します。
d. **パスワード**には、ユーザー ID のパスワードを入力します。
e. **JMSAPIUser** のテーブル行の前面にあるチェック・ボックスを選択します。
f. 「**適用**」をクリックします。これは、ユーザー ID とロールを関連付けます。ユーザー ID がテーブルに追加されます。
g. 「**OK**」をクリックして、変更を保管します。

7. ビジネス・プロセス・コンテナー・アプリケーションの システム管理者およびシステム・モニター用のセキュリティー・ロール・マッピングを追加します。
a. 管理コンソールで、ビジネス・プロセス・コンテナー・アプリケーションを検索します。 「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」 → **BPEContainer_<*your_node*>_<*your_server*>**を順にクリックします。
b. 「**追加プロパティー**」の下で、「**ユーザー/グループにセキュリティー・ロールをマップする (Map security roles to users/group)**」をクリックします。
c. **BPESystemAdministrator** および **BPESystemMonitor** のテーブル行の前面にあるチェック・ボックスを選択します。
d. 「**ユーザーを検索 (Lookup Users)**」をクリックします。
e. 「**検索文字列 (Search String)**」で、文字 * を入力し、「**検索**」をクリックします。
f. 「**使用可能 (Available)**」リストで、ビジネス・プロセス管理者およびビジネス・プロセス・モニターのロールを実行するユーザーやグループのエントリーを選択し、**>>** をクリックします。
g. 「**OK**」をクリックします。
h. **BPESystemAdministrator** の前面にあるチェック・ボックスを選択し、他のチェック・ボックスが選択されている場合は、それらをクリアします。
i. 「**グループを検索 (Lookup Groups)**」をクリックします。
j. 「**選択済み**」フィールドに表示されているグループを選択してそのグループを除去し、**<<**、「**OK**」を順にクリックします。
k. 「**OK**」をクリックして、変更を保管します。

8. ヒューマン・タスク・コンテナー・アプリケーションの EscalationUser ロールの新規ユーザー名マッピングを追加します。
a. 管理コンソールで、ヒューマン・タスク・コンテナー・アプリケーションを検索します。 「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」 → **TaskContainer_<*your_node*>_<*your_server*>**を順にクリックします。
b. 「**追加プロパティー**」の下で、「**RunAs ロールをユーザーにマップする (Map RunAs roles to users)**」をクリックします。

c. **ユーザー名**には、LDAP ユーザー・レジストリーで定義される正当なユーザー ID を入力します。

d. **パスワード**には、ユーザー ID のパスワードを入力します。

e. **EscalationUser** のテーブル行の前面にあるチェック・ボックスを選択します。

f. 「**適用**」をクリックします。これは、ユーザー ID とロールを関連付けます。ユーザー ID がテーブルに追加されます。

g. 「**OK**」をクリックして、変更を保管します。

9. ヒューマン・タスク・コンテナー・アプリケーションの システム管理者およびシステム・モニター用のセキュリティー・ロール・マッピングを追加します。

a. 管理コンソールで、ビジネス・プロセス・コンテナー・アプリケーションを検索します。 「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」 → **TaskContainer_<*your_node*>_<*your_server*>**を順にクリックします。

b. 「**追加プロパティー**」の下で、「**ユーザー/グループにセキュリティー・ロールをマップする (Map security roles to users/group)**」をクリックします。

c. **TaskSystemAdministrator** および **TaskSystemMonitor** のテーブル行の前面にあるチェック・ボックスを選択します。

d. 「**ユーザーを検索 (Lookup Users)**」をクリックします。

e. 「**検索文字列 (Search String)**」で、文字 * を入力し、「**検索**」をクリックします。

f. 「**使用可能 (Available)**」リストで、ステップ 1c (110 ページ) で指定したユーザー ID のエントリーを選択し、**>>** をクリックします。

g. 「**OK**」をクリックします。

h. **TaskSystemAdministrator** の前面にあるチェック・ボックスを選択し、他のチェック・ボックスが選択されている場合は、それらをクリアします。

i. 「**グループを検索 (Lookup Groups)**」をクリックします。

j. 「**選択済み**」フィールドに表示されているグループを選択してそのグループを除去し、**<<**、「**OK**」を順にクリックします。

k. 「**OK**」をクリックして、変更を保管します。

10. サンプル LDAP スタッフ・プラグイン構成を変更します。

a. 管理コンソールで、「**リソース**」 → 「**スタッフ・プラグイン・プロバイダー**」 → 「**LDAP スタッフ・プラグイン・プロバイダー (LDAP Staff Plugin Provider)**」をクリックします。

b. 「**追加プロパティー**」の下の「**スタッフ・プラグイン構成 (Staff Plugin Configuration)**」をクリックします。

c. 「**LDAP スタッフ・プラグイン構成サンプル (LDAP Staff Plugin Configuration sample)**」をクリックします。

d. 「**追加プロパティー**」の下の「**カスタム・プロパティー (Custom properties)**」をクリックします。

e. **BaseDN** プロパティーの値を、ステップ 1g (111 ページ) で **Base Distinguished Name (DN)** に入力した同じ値に設定します。

f. **ProviderURL** の値を LDAP サーバーの URL に設定します。 例えば `ldap://host:port`。ここで、*host* および *port* は、ステップ 1e (110 ページ) および 1f (111 ページ) で入力した値です。

g. 「**OK**」をクリックして、変更を保管します。

11. J2EE コネクター・アーキテクチャー (J2C) の認証データ入力を変更します。

a. 管理コンソールで、「**セキュリティー (Security)**」 → 「**グローバル・セキュリティー (Global Security)**」をクリックします。

b. 「**JAAS 構成 (JAAS Configuration)**」で、「**J2C 認証データ (J2C Authentication data)**」をクリックします。

c. 各ユーザー別名を変更し、ユーザー ID およびパスワードを、LDAP ユーザー・レジストリーで定義される有効なユーザーID の値に設定します。データベースの別名は変更しないでください。

d. 変更を保管します。

12. サーバーを再始動します。

- ND の場合: クラスター、すべてのノード・エージェント、およびデプロイメント・マネージャーを停止して再始動します。
- 単一サーバーの場合: サーバーを再始動します。

注: ステップ 1c (110 ページ) で指定したサーバー・ユーザー ID を使用して、サーバー、ノード・エージェント、およびデプロイメント・マネージャーを停止します。

13. Business Process Choreographer アプリケーションが稼働していることを確認します。

a. 管理コンソールで、「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」をクリックします。

b. アプリケーション `BPEContainer_node_server`、 および `TaskContainer_node_server` が実行していることを確認します。

これで、すべてのスタッフ照会が、選択された LDAP サーバーに対して実行されます。

Business Process Choreographer が構成されているスタンドアロン・ノードの統合

ここでは、ビジネス・プロセスまたはヒューマン・タスク (あるいはその両方) を含んでいるアプリケーションを実行する、スタンドアロン・プロファイル内のサーバーを新規のデプロイメント・マネージャー・セルに統合する方法を説明します。

デプロイメント・マネージャーが実行中であり、そのホスト名およびポート番号がわかっています。 Business Process Choreographer は、サーバー上のスタンドアロン・プロファイルで構成されます。スタンドアロン・プロファイル内の Business Process Choreographer データベースは、デプロイメント・マネージャー・セルからリモートにアクセスできなければなりません。このため、サーバーは組み込み Cloudscape を使用するサンプル Business Process Choreographer 構成を基にすることはできません。

スタンドアロン・サーバーで実行している 1 つ以上のアプリケーションがあり、これにはビジネス・プロセスまたはヒューマン・タスクが含まれています。このサーバーをネットワーク・デプロイメント環境に統合します。以下のいずれかの場合があります。

- アプリケーションをテスト環境から実稼働環境に移動させたい。
- ND でサーバーおよびアプリケーションを管理するようにしたい。

1. ノードに多数のアプリケーションが含まれる場合、管理コネクター用のタイムアウトを大きくします。
2. コマンド行から、-includeapps および -includebuses オプションを指定して addNode コマンドを実行します。 このコマンドおよび発生する可能性のあるエラーについての詳細は、WebSphere Application Server for z/OS インフォメーション・センターで、『addNodeコマンド』を参照してください。 例えば、デプロイメント・マネージャーのホスト名が dmgr_host で、ポート dmgr_port を使用する場合、次のコマンドを入力します。

   ```
   addNode dmgr_host dmgr_port -includeapps -includebuses
   ```

   例えば、デプロイメント・マネージャーのホスト名が any.hostname.com で、ポート 9043 を使用し、プロファイル名が ProcSvr07、ユーザー ID が admin、およびパスワードが secret の場合、次のコマンドを入力します。

   ```
   addNode any.hostname.com 9043 -profileName ProcSvr07 -username admin
           -password secret -includeapps -includebuses
   ```

   前提条件のいずれかを満たしていない場合、エラー・メッセージが表示されます。満たしている場合は、サーバーは停止され、新規のデプロイメント・マネージャー・セルに統合されます。
3. JDBC プロバイダーの WebSphere 変数をデプロイメント・マネージャー・ノードで設定します。 使用している JDBC ドライバーのパスの変数を設定する必要があります。以下のいずれかです。
   - DB2UNIVERSAL_JDBC_DRIVER_PATH (DB2 の場合)
   - UNIVERSAL_JDBC_DRIVER_PATH (DB2 の場合)
   - CONNECTJDBC_JDBC_DRIVER_PATH (SQL Server の場合)
   - ORACLE_JDBC_DRIVER_PATH (Oracle の場合)
4. 変更を有効にするために、サーバーを始動します。
5. サーバー上で実行しているビジネス・アプリケーションにアクセスできない場合、デプロイメント・マネージャー上の管理コンソールを使用して、アプリケーション・サーバー用の仮想ホストおよび別名定義が新規セルと一致することを確認します。

現在、アプリケーションは同じサーバー上で稼働していますが、そのサーバーはデプロイメント・マネージャーを使用して管理できるセルにあります。

必要に応じて、サーバーをクラスターにプロモートできます。

関連タスク

116 ページの『Business Process Choreographer がクラスターに対して構成されているサーバーのプロモート』

ここでは、ビジネス・プロセスまたはヒューマン・タスク (あるいはその両方) を含んでいるアプリケーションを実行するサーバーをクラスターにプロモートする方法を説明します。

Business Process Choreographer がクラスターに対して構成されているサーバーのプロモート

ここでは、ビジネス・プロセスまたはヒューマン・タスク (あるいはその両方) を含んでいるアプリケーションを実行するサーバーをクラスターにプロモートする方法を説明します。

ND セル内のスタンドアロン・サーバーで実行している 1 つ以上のアプリケーションがあり、これにはビジネス・プロセスまたはヒューマン・タスクが含まれています。このサーバーからクラスターを作成することができます。例えば、作業負荷の容量、可用性を増大させたり、独自のプロファイルを持つ複数のアプリケーションを統合するためです。

1. クラスターを作成します。 Business Process Choreographer が構成されている既存のサーバーを使用して、クラスターの作成を実行します。

   制約事項:

   - Business Process Choreographer が構成されている既存のサーバーがある場合は、それがクラスター内の最初のサーバーでなければなりません。
   - 提供されているテンプレートのいずれかを使用してクラスターを作成する場合は、defaultProcessServer テンプレートを使用します。
2. 変更を有効にします。 サーバーが稼働していない場合は、始動してください。サーバーが稼働している場合は、それを停止して始動してください。

現在、アプリケーションは同じサーバー上で実行していますが、そのサーバーはクラスターのメンバーであり、リソースにはクラスター・スコープがあります。

メンバーをクラスターに追加して、作業負荷を共有し、可用性を増大させることができます。

第 3 章 Business Process Choreographer 構成の除去

このタスクを使用して、ビジネス・プロセス・コンテナー、ヒューマン・タスク・コンテナー、Business Process Choreographer Explorer および関連するリソースを除去します。

1. すべてのスタンドアロン・サーバー、データベース、およびアプリケーション・サーバー (またはクラスターごとに少なくとも 1 つのアプリケーション・サーバー) が稼働していることを確認します。
2. ヒューマン・タスクまたはビジネス・プロセスを収容しているエンタープライズ・アプリケーションごとに、すべてのヒューマン・タスク・テンプレートとすべてのビジネス・プロセス・テンプレートを停止およびアンインストールし、その後アプリケーションをアンインストールします。
3. 以下のいずれかのアクションを実行します。
   - ビジネス・プロセス・コンテナー、ヒューマン・タスク・コンテナー、Business Process Choreographer Explorer、および関連するリソースをアンインストールするには、『スクリプトを使用した Business Process Choreographer 構成の削除』を実行します。
   - 既存の構成の一部を再利用する場合は、120 ページの『管理コンソールを使用した Business Process Choreographer 構成の除去』を実行します。

Business Process Choreographer 構成が除去されました。

スクリプトを使用した Business Process Choreographer 構成の削除

このタスクを使用して、ビジネス・プロセス・コンテナー、タスク・コンテナー、Business Process Choreographer Explorer および Business Process Choreographer Observer 構成、および関連するリソースを削除します。

Business Process Choreographer 構成を削除するには、まずすべてのプロセス・テンプレートおよびタスク・テンプレートを停止してすべてのプロセス・インスタンスおよびタスク・インスタンスを削除してから、ビジネス・プロセスまたはヒューマン・タスクを含むすべてのエンタープライズ・アプリケーションの構成を停止して削除します。

1. Business Process Choreographer `sample` ディレクトリーに移動します。 次のように入力します。

   ```
   cd install_root/ProcessChoreographer/config
   ```

2. スクリプト `bpeunconfig.jacl` を実行します。 次のケースでは、該当するオプションも指定します。
   - スタンドアロン・サーバーの場合は、アプリケーション・サーバーを停止して `-conntype NONE` オプションを使用します。このステップでは、すべての Cloudscape データベースがロックされておらず、自動的に削除可能であることを確認します。
   - Network Deployment (ND) 環境では、次のようにスクリプトを実行します。

– デプロイメント・マネージャーを実行していない場合は、-conntype NONE オプションを使用してデプロイメント・マネージャー上でスクリプトを実行します。
– デプロイメント・マネージャーを実行している場合は、構成を削除するアプリケーション・サーバーを停止してから、-conntype NONE オプションを省略してスクリプトを実行します。

Business Process Choreographer 構成を削除するアプリケーション・サーバーのノード上でスクリプトを実行中の場合は、そのスクリプトによっていずれの Cloudscape データベースも自動的に削除できます。

- WebSphere セキュリティーが有効になっている場合は、ユーザー ID およびパスワードも指定します。

  -userid *userID* -password *password*
- デフォルト・プロファイルを構成していない場合は、プロファイル名も指定します。

  -profileName *profileName*

WebSphere セキュリティーが使用可能になっている単一サーバーの構成を除去する場合は、次のコマンドを入力します。

```
install_root/bin/wsadmin.sh -f bpeunconfig.jacl -server Server -node Node
            -userid userID -password password
```

WebSphere セキュリティーが使用不可になっている単一サーバーの構成を除去する場合は、次のコマンドを入力します。

```
install_root/bin/wsadmin.sh -f bpeunconfig.jacl -server Server -node Node
```

WebSphere セキュリティーが使用可能になっていクラスターの構成を除去する場合は、次のコマンドを入力します。

```
install_root/bin/wsadmin.sh -f bpeunconfig.jacl -cluster Cluster
            -userid userID -password password
```

WebSphere セキュリティーが無効な状態でクラスターの構成を削除する場合は、次のコマンドを入力します。

```
install_root/bin/wsadmin.sh -f bpeunconfig.jacl -cluster Cluster
```

各部の意味は、次のとおりです。

Server アプリケーション・サーバーの名前。サーバーが 1 つしか存在しない場合、このパラメーターはオプションです。

Node ノードの名前。これはオプションです。ノードを省略した場合は、ローカル・ノードが使用されます。

Cluster
クラスターの名前。

パラメーターを省略した場合は、入力を求めるプロンプトが出されます。

3. **オプション:** Business Process Choreographer が使用するデータベースを削除します。

   Business Process Choreographer データベースおよびメッセージング・データベースの場合は、以下が適用されます。

- bpeunconfig.jacl スクリプトには、削除された構成によって使用されていたデータベースがリストされます。従って、削除するデータベースをより簡単に識別することができます。
- Cloudscape データベースが Business Process Choreographer データベース用に使用されている場合は、実行中のアプリケーション・サーバーによってデータベースがロックされていない限り、オプションとして bpeunconfig.jacl スクリプトによりそのデータベースを削除することもできます。データベースがロックされている場合は、サーバーを停止してから -conntype NONE オプションを使用します。

4. **オプション:** ログ・ファイル *install_root*/profiles/*profileName*/logs/bpeunconfig.log を確認します。
5. 必須: WebSphere デフォルト・メッセージングが使用するデータベースを削除します。このデータベースを新規の構成で再使用することはできません。

   Business Process Choreographer データベースおよびメッセージング・データベースの場合は、以下が適用されます。

   - bpeunconfig.jacl スクリプトは、削除された構成が使用していたデータベースのリストを表示します。データベースのリストは *install_root*/profiles/*profileName*/logs/bpeunconfig.log ログ・ファイルにも書き込まれます。 この情報を使用して、削除の対象となるデータベースを確認します。
   - Cloudscape データベースがメッセージング・データベースの場合は、実行中のアプリケーション・サーバーによってデータベースがロックされていない限り、オプションとして bpeunconfig.jacl スクリプトによりそのデータベースを削除することもできます。データベースがロックされている場合は、サーバーを停止してから -conntype NONE オプションを使用します。

6. **オプション:** WebSphere MQ の場合のみ、Business Process Choreographer が使用するキュー・マネージャーを削除します。
7. **オプション:** bpeunconfig.jacl では元に戻らない残りの設定を手動で元に戻します。 以下の設定は bpeunconfig.jacl スクリプトでは元に戻りません。これは、設定が他のコンポーネントによって引き続き必要とされているかどうかをこのスクリプトが判別できないためです。
   - WorkAreaService の使用可能化
   - ApplicationProfileService の使用可能化
   - ObjectPoolService の使用可能化
   - StartupBeansService の使用可能化
   - CompensationService の使用可能化
   - WorkareaPartitionService の使用可能化
   - WebSphere セキュリティーおよび Java 2 セキュリティーの使用可能化
   - WebSphere 変数の設定
   - SchedulerCalendars アプリケーションのターゲット・マッピングのインストールまたは追加
8. Business Process Choreographer Observer を構成した場合、次の説明に従って両方のセットアップ・スクリプトを実行し、remove オプションを選択することによって、それをイベント・コレクターと共に削除します。

a. 103 ページの『Business Process Choreographer Event Collector の構成』
b. 107 ページの『Business Process Choreographer Observer の構成』

Business Process Choreographer アプリケーションおよび関連するリソース (スケジューラー、データ・ソース、リスナー・ポート、接続ファクトリー、キュー宛先、活動化仕様、作業域区画、メール・セッション、認証別名など) が除去されます。

管理コンソールを使用した Business Process Choreographer 構成の除去

このタスクを使用して、ビジネス・プロセス・コンテナー、タスク・コンテナー、Business Process Choreographer Explorer 構成および関連するリソースの一部またはすべてを除去します。

Business Process Choreographer 構成を削除する前に、すべてのプロセス・テンプレートおよびタスク・テンプレートを停止し、すべてのプロセス・インスタンスおよびタスク・インスタンスを削除してから、ビジネス・プロセスまたはヒューマン・タスクを含むすべてのエンタープライズ・アプリケーションを停止してアンインストールする必要があります。

1. Business Process Choreographer エンタープライズ・アプリケーションをアンインストールします。
   a. エンタープライズ・アプリケーションを表示します。

      管理コンソールで、「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」を選択します。

   b. Business Process Choreographer インストールの有効範囲を確認します。

      次の名前を持ったアプリケーションを探します。

      - `BPEContainer_`*`scope`* は、ビジネス・プロセス・コンテナー・アプリケーションです。
      - `TaskContainer_`*`scope`* は、ヒューマン・タスク・コンテナー・アプリケーションです。
      - `BPCExplorer_`*`scope`* は、Business Process Choreographer Explorer アプリケーションです。
      - `BPCObserver_`*`scope`* は、Business Process Choreographer Observer アプリケーションです。
      - `BPCECollector_`*`scope`* は、Business Process Choreographer Observer が必要とする Event Collector アプリケーションです。

      ここで、*scope* の値は構成によって異なります。

      - Business Process Choreographer がアプリケーション・サーバー上で構成された場合、 *scope* は *`nodeName_serverName`* の値を持ちます。
      - Business Process Choreographer がクラスター上で構成された場合、*scope* は *`clusterName`* の値を持ちます。

   c. **オプション:** ビジネス・プロセス・コンテナーをインストールした場合は、アンインストールします。 `BPEContainer_`*`Scope`* を選択し、「**アンインストール**」 → 「**OK**」 → 「**保管**」 → 「**保管**」をクリックします。

d. **オプション:** ヒューマン・タスク・コンテナーをインストールした場合は、アンインストールします。

1) TaskContainer_*Scope* を選択して「**停止**」をクリックします。

2) もう一度アプリケーションを選択して、「**アンインストール**」 → 「**OK**」 → 「**保管**」 → 「**保管**」をクリックします。

e. **オプション:** Business Process Choreographer Explorer をインストールした場合は、アンインストールします。

1) BPCExplorer_*Scope* を選択して、「**停止**」をクリックします。

2) もう一度アプリケーションを選択して、「**アンインストール**」 → 「**OK**」 → 「**保管**」 → 「**保管**」をクリックします。

f. **オプション:** Business Process Choreographer Observer をインストールした場合は、それと Business Process Choreographer Common Event Infrastructure コンシューマーをアンインストールします。

1) bpcobserver_*Scope* と bpcobservereventconsumer_*Scope* を選択して、「**停止**」をクリックします。

2) もう一度アプリケーションを選択して、「**アンインストール**」 → 「**OK**」 → 「**保管**」 → 「**保管**」をクリックします。

2. 再利用しない次のリソースのすべてまたは一部を削除します。

a. **オプション:** Business Process Choreographer データ・ソース (デフォルト名は BPEDataSource*dbType*) を検索し、削除する前に、それに関連する認証データ別名 (ある場合) と Java Naming and Directory Interface (JNDI) 名 (単一サーバーの場合のデフォルト名は jdbc/BPEDB) をメモしておきます。

データ・ソースを検索するには:

1) 「**リソース**」 → 「**JDBC プロバイダー**」をクリックします。

2) Business Process Choreographer がアプリケーション・サーバー上にインストールされていた場合は、「**サーバー**」を選択します。

3) Business Process Choreographer がクラスター上にインストールされていた場合は、クラスターを選択します。

4) 「**適用**」をクリックします。

5) 該当の JDBC プロバイダーを選択して、「**データ・ソース**」をクリックします。

6) Oracle データベース管理システムを使用している場合は、2 次データ・ソース BPEDataSourceOracleNonXA も除去します。

b. **オプション:** Cloudscape データベース以外のデータベースの場合は、必要なデータ・ソースがデータベースに含まれていない場合に限り、ステップ 2 で識別されたデータ・ソースの JDBC プロバイダーを除去します。

c. **オプション:** 該当する接続ファクトリーとキューを除去します。

- デフォルトのメッセージングの場合は、接続ファクトリーを除去する前に、関連する認証データの別名をメモしておきます。その後、JMS 接続ファクトリーと JMS キューを除去します。

1) 「**リソース**」 → 「**JMS プロバイダー**」 → 「**デフォルト・メッセージング**」をクリックします。

2) 「デフォルトのメッセージング・プロバイダー (Default messaging provider)」ペインで、以下のいずれかを実行します。
   - Business Process Choreographer をクラスター上で構成した場合は、「**クラスター**」を選択してから「**適用**」をクリックします。
   - Business Process Choreographer をサーバー上で構成した場合は、「**サーバー**」を選択してから「**適用**」をクリックします。

- WebSphere MQ の場合は、JMS キューの接続ファクトリーおよび JMS キュー宛先を削除します。
  1) 「**リソース**」 → 「**JMS プロバイダー**」 → 「**WebSphere MQ**」をクリックします。
  2) 「WebSphere MQ メッセージング・プロバイダー (WebSphere MQ messaging provider)」ペインで、「**サーバー**」を選択します。次に「**適用**」をクリックします。

  Business Process Choreographer をクラスター上で構成した場合は、この手順をクラスターのメンバーであるサーバーごとに繰り返す必要があります。

ビジネス・プロセス・コンテナーの場合は、通常、JNDI 名は次のようになります。

```
jms/BPECF
jms/BPECFC
jms/BPEIntQueue
jms/BPERetQueue
jms/BPEHldQueue
```

ヒューマン・タスク・コンテナーの場合は、通常、JNDI 名は次のようになります。

```
jms/HTMCF
jms/HTMIntQueue
jms/HTMHldQueue
```

d. **オプション:** WebSphere デフォルト・メッセージングを JMS プロバイダーとして使用している場合は、活動化仕様を削除します。
   1) 「**リソース**」 → 「**JMS プロバイダー**」 → 「**デフォルト・メッセージング**」 → 「**JMS 活動化仕様**」をクリックします。
   2) 以下の活動化仕様を削除します。

```
BPEInternalActivationSpec
HTMInternalActivationSpec
```

e. **オプション:** WebSphere MQ を JMS プロバイダーとして使用している場合は、リスナー・ポートを削除します。
   1) 「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***serverName***」をクリックします。
   2) 「通信」の下で「**メッセージング**」 → 「**メッセージ・リスナー・サービス**」 → 「**リスナー・ポート**」をクリックします。
   3) 「アプリケーション・サーバー」ペインで、次のリスナー・ポートを削除します。

```
BPEInternalListenerPort
```

BPEApiListenerPort
BPEHoldListenerPort
HTMInternalListenerPort

f. **オプション:** 認証データ別名を削除します。

- ステップ 2 (121 ページ) で確認したデータ・ソースが認証データ別名を持つ場合は、その別名を除去します。

  通常、データベースの別名は *cellName*/BPEAuthDataAlias*dbType_Scope* となります。

  cellName
  セルの名前

  dbType
  データベース・タイプ

  Scope
  ステップ 1b (120 ページ) で指定する値のいずれか

- ステップ 2c (121 ページ) で指定した接続ファクトリーのいずれかに認証データ別名がある場合は、この別名を削除します。

  通常、データベースの別名は *cellName*/BPEAuthDataAliasJMS_*Scope* となります。

  cellName
  セルの名前

  Scope
  ステップ 1b (120 ページ) で指定する値のいずれか

認証データ別名は、「**セキュリティー**」 → 「**グローバル・セキュリティー**」 → 「**JAAS 構成**」 → 「**J2C 認証データ**」にあります。

g. **オプション:** データ・ソースの JNDI 名のスケジューラー構成を除去します。

1) 「**リソース**」 → 「**スケジューラー**」をクリックします。
2) Business Process Choreographer 構成の有効範囲を選択します。「**サーバー**」または「**クラスター**」のいずれかです。次に「**適用**」をクリックします。
3) 「スケジューラー」ペインで、作業マネージャー名をメモしてからスケジューラー「BPEScheduler」を選択して削除します。

h. **オプション:** 作業マネージャーを除去します。

1) 「**リソース**」 → 「**非同期 Bean**」 → 「**作業マネージャー (Work managers)**」をクリックします。
2) Business Process Choreographer 構成の有効範囲を選択します。「**サーバー**」または「**クラスター**」のいずれかです。次に「**適用**」をクリックします。
3) 「作業マネージャー (Work managers)」ペインで、ステップ 2g でメモした名前の作業マネージャーを選択して削除します。

i. **オプション:** 作業域区画を除去します。

1) 「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***serverName***」をクリックします。

2) Business Process Services の下の「**作業域区画サービス**」をクリックします。

3) 「アプリケーション・サーバー」ペインで、作業域区画「BPECompensation」を選択して削除します。

j. **オプション:** メール・セッションを削除します。

1) **「リソース」 → 「メール・プロバイダー」**をクリックします。

2) 「メール・プロバイダー」ペインで**「セル」**を選択します。次に**「適用」**をクリックします。

3) **「組み込みメール・プロバイダー」**をクリックします。

4) 「追加プロパティー」の下の**「メール・セッション」**を選択します。

5) 「HTMMailSession_*Scope*」を選択して削除します。*Scope* はステップ 1b (120 ページ) で確認した有効範囲です。

k. クラスターで、クラスター・メンバーごとにサーバー・レベル・リソースの除去を繰り返します。

l. 構成の変更を保管します。

m. アプリケーション・サーバーを再始動します。

3. **オプション:** Business Process Choreographer で WebSphere デフォルト・メッセージングを使用している場合は、バス・メンバー、バス、データ・ソースを削除できます。

a. **「サービス統合」 → 「バス」 → 「BPC.*cellName*.Bus」 → 「メッセージング・エンジン」**をクリックします。

b. メッセージング・エンジンを選択します。

- Business Process Choreographer をサーバー上で構成した場合は、***nodeName*.*serverName*-BPC.*cellName*.Bus**。
- Business Process Choreographer をクラスター内で構成した場合は、***clusterName*-BPC.*cellName*.Bus**。

注: リモート・メッセージング・エンジンを使用するように Business Process Choreographer を構成した場合、*clusterName* は Business Process Choreographer を構成したクラスターの名前と一致しなくなることがあります。

c. 「*追加プロパティー*」で、「データ・ストア」を選択し、データ・ソースの JNDI 名をメモしておきます。

d. **「サービス統合」 → 「バス」 → 「BPC.*cellName*.Bus」 → 「バス・メンバー」**に移動して、次のいずれかで識別できるバス・メンバーを削除します。

- Business Process Choreographer をサーバー上で構成した場合は、Node=*nodeName*, Server=*serverName*。
- Business Process Choreographer をクラスター上で構成した場合は、Cluster=*clusterName*。

e. **オプション:** バス BPC.*cellName*.Bus の最後のメンバーを除去したら、バスも除去できます。

f. **オプション:** データ・ソースを除去します。 「**リソース**」 → 「**JDBC プロバイダー**」 → 「**サーバー**」 → 「**適用**」 → 「**Cloudscape JDBC プロバイダー**」 → 「**データ・ソース**」をクリックして、ステップ 3c (124 ページ) でメモしたデータ・ソースを削除します。

4. **オプション:** Business Process Choreographer をクラスター上で構成した場合は、次のようにして BPC_REMOTE_DESTINATION_LOCATION 変数を削除します。「**環境**」 → 「**WebSphere 変数**」 → 「**クラスター**」 → 「**適用**」をクリックします。 BPC_REMOTE_DESTINATION_LOCATION という名前の変数を選択して、「**削除**」をクリックします。
5. 「**保管**」をクリックして、削除したすべての内容をマスター構成に保管します。
6. **オプション:** Business Process Choreographer データベースを削除します。
7. **オプション:** WebSphere MQ を使用している場合は、Business Process Choreographer が使用するキュー・マネージャーを削除します。
8. Business Process Choreographer で WebSphere デフォルト・メッセージングを使用している場合は、メッセージ・エンジンのデータ・ストアを削除します。デフォルトのデータ・ストアを使用している場合は、次のディレクトリーを削除することにより、そのデータ・ストアを削除できます。
   - Windows システムでは、以下を削除します。

     install_root¥profiles¥*profileName*¥databases¥com.ibm.ws.sib¥
     nodeName.serverName-BPC.*cellName*.Bus
   - UNIX システムおよび Linux システムでは、

     install_root/profiles/*profileName*/databases/com.ibm.ws.sib/ を削除します。
     nodeName.serverName-BPC.*cellName*.Bus
9. **オプション:** Business Process Choreographer Observer を構成した場合、以下のようにします。
   a. 各 Event Collector の『管理コンソールを使用した Business Process Choreographer Event Collector の除去』。
   b. 127 ページの『管理コンソールを使用した Business Process Choreographer Observer の除去』.

Business Process Choreographer 構成が除去されました。

管理コンソールを使用した Business Process Choreographer Event Collector の除去

このタスクを使用して、Business Process Choreographer Event Collector 構成、および Business Process Choreographer Observer によって必要とされる関連するリソースを除去します。

1. エンタープライズ・アプリケーションを表示します。

   管理コンソールで、「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」を選択します。

2. Business Process Choreographer Event Collector アプリケーションをアンインストールします。 BPCECollector_*node_name_server_name* の前のチェック・ボックスを選択し、「**アンインストール**」 → 「**OK**」 → 「**保管**」をクリックします。
3. BPCTransformerQueueDestination 宛先を削除します。
   a. 「**サービス統合 (Service integration)**」 → 「**バス (Buses)**」 → **CommonEventInfrastructure_Bus** をクリックします。
   b. 「**宛先リソース (Destination Resources)**」の下の「**宛先**」をクリックします。
   c. BPCTransformerQueueDestination_*node_name_server_name* のチェック・ボックスを選択します。
   d. 「**削除**」をクリックします。
4. BPCCEIConsumerQueueConnectionFactory JMS キュー接続ファクトリーを削除します。
   a. 「**リソース**」 → 「**JMS プロバイダー**」 → 「**デフォルトのメッセージング (サーバー有効範囲) (Default messaging (server scope))**」 → 「**JMS キュー接続ファクトリー (JMS queue connection factory)**」をクリックします。
   b. BPCCEIConsumerQueueConnectionFactory のチェック・ボックスを選択します。
   c. 「**削除**」をクリックします。
5. JMS キューを削除します。
   a. 「**リソース**」 → 「**JMS プロバイダー**」 → 「**デフォルト・メッセージング**」 → 「**JMS キュー**」をクリックします。
   b. BPCCEIConsumerQueue_*serverName* と BPCTransformerQueue_*nodeName_serverName* のチェック・ボックスを選択します。
   c. 「**削除**」をクリックします。
6. JMS 活動化仕様を削除します。
   a. 「**リソース**」 → 「**JMS プロバイダー**」 → 「**デフォルト・メッセージング**」 → 「**JMS 活動化仕様**」をクリックします。
   b. BPCCEIConsumerActivationSpec と BPCTransformerActivationSpec のチェック・ボックスを選択します。
   c. 「**削除**」をクリックします。
7. サーバー有効範囲 BFMEvents のイベント・プロファイル・グループを削除します。
   a. 「**リソース**」 → 「**Common Event Infrastructure プロバイダー**」 → 「**イベント・グループ・プロファイル・リスト**」をクリックします。
   b. 「**イベント・グループ・リスト (Event groups list)**」リンクをクリックします。
   c. 「**イベント・グループ・プロファイル**」を選択します。
   d. BFMEvents のチェック・ボックスを選択します。
   e. 「**削除**」をクリックします。

8. 認証データ別名
   BPCEventCollectorJMSAuthenticationAlias_*nodeName_serverName*を削除します。 「**セキュリティー**」 → 「**グローバル・セキュリティー**」 → 「**JAAS 構成**」 → 「**J2C 認証データ**」で認証データ別名を見つけて、それを削除します。
9. 「**保管**」をクリックして、変更した内容をマスター構成に保管します。
10. Business Process Choreographer Observer によって使用されるデータベース、スキーマ、およびテーブル・スペースを除去します。これは、Windows プラットフォームでは *install_root*¥dbscripts¥ProcessChoreographer¥*database_type* ディレクトリー、Linux および UNIX プラットフォームでは *install_root*¥dbscripts¥ProcessChoreographer¥*database_type* ディレクトリーにある以下のスクリプトを実行することによって行います。
    - dropSchema_Observer.sql
    - dropTablespace_Observer.sql
    - dropDataBase_Observer.sql

Business Process Choreographer Event Collector 構成が除去されました。

管理コンソールを使用した Business Process Choreographer Observer の除去

このタスクを使用して、Business Process Choreographer Observer 構成および関連するリソースを除去します。

1. エンタープライズ・アプリケーションを表示します。

   管理コンソールで、「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」を選択します。
2. Business Process Choreographer Observer インストールの有効範囲を確認します。

   BPCObserver_*scope* という名前のアプリケーションを探します。
   - Business Process Choreographer Observer がアプリケーション・サーバーにインストールされている場合は、*scope* の値は *nodeName_serverName* になります。
   - Business Process Choreographer Observer がクラスターにインストールされている場合は、*scope* の値は *clusterName* になります。
3. Business Process Choreographer Observer アプリケーションをアンインストールします。
   a. BPCObserver_*scope* を選択して、「**停止**」をクリックします。
   b. もう一度アプリケーションを選択して、「**アンインストール**」 → 「**OK**」 → 「**保管**」 → 「**保管**」をクリックします。
4. 「**保管**」をクリックして、変更した内容をマスター構成に保管します。

Business Process Choreographer Observer 構成が除去されました。

第 4 章 管理

Business Process Choreographer Explorer の使用

ビジネス・プロセスまたはヒューマン・タスクの管理者であれば、Business Process Choreographer Explorer を使用して、ビジネス・プロセス・オブジェクトやヒューマン・タスク・オブジェクト (プロセス・インスタンス、失敗したアクティビティー、作業割り当てなど) を管理できます。ビジネス・ユーザーなら、Business Process Choreographer Explorer を使用して、割り当てられたタスクを処理することができます。

Business Process Choreographer Explorer ユーザー・インターフェース

Business Process Choreographer Explorer とは、ビジネス・プロセスおよびヒューマン・タスクを管理するための管理機能のセットを提供するスタンドアロン Web アプリケーションです。このインターフェースは、タスクバー、ナビゲーション・ペイン、およびワークスペースで構成されています。

以下の図は、Business Process Choreographer Explorer ユーザー・インターフェースのレイアウトを示しています。

ユーザー・インターフェースには以下のような主要な領域があります。

タスクバー

すべてのユーザーは、タスクバーを使用して、 Business Process Choreographer Explorer からログアウトしたり、オンライン・ヘルプにアクセスすることができます。システム管理者権限を持っている場合、タスクバーには以下のオプションも含まれます。

カスタマイズ
このオプションは、Business Process Choreographer Explorer のこのインスタンスのナビゲーション・ペインに、ビューを追加したり除去したりするために選択します。また、ユーザーがログインしたときに表示されるビューを定義することもできます。

ビューの定義
このオプションは、ユーザー・グループのカスタマイズ・ビューを定義するために選択します。

ナビゲーション・ペイン

ユーザー・インターフェースの左側のナビゲーション・ペインには、オブジェクト (例えば、開始済みのプロセス・インスタンス、または管理する許可が与えられているヒューマン・タスク) の管理に使用するビューへのリンクが含まれています。デフォルト・ユーザー・インターフェースには、ビジネス・プロセスおよびタスク用の事前定義ビューへのリンクが含まれています。

システム管理者は、ナビゲーション・ペインで事前定義ビューの追加や除去を行い、ナビゲーション・ペインに追加するためのカスタム・ビューを定義することにより、デフォルトのナビゲーション・ペインの内容をカスタマイズできます。すべてのユーザーは、ナビゲーション・ペインから個別設定ビューを定義できます。

ワークスペース

ユーザー・インターフェースの右側のワークスペースには、ビジネス・プロセスおよびヒューマン・タスク関連オブジェクトを表示および管理するために使用するページが含まれています。ナビゲーション・ペインのリンクをクリックするか、アクション・バーのアクションをクリックするか、またはワークスペース・ページ自体の内部のリンクをクリックすることによって、これらのページにアクセスします。

Business Process Choreographer Explorer ナビゲーション・ペイン

ナビゲーション・ペインは、ビジネス・プロセスおよびヒューマン・タスク・オブジェクト (プロセス・インスタンスおよび作業割り当てなど) を管理するために使用するビューへのアクセスに使用します。デフォルト・ユーザー・インターフェースには、ビジネス・プロセスおよびタスク用の事前定義ビューへのリンクが含まれています。また、独自のビューを定義して、ナビゲーション・ペインに追加することもできます。

使用可能なアクション

ナビゲーション・ペインでは、次のアクションが使用可能です。

- ビューへのナビゲート。

ビュー名をクリックして、そのビューにナビゲートします。

- グループの縮小および展開。

  ナビゲーション・ペインの項目の横にある正符号 (+) をクリックして拡張するか、負符号 (-) をクリックしてその項目を縮小表示します。その項目自体をクリックして、拡張状態および縮小状態を切り替えることもできます。

- 検索。

  「**検索**」アイコン () をクリックして、オブジェクトを検索するか、個別設定ビューを定義します。

ビュー型によっては、ポップアップ・メニューから追加アクションを使用できます。アイコンによって、ポップアップ・メニューが使用可能であるかどうか示されます。

- ビューを削除するには、「**削除**」アイコン () をクリックします。
- ビューを変更するには、「**編集**」アイコン () をクリックします。
- ビューのコピーを作成して、そのコピーを変更するには、「**コピー**」アイコン () をクリックします。
- ビューをリスト内で上下に移動させるには、「**上へ**」アイコン () または「**下へ**」アイコン () をクリックします。

デフォルトのナビゲーション・ペインの事前定義ビュー

デフォルトのナビゲーション・ペインには、以下のビューのグループが含まれています。ご使用の Business Process Choreographer Explorer のナビゲーション・ペインに表示されるビューは、システム管理者がナビゲーション・ペインにビューを追加したり、あるいはナビゲーション・ペインからビューを除去したりすることがあるので、それに応じて異なる場合があります。

プロセス・テンプレート

プロセス・テンプレート・グループには、以下のビューが含まれます。

ユーザーのプロセス・テンプレート

このビューには、プロセス・テンプレートのリストが表示されます。このビューから、プロセス・テンプレートとその構造に関する情報を表示したり、テンプレートに関連したプロセス・インスタンスのリストを表示したり、プロセス・インスタンスを開始したりすることができます。

プロセス・インスタンス

プロセス・インスタンス・グループには、以下のビューが含まれます。

ユーザーが開始

このビューには、開始したプロセス・インスタンスが表示されま

す。このビューから、プロセス・インスタンスの進行をモニターし、それに関連したアクティビティー、プロセス、またはタスクをリストできます。

ユーザーが管理
このビューには、管理する許可が与えられているプロセス・インスタンスが表示されます。このビューから、プロセス・インスタンスを操作することができます。例えば、プロセスの中断と再開、プロセス・インスタンス内のアクティビティーの進行のモニターを行えます。

重大なプロセス
このビューには、停止状態のアクティビティーを含む、実行状態のプロセス・インスタンスが表示されます。このビューから、プロセス・インスタンスを操作したり、アクティビティーをリストしてからそれらを操作することができます。

終了したプロセス
このビューには、終了状態のプロセス・インスタンスが表示されます。このビューから、それらのプロセス・インスタンスを操作できます。

失敗した補正
このビューには、microflow についての失敗した補正アクションが表示されます。

アクティビティー・インスタンス
アクティビティー・インスタンス・グループの場合、デフォルト・ナビゲーション・ペインにビューは含まれません。

タスク・テンプレート
タスク・テンプレート・グループには、以下のビューが含まれます。

ユーザーのタスク・テンプレート
このビューには、タスク・テンプレートのリストが表示されます。このビューから、タスク・インスタンスの作成と開始を行い、テンプレートに関連したタスク・インスタンスのリストを表示できます。

タスク・インスタンス
タスク・インスタンス・グループには、以下のビューが含まれます。

ユーザーのタスク
このビューには、処理する許可が与えられているタスク・インスタンスのリストが表示されます。このビューから、タスク・インスタンスを処理したり、要求したタスク・インスタンスを解放したり、タスク・インスタンスを別のユーザーに転送することができます。

すべてのタスク
このビューには、ユーザーが所有者、潜在的所有者、または編集者となっているタスクがすべて表示されます。このビューから、タスク・インスタンスを処理したり、要求したタスク・インスタンスを解放したり、タスク・インスタンスを別のユーザーに転送することができます。

ユーザーが開始
このビューには、開始したタスク・インスタンスが表示されます。このビューから、タスク・インスタンスを処理したり、要求したタスク・インスタンスを解放したり、タスク・インスタンスを別のユーザーに転送することができます。

ユーザーが管理
このビューには、管理する許可が与えられているタスク・インスタンスが表示されます。このビューから、タスク・インスタンスを処理することができます。例えば、プロセスの中断と再開、タスク・インスタンスの作業項目の作成、タスク・インスタンスの現行作業項目のリストの表示を行えます。

ユーザーのエスカレーション
このビューには、ログオンしたユーザーのすべてのエスカレーションが表示されます。

ビュー型

ナビゲーション・ペインには、以下のタイプのビューが含まれます。ビューによっては、ポップアップ・メニューから追加アクションを使用可能です。

- デフォルトのナビゲーション・ペインの事前定義ビュー。

  これらのビュー・グループは、ナビゲーション・ペインが、「ナビゲーション・ツリーおよびログイン・ビューのカスタマイズ」ページにおいてシステム管理者によって変更されていない場合にのみ使用可能です。これらのビューでは、ポップアップ・メニューは使用不可です。
- システム管理者によってナビゲーション・ペインに追加されたカスタマイズ・ビューと事前定義ビュー。ポップアップ・メニューは、システム管理者は使用可能ですが、ビジネス・ユーザーは使用不可です。
  - 事前定義ビューは「**事前定義ビュー (Predefined view)**」アイコン で示されます。システム管理者はポップアップ・メニューを使用して、ナビゲーション・ペインのビューの位置を変更できます。
  - カスタマイズ・ビューは「**カスタム・ビュー (Custom view)**」アイコン で示されます。システム管理者は、これらのビューを削除、編集、コピー、および移動することができます。
- 個別設定ビュー。

  これらのビューは「**カスタム・ビュー (Custom view)**」アイコン で示されます。このビューは、ビューを作成したユーザーが、削除、編集、コピー、および移動することができます。

Business Process Choreographer Explorer の開始

Business Process Choreographer Explorer は Web アプリケーションで、ビジネス・プロセス・コンテナーの構成の一部としてインストールされます。 Web ブラウザーから Business Process Choreographer Explorer の使用を開始するためには、その前に、ビジネス・プロセス・コンテナー、ヒューマン・タスク・コンテナー、および

Business Process Choreographer Explorer アプリケーションがインストールされていて、かつこのアプリケーションが動作している必要があります。

Business Process Choreographer Explorer を開始するには、次のステップを実行します。

1. Web ブラウザーで、Business Process Choreographer Explorer の URL にアクセスします。

   URL の形式は次のようになっています。URL の値は、ご使用のシステムで仮想ホストとコンテキスト・ルートがどのように構成されているかによって異なります。

   `http://app_server_host:port_no/context_root`

   各部の意味は、次のとおりです。

   app_server_host
   使用するビジネス・プロセス・アプリケーションを提供するアプリケーション・サーバーのホストのネットワーク名。

   port_no
   Business Process Choreographer Explorer が使用するポート番号。ポート番号はシステム構成によって異なります。デフォルト・ポート番号は `9080` です。

   context_root
   Business Process Choreographer Explorer アプリケーションの、アプリケーション・サーバー上のルート・ディレクトリー。デフォルトは `bpc` です。
2. セキュリティーが有効になっている場合は、ユーザー ID とパスワードを入力して、「**ログイン**」をクリックする必要があります。

Business Process Choreographer Explorer の最初のページが表示されます。デフォルトでは、このページには「ユーザーのタスク」ビューが表示されます。

Business Process Choreographer Explorer のカスタマイズ

Business Process Choreographer Explorer では、管理者がビジネス・プロセスやヒューマン・タスクを管理するため、およびビジネス・ユーザーが割り当てられたタスクを処理するためのユーザー・インターフェースが用意されています。これは汎用インターフェースなので、このインターフェースをユーザー・グループのビジネス上のニーズに合わせてカスタマイズすることもできます。

ユーザー・インターフェースのカスタマイズには、いくつかの方法があります。

さまざまなユーザー・グループに応じた Business Process Explorer インターフェースのカスタマイズ

デフォルトの Business Process Choreographer Explorer ユーザー・インターフェースのナビゲーション・ペインには、事前定義ビューへのリンクがいくつか用意されています。「ユーザーのタスク」ビューは、ログイン後に表示されるデフォルト・ビューです。Business Process Choreographer システム管理者ロールのいずれかを持っていると、ナビゲーション・ペインに表示されるリンク、ユーザーのログイン後に表示されるビュー、およびビューに表示される情報をカスタマイズできます。

例えば、Business Process Choreographer Explorer のデフォルトのユーザー・インターフェースには、ビジネス状態マシンを操作するためのビューが組み込まれていません。事前定義ビューを追加すると、ビジネス状態マシン用のプロセス・テンプレートとプロセス・インスタンスを操作できるようになります。

あるいは、顧客オーダーを担当するユーザーに、顧客サービス照会を担当するユーザーとは別のインターフェースを提供することもできます。Business Process Choreographer Explorer のインスタンスは、特定のユーザー・グループのワークフロー・パターンに応じてカスタマイズできます。

デフォルトのユーザー・インターフェースをカスタマイズするには、 Business Process Choreographer Explorer で次のステップを実行します。

1. ナビゲーション・ペインと、このインスタンスのデフォルトのログイン・ビューをカスタマイズします。
   a. タスクバーの「**カスタマイズ**」をクリックします。
   b. 「ナビゲーション・ツリーおよびログイン・ビューのカスタマイズ」ページで、ナビゲーション・ペインに組み込むビューを選択し、ナビゲーション・ペインから除去するビューを選択解除します。
   c. ユーザーが Business Process Choreographer Explorer にログインしたときに表示されるビューを選択します。

      リストには、上のステップで選択したビューと、「ビューの定義」ページで作成したカスタマイズ・ビューが入っています。
   d. 変更を保管するには、「**保管**」をクリックします。

      このインスタンスのビューをデフォルト・ビューに戻すには、「**デフォルトの復元**」をクリックします。このアクションで、ナビゲーション・ペインが事前定義ビューのリストにリセットされます。ナビゲーション・ペインのカスタマイズ・ビューは、このアクションの影響を受けません。
2. **オプション:** ビューをカスタマイズします。

   この Business Process Choreographer Explorer インスタンスのビューに表示される情報は、指定できます。
   a. タスクバーの「**ビューの定義**」をクリックします。
   b. 「検索ビューおよび定義ビュー」ページで、カスタマイズするビューのタイプ (例えばプロセス・テンプレート) を選択します。
   c. ビューの「検索ビューおよび定義ビュー」ページで、そのビューに組み込むプロパティーを選択します。

      これが管理目的のタスク・インスタンス・ビューまたはプロセス・インスタンス・ビューである場合は、「**管理ビュー**」を選択して、ログオン・ユーザーが作業項目を持つ項目のすべてを表示し、ビューのアクション・バーに管理アクションを追加します。ログオン・ユーザーがシステム管理者である場合、システム管理者がそれらの項目の作業項目を持っているかに関係なく、検索条件と一致する項目のすべてが表示されます。

      「**管理ビュー**」を選択しないと、ログオン・ユーザーが閲覧する権限のある全項目が新規ビューに表示されます。

d. 「**リスト名**」 フィールドにビューの表示名を入力し、「**保管**」をクリックします。

新しいビューがナビゲーション・ペインに表示されます。ユーザー・グループのメンバーが次に Business Process Choreographer Explorer にログインすると、新しいビューが表示されます。

ビジネス状態マシンのプロセス・テンプレート用ビューのカスタマイズ:

ビジネス状態マシンのプロセス・テンプレートには、事前定義ビューが用意されていますが、このタイプのテンプレート用に独自のビューを定義することもできます。

カスタマイズ・ビューを作成するには、BPCSystemAdministrator 権限が必要です。

1. タスクバーの「**ビューの定義**」をクリックします。
2. 「検索ビューおよび定義ビュー」ページで、「**プロセス・テンプレートの検索ビューおよび定義ビュー**」を選択します。
3. 「リスト・プロパティー」タブをクリックします。
   a. 「カスタム・プロパティーのリスト列」で、以下の設定でカスタム・プロパティーを追加します。
      - 「**プロパティー名**」フィールドに、`generatedBy` と入力します。
      - 「**プロパティー値**」フィールドに、`BusinessStateMachine` と入力します。
      - 「**表示名**」フィールドに、列の表示名を入力します。
   b. 他のカスタム・プロパティーを追加するか、選択した列のリストに列を追加するか、リストから列を除去します。
4. 「**リスト名**」フィールドに照会の表示名を入力し、「**保管**」をクリックします。
5. ナビゲーション・ペインにビューを追加します。
   a. タスクバーの「**カスタマイズ**」をクリックします。
   b. 「ナビゲーション・ツリーおよびログイン・ビューのカスタマイズ」 ページで、新しいビジネス状態マシン・ビューを選択します。
   c. 変更を保管するには、「**保管**」をクリックします。

ビジネス状態マシン・ビューへのリンクが、ナビゲーション・ペインの「プロセス・テンプレート」グループに追加されます。ユーザーが次回 Business Process Choreographer Explorer にログインすると、このビューが表示されます。

ビジネス状態マシンのプロセス・インスタンス用ビューのカスタマイズ:

ビジネス状態マシンのプロセス・インスタンスには、事前定義ビューが用意されていますが、このタイプのプロセス・インスタンス用に独自のビューを定義することもできます。

カスタマイズ・ビューを作成するには、BPCSystemAdministrator 権限が必要です。

1. タスクバーの「**ビューの定義**」をクリックします。
2. 「検索ビューおよび定義ビュー」ページで、「**プロセス・インスタンスの検索ビューおよび定義ビュー**」を選択します。
3. 「カスタム・プロパティー」タブをクリックします。

a. 以下の設定でカスタム・プロパティーを追加します。

「**プロパティー名**」フィールドに generatedBy、「**プロパティー値**」フィールドに BusinessStateMachine と入力します。

b. 必要に応じて他のカスタム・プロパティーを追加します。

4. 「リスト・プロパティー」タブをクリックします。

a. 「照会プロパティーのリスト列」で、以下の照会プロパティーを追加します。

- ビジネス状態情報をビューに追加するには、「**プロパティー名**」フィールドに name、「**変数名**」フィールドに DisplayState、「**ネーム・スペース**」フィールドに tns を入力します。ここで、tns は、*-process* のサフィックスが付けられたビジネス状態マシンのターゲット・ネーム・スペースです。また、「**表示名**」フィールドに列の表示名を指定します。
- 相関情報をビューに追加するには、「**プロパティー名**」フィールド、「**変数名**」フィールド、「**ネーム・スペース**」フィールドに該当する情報を入力します。これらの値は、ビジネス状態マシンの定義から派生します。また、「**表示名**」フィールドで列の表示名を指定します。

プロパティー名
ビジネス状態マシンに定義した相関プロパティーの名前です。

変数名 相関セットが着信パラメーターによって初期化される場合、変数名は以下の形式になります。

*operation_name*_Input_*operation_parameter_name*

ここで、*operation_name* は、初期状態からの移行操作の名前です。

相関セットが発信パラメーターによって初期化される場合、変数名は以下の形式になります。

*operation_name*_Output_*operation_parameter_name*

ネーム・スペース
照会プロパティーのネーム・スペース。ここで、tns は、*-process* のサフィックスが付けられたビジネス状態マシンのターゲット・ネーム・スペースです。

b. 他のカスタム・プロパティーまたは照会プロパティーを追加するか、選択した列のリストに列を追加するか、リストから列を除去します。

5. 「**リスト名**」フィールドに照会の名前を入力し、「**保管**」をクリックします。
6. ナビゲーション・ペインにビューを追加します。

a. タスクバーの「**カスタマイズ**」をクリックします。

b. 「ナビゲーション・ツリーおよびログイン・ビューのカスタマイズ」ページで、新しいビジネス状態マシン・ビューを選択します。

c. 変更を保管するには、「**保管**」をクリックします。

ビジネス状態マシン・ビューへのリンクが、ナビゲーション・ペインの「プロセス・インスタンス」グループに追加されます。ユーザーが次回 Business Process Choreographer Explorer にログインすると、このビューが表示されます。

Business Process Choreographer Explorer インターフェースの個別設定

デフォルトの Business Process Choreographer Explorer ユーザー・インターフェースのナビゲーション・ペインには、事前定義ビューやシステム管理者が定義したビューへのリンクがいくつか用意されています。ナビゲーション・ペインには、例えば特定のタスクまたはプロセスをモニターするために、独自のビューを追加することができます。

ユーザー・インターフェースの個別設定を行うには、Business Process Choreographer Explorer で次のステップを実行します。

1. ナビゲーション・ツリーの新規ビューを定義するセクションで、「**検索**」アイコン（  ）をクリックします。
2. 新規ビューの「検索ビューおよび定義ビュー」ページで、そのビューに組み込むプロパティーを選択します。

   これが管理目的のタスク・インスタンス・ビューまたはプロセス・インスタンス・ビューである場合は、「**管理ビュー**」を選択して、管理権限を持つ項目のすべてを表示し、ビューのアクション・バーに管理アクションを追加します。

   「**管理ビュー**」を選択しない場合は、閲覧する権限のあるすべての項目が新規ビューに表示されます。
3. 「**リスト名**」 フィールドにビューの表示名を入力し、「**保管**」をクリックします。

新しいビューがナビゲーション・ペインに表示されます。

デフォルトの Web アプリケーションの外観の変更

Business Process Choreographer Explorer は、JavaServer Pages (JSP) ファイルと JavaServer Faces (JSF) ファイルに基づいた、すぐに使用できる Web ユーザー・インターフェースを備えています。ユーザー・インターフェースを改変して、特定のルック・アンド・フィールに合わせるために、新規コードを記述する必要はありません。

この Web インターフェースのレンダリング方法は、カスケーディング・スタイル・シート (CSS) によって制御されます。 CSS を変更することによって、例えばデフォルトのインターフェースをコーポレート・アイデンティティーの指針と合致させることができます。

1. **オプション:** ヘッダーを変更します。 ユーザー・インターフェースには、常に `Menubar.jsp` ファイルが表示されています。 デフォルトの `Menubar.jsp` ファイルには、ロゴ、画像、およびインフォメーション・センターへのリンクが含まれます。
2. **オプション:** スタイル・シートを変更します。 デフォルトのスタイル・シートである `style.css` には、ヘッダー、ナビゲーション・ペイン、コンテンツ・ペインの要素のスタイルが収容されています。

Business Process Choreographer Explorer のインターフェースで使用されるスタイル:

style.css ファイルには、デフォルトのユーザー・インターフェースのルック・アンド・フィールを改変するために変更できるスタイルが格納されています。

style.css ファイルには、デフォルトのユーザー・インターフェースの以下の要素に対応するスタイルが格納されています。

- 『ページの本体』
- 『ログイン・ページ』
- 140 ページの『メニュー・バー』
- 140 ページの『ナビゲーター』
- 140 ページの『コンテンツ・パネル』
- 140 ページの『コマンド・バー』
- 141 ページの『リスト』
- 141 ページの『詳細パネル』
- 141 ページの『メッセージ・データ』
- 141 ページの『タブ付きペイン』
- 142 ページの『検索ページ』
- 142 ページの『エラー詳細』
- 142 ページの『ソート』

このファイルは、次のディレクトリーにあります。

```
<profile_directory>¥installedApps¥<node_name>¥<explorer_instance>¥bpcexplorer.war¥theme
```

ページの本体

スタイル名	説明
.pageBody	2 つの列 (ナビゲーターおよびコンテンツ) がある表レイアウトのメイン・コンテンツ領域。
.pageBody td	ページ本体レイアウト全体の中の個々のセル。
.pageBodyNavigator	ナビゲーターを収容する列。
.pageBodyContent	コンテンツを収容する列。

ログイン・ページ

スタイル名	説明
.loginPanel	ログイン・フォームを収容するパネル。
.loginTitle	フォームの表題。
.loginText	説明文。
.loginForm	入力コントロールを収容するフォーム。
.loginValues	コントロールのレイアウトを決定する表。
.loginField	ログオン・フィールドに使用されるラベル。例えば、「名前」や「パスワード」。
.loginValue	テキスト入力フィールド。

メニュー・バー

スタイル名	説明
.menubar	JSF サブビュー。
.menuContainer	メニュー項目を含むコンテナー・パネル。例えば、ラベルやリンク。
.menuItem	メニュー・バーの項目。
.menuitem a	リンクになっているメニュー項目。
.menuitem a:visited	ユーザーがアクセス済みのリンクを表すメニュー項目。
.menuitem a:hover	リンクになっているメニュー項目上での移動。

ナビゲーター

スタイル名	説明
.navigator	リストへのリンクを含むナビゲーターの JSF サブビュー。
.navigatorTitle	ナビゲーター・ボックスごとの表題。
.navigatorFrame	(例えば、境界線を描画する場合の) ナビゲーター・ボックスごとの境界。
.taskNavigatorTitle	ナビゲーション・ボックスの表題のクラス。ビジネス・プロセス・オブジェクトのリストへのリンクとヒューマン・タスク・オブジェクトのリストへのリンクを区別するために使用します。
.navigatorItem	ナビゲーター・ボックス内の項目。
.navigatorItemList	リストを表す項目。
.expanded / .expanded div / .expanded a .expanded a:visited	ナビゲーター・ボックスの展開時に使用します。
.collapsed	ナビゲーター・ボックスの縮小時に使用します。

コンテンツ・パネル

スタイル名	説明
.panelContainer	リスト、詳細、メッセージのいずれかが入っている分割パネル。 この要素は pageBodyContent 列に埋め込まれます。
.panelTitle	表示コンテンツの表題。例えば、「自分のタスク」。
.panelHelp	ヘルプ・テキストやアイコンが入っている分割コンテナー。
.panelGroup	コマンド・バーおよびリスト、詳細、メッセージのいずれかが入っている分割コンテナー。

コマンド・バー

スタイル名	説明
.commandbarHeader	コマンド・バーの上にある表題。
.commandbar	コマンド・バー領域の周囲の分割コンテナー。

リスト

スタイル名	説明
.listHeader	リストのヘッダー行に使用されているスタイル。
.list	複数の行を含む表。
.list tbody td、.list th	ヘッダー行のスタイル。
.list thead input、.list tbody input	リストのチェック・ボックス。
.list tfoot td	最終行の項目はフッター情報です。
.list tfoot div	フッター要素の周囲の分割コンテナー。
.list tfoot input	フッター内の入力コントロール。
.list a / .list a:visited	リスト内に表示されるリンクが対象です。

詳細パネル

スタイル名	説明
.details	詳細パネルの周囲の分割コンテナー。
div.details	分割コンテナーに組み込まれる詳細スタイル。
table.details	表コンテナーに組み込まれる詳細スタイル。
td.detailsProperty	プロパティー名のラベル。
td.detailsValue	プロパティー値のテキスト。

メッセージ・データ

スタイル名	説明
.messageData	メッセージの周囲の分割コンテナー。
.messageData table	メッセージの格納先となる表コンテナー。
.messageDataButton	メッセージ・フォームの「追加」ボタンと「削除」ボタンのボタン・スタイル。
.messageData td / .messageData th	本体セルとヘッダー・セル。
.messageDataOutput	読み取り専用テキストの表示用。
.messageDataValidInput	有効なメッセージ値用。
.messageDataInvalidInput	無効なメッセージ値用。

タブ付きペイン

スタイル名	説明
.tabbedPane	すべてのタブ付きペインの周囲の分割コンテナー。
.tabHeader	タブ付きペインのタブ・ヘッダー。
.tabHeader ul	各ヘッダーは番号なしリストに編成されます。
.tabHeader li	各ヘッダー・ラベルはリスト項目です。

スタイル名	説明
.tabHeader a / .tabHeader a:hover / .tabHeader a.tab	リンクとしてのヘッダー・ラベル。
.tabHeader a.selectedTab	選択状態のタブ・ヘッダー。
.tabPane	タブ付きペインを囲む分割コンテナー。
.tabPane table	ペインは必ずパネル格子に埋め込まれます。この動作の結果、ペインの周囲に表コンテナーが配置されます。
.tabPane .list th, .tabPane .list tfoot div	タブ付きペイン上にあるリストの設定。

検索ページ

スタイル名	説明
.searchPanel	検索パネル用のタブ付きペイン。タブ付きペインも参照してください。
.searchPanelFilter	検索書式用の表コンテナー。
.searchLabel	検索書式コントロールのラベル。
.searchListBox	選択オプション用のリスト・ボックス・コントロール。

エラー詳細

スタイル名	説明
.errorPage	エラー・ページ用のタブ付きペイン。
.errorLink / .errorLink a / .errorLink a:visited	ページ上にボタン・リンクをレンダリングするために使用するスタイル。
.errorDetails	エラー詳細を表示するタブ付きペイン。
.errorDetailsStack	例外スタックを表示するタブ付きペイン。
.errorDetailsStack table / .errorDetailsStack td	表の行として表示される例外スタック。
.errorDetailsMessage	エラー・メッセージのテキスト・スタイル。

ソート

スタイル名	説明
.ascending	リストをこの列で昇順にソートする場合のリスト・ヘッダー・クラスのスタイル。
.descending	リストをこの列で降順にソートする場合のリスト・ヘッダー・クラスのスタイル。
.unsorted	リストをこの列でソートしない場合のリスト・ヘッダー・クラスのスタイル。

入力フォームと出力フォームのカスタマイズ

Business Process Choreographer Explorer インターフェースには、ビジネス・データの表示や入力のためのデフォルトの入力フォームおよび出力フォームが用意されています。 JSP 文書を使用すると、これらのデフォルトの入力フォームおよび出力フォームをカスタマイズできます。

ユーザー定義の JavaServer Pages (JSP) 文書を Web クライアントに組み込むには、WebSphere Integration Developer でヒューマン・タスクをモデル化するときにこれらの文書を指定する必要があります。 例えば、JSP 文書の指定先は、特定のタスクやその入出力メッセージ、特定のユーザーのロールまたはすべてのユーザーのロールのいずれでも構いません。ユーザー定義 JSP 文書は、出力データを表示して入力データを収集するために実行時にユーザー・インターフェースに組み込まれます。

カスタマイズ・フォームは自己完結した Web ページではなく、Business Process Choreographer Explorer によって HTML フォームに組み込まれる HTML フラグメントです。例えば、ラベルや入力フィールドのフラグメントが該当します。

カスタマイズ・フォームがあるページでボタンをクリックすると、入力データは Business Process Choreographer Explorer に送信されて検証されます。検証は、提供されたプロパティーのタイプとブラウザーで使用されているロケールに基づいて行われます。 入力データを検証できない場合は同じページがもう一度表示され、検証エラーの情報が messageValidationErrors 要求属性に書き込まれます。

カスタマイズ・フォームを Business Process Choreographer Explorer に追加するには、WebSphere Integration Developer を使用して以下のステップを実行します。

1. カスタマイズ・フォームを作成します。

   Web インターフェースで使用される、入出力フォーム用のユーザー定義 JSP 文書は、メッセージ・データにアクセスします。要求コンテキストからビジネス・データにアクセスするには、Java 断片または JSP 実行言語を使用します。
2. JSP 文書にタスクを割り当てます。

   ヒューマン・タスク・エディターでヒューマン・タスクを開きます。 クライアント設定で、ユーザー定義 JSP 文書の場所と、カスタマイズ・フォームの適用先のロール (例: 管理者) を指定します。Business Process Choreographer Explorer のクライアント設定は、タスク・テンプレートに格納されます。これらの設定は、実行時にタスク・テンプレートを使用して取得されます。
3. ユーザー定義 JSP 文書を Web アーカイブ (WAR ファイル) にパッケージ化します。

   WAR ファイルは、タスクが格納されているモジュールと一緒にエンタープライズ・アーカイブに組み込むことも、個別に配置することもできます。

カスタマイズ・フォームは、Business Process Choreographer Explorer で実行時にレンダリングされます。

ユーザー定義 JSP フラグメント:

ユーザー定義 JSP フラグメントは、HTML フォーム・タグに埋め込まれます。 Business Process Choreographer Explorer は、実行時にこれらのフラグメントを、レンダリングされるページに埋め込みます。

入力メッセージのユーザー定義 JSP フラグメントは、出力メッセージの JSP フラグメントより先に埋め込まれます。

```
<html....>
   ...
   <form...>
     Input JSP (display task input message)

     Output JSP (display task output message)

   </form>
   ...
</html>
```

ユーザー定義 JSP フラグメントは、HTML フォーム・タグに埋め込まれているため、入力要素を追加できます。 入力要素の名前は、データ要素の XML Path Language (XPath) 表現と一致する必要があります。入力要素の名前には、以下に示す提供されたプレフィックス値を使用してプレフィックスを付けることが重要です。

```
<input id="address"
       type="text"
       name="${prefix}/selectPromotionalGiftResponse/address"
       value="${messageMap['/selectPromotionalGiftResponse/address"]}
       size="60"
       align="left" />
```

プレフィックス値は要求属性として提供されます。この属性により、引用符で囲まれた書式内の入力名は固有であることが保証されます。プレフィックスは Business Process Choreographer Explorer によって次のように生成されるため、変更しないでください。

```
String prefix = (String)request.getAttribute("prefix");
```

プレフィックス要素が設定されるのは、与えられたコンテキストにおいてメッセージが編集できる場合に限ります。 出力データは、ヒューマン・タスクの状態に応じてさまざまな方法で表示できます。例えば、タスクが要求状態にある場合は、出力データを変更できます。ただし、タスクが完了状態にある場合、出力データは表示専用になります。JSP フラグメントでは、プレフィックス要素が存在し、それに応じてメッセージを表示するかどうかをテストできます。 以下の JSTL ステートメントでは、プレフィックス要素が設定されているかどうかをテストする 1 つの方法を示しています。

```
...
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%
...
<c:choose>
  <c:when test="${not empty prefix}">
    <!--Read/write mode-->
  </c:when>
  <c:otherwise>
    <!--Read-only mode-->
  </c:otherwise>
</c:choose>
```

Business Process Choreographer の管理

Business Process Choreographer は、管理コンソールまたはスクリプトを使用して管理できます。

管理コンソールによる Business Process Choreographer の管理

管理コンソールを使用して実行可能な管理操作について説明します。

サーバーの補正サービスの管理

管理コンソールを使用して、アプリケーションの始動時に、自動的に補正サービスを開始し、リカバリー・ログの場所および最大サイズを指定します。

ビジネス・プロセスがアプリケーション・サーバーで実行されるときは、そのサーバーに補正サービスが開始されている必要があります。プロセスが完了する前に、いくつかのトランザクションで行われた更新を管理するため、補正サービスが使用されます。

管理コンソールを使用して、アプリケーション・サーバーの補正サービスのプロパティーを表示および変更することができます。

1. 管理コンソールを表示します。
2. ナビゲーション・ペインで、「**サーバー**」 → 「**アプリケーション・サーバー**」 → ***server_name*** をクリックします。
3. 「構成」タブの「コンテナーの設定」の下で、「**コンテナー・サービス (Container services)**」 → 「**補正サービス (Compensation service)**」をクリックします。 このアクションにより、パネルに補正サービス・プロパティーが表示されます。

   サーバー始動時にサービスを使用可能にする
   アプリケーション・サーバーの始動時は必ず、補正サービスの開始を試行するように指定します。

   このチェック・ボックスが選択されていることを確認します。ビジネス・プロセスを使用する際は、補正サービスが使用可能になっている必要があります。クラスターでビジネス・プロセスを実行する場合、クラスターの各サーバーで補正サービスを使用可能にする必要があります。

   リカバリー・ログ・ディレクトリー
   補正サービスがリカバリー用のログ・ファイルを保管する、このサーバーでのディレクトリー名を指定します。補正を使用すると、WebSphere 製品が、補正に必要な情報を保管します。

   リカバリー・ログ・ファイル・サイズ
   このアプリケーション・サーバーでの補正ログ・ファイルの最大サイズを MB 単位で指定します。
4. **オプション:** 必要に応じて、補正サービス・プロパティーを変更します。
5. 「**OK**」をクリックします。

6. 構成を保管するには、管理コンソール・ウィンドウのメッセージ・ボックスで「**保管**」をクリックします。次に「アプリケーション・サーバーの保管 (Application Servers Save)」ペインの「**保管**」をクリックします。

管理コンソールを使用した、失敗したメッセージの照会と再生

ここでは、処理できなかったビジネス・プロセスまたはヒューマン・タスクのメッセージの有無を確認し、メッセージが存在する場合には再生する方法について説明します。

メッセージの処理中に問題が発生すると、そのメッセージは保存キューまたは保留キューに移されます。このタスクでは、失敗したメッセージが存在するかどうかを判別する方法と、それらのメッセージを内部キューへ再び送信する方法を説明します。

1. 保留キューおよび保存キューにメッセージがいくつかあるかを確認します。
   a. 「**サーバー**」 → 「**アプリケーション・サーバー**」 → ***server_name*** をクリックします。
   b. 「**構成**」タブの「コンテナーの設定」セクションで、以下の手順の 1 つをクリックします。
      - ビジネス・プロセスの場合: 「**ビジネス・プロセス・コンテナーの設定**」 → 「**ランタイム構成**」
      - ヒューマン・タスクの場合: 「**ヒューマン・タスク・コンテナーの設定**」 → 「**ランタイム構成**」

      保留キューおよび保存キュー上のメッセージの数が、「**一般プロパティー**」に表示されます。
2. 保留キューまたは保存キューにメッセージが含まれている場合は、内部作業キューへメッセージを移動できます。

   以下のいずれかのオプションをクリックします。
   - ビジネス・プロセスの場合: 「**保留キューの再生**」および「**保存キューの再生**」
   - ヒューマン・タスクの場合: 「**保留キューの再生**」

   注: セキュリティーが使用可能になっている場合は、オペレーター権限を持つユーザーに対してのみ再生ボタンが表示されます。

Business Process Choreographer は、再生された全メッセージを再び処理しようとします。

失敗したメッセージ数の最新表示:

管理コンソールを使用して、ビジネス・プロセスまたはヒューマン・タスクの失敗したメッセージの数を最新表示します。

保留キューおよび保存キュー上に表示されたメッセージの数、およびメッセージ例外の数は、最新表示されるまで静的なままになっています。このタスクでは、これらのキュー上のメッセージ数、およびメッセージ例外の数の更新および表示方法について説明します。

1. 該当するアプリケーション・サーバーを選択します。

「**サーバー**」 → 「**アプリケーション・サーバー**」 → ***server_name*** をクリックします。

2. メッセージ数を最新表示します。

「**構成**」タブの「コンテナーの設定」セクションで、以下の手順の 1 つをクリックします。

- ビジネス・プロセスの場合: 「**ビジネス・プロセス・コンテナーの設定**」 → 「**ランタイム構成**」 → 「**メッセージ・カウントの更新**」
- ヒューマン・タスクの場合: 「**ヒューマン・タスク・コンテナーの設定**」 → 「**ランタイム構成**」 → 「**メッセージ・カウントの更新**」

以下の更新済み値が、「**一般プロパティー**」に表示されます。

- ビジネス・プロセスの場合: 保留キューおよび保存キュー上のメッセージの数
- ヒューマン・タスクの場合: 保留キュー上のメッセージの数
- キューへのアクセス中に何らかの例外が発生すると、「メッセージ例外」フィールドにメッセージ・テキストが表示されます。

このページで、これらのキューに対するメッセージを再生することもできます。

メッセージ処理の失敗と静止モード:

Business Process Choreographer は、一時的なインフラストラクチャー障害を処理するための機能を提供します。

このセクションでは、失敗したメッセージをビジネス・プロセス・コンテナーによって処理する仕組みを説明します。この仕組みは、150 ページの『ヒューマン・タスクでの失敗したメッセージ処理』で説明するヒューマン・タスク・コンテナーによって使用される簡略化された仕組みとは対照的です。

長期実行プロセスは、一連のトランザクションで構成されています。トランザクションは、Java Message Service (JMS) メッセージによって分離されます。サーバーは、メッセージ駆動型 Bean に送信します。この Bean の処理は、着信メッセージをプロセス・サーバーに受け渡すことによって行います。それぞれのトランザクションは、以下のアクションで構成されています。

- メッセージを受信します。
- メッセージを代行してナビゲートします。
- 後続のトランザクションを起動するメッセージを送信します。

サーバーは、以下の理由のいずれかで、メッセージ駆動型 Bean が受信したメッセージの処理に失敗することがあります。

- 指定された数の連続メッセージは処理されませんでした。この場合、インフラストラクチャーが使用不可になっていると考えられます。
- 一部のメッセージしか処理できません。処理できなかったすべての単一メッセージは、破損していると考えられます。

これらの原因に対する対応は、以下のとおりです。

原因	応答
インフラストラクチャーが使用不可	メッセージ駆動型 Bean は、指定した期間、この状態からのリカバリーを試行します。この Bean は、サーバーが再び作動可能になるまで、すべてのメッセージを使用可能な状態に保持するように試行します。例えばこの問題は、データベース障害によって発生している場合があります。
メッセージの破損	指定した回数の試行後、メッセージは保留キューに書き込まれます。この保留キューで、メッセージの操作および検討を行います。保留キューから入力キューに戻して、トランザクションを再試行することもできます。

ビジネス・プロセス用のメッセージのインプリメンテーションは以下のとおりです。

- メッセージの処理が失敗した場合は、サーバーはそのメッセージを保存キューに書き込みます。この保存キューで、使用可能な状態が保持されます。これは、指定した時間内でインフラストラクチャーの問題が解決される場合です。
- メッセージが保存キューにある場合、オプションは以下のとおりです。
  - 後続のメッセージが正常に処理できる場合、保存キューからのすべてのメッセージは、メッセージ駆動型 Bean の入力キューに戻されます。それぞれのメッセージごとに、メッセージが保存キューに送信された頻度数が保持されます。この数が所定のメッセージの再試行限度を超えた場合、メッセージは保留キューに書き込まれます。
  - 次のメッセージの処理に失敗した場合、そのメッセージも保存キューに書き込まれます。この処理は、保存キューの最大メッセージしきい値に到達するまで継続します。このしきい値に到達すると、メッセージ駆動型 Bean は、保存キューからすべてのメッセージを入力キューに戻し、静止モードにスイッチします。

メッセージ駆動型 Bean が静止モードで作動している場合、この Bean は、定期的にメッセージの処理を試行します。処理に失敗したメッセージは、配信回数を増加させたり、保存キューの巡回数を増加させたりすることなく、保留キューに書き戻されます。メッセージの処理を正常に行うことが可能になったら、即時にメッセージ駆動型 Bean は標準処理モードにスイッチバックします。

この機能は、2 つの数値限度、2 つのキュー、静止モード、およびメッセージ再試行の動作から構成されています。

再試行限度

再試行限度は、保留キューに書き込む前に保存キューでメッセージを転送できる最大回数を定義しています。

保存キューに書き込まれるには、メッセージの処理が 3 回失敗する必要があります。

例えば、再試行限度が 5 である場合は、メッセージが保存キューを 5 回通過してから (3 * 5 = 15 回失敗してから) 最後の再試行ループが開始されます。最後の再

試行ループでさらに 2 回失敗すると、メッセージは保留キューに入れられます。つまり、メッセージは、(3 * *RetryLimit*) + 2 回失敗してから保留キューに入れられます。

信頼できるインフラストラクチャーで実行中の、パフォーマンスが重要なアプリケーションでは、再試行限度を少なく、例えば 1 または 2 にしておく必要があります。

管理コンソールでこのパラメーターを見つけるには、「**サーバー**」 → 「**アプリケーション・サーバー**」 → ***server_name*** をクリックします。次に、「ビジネス・プロセス・コンテナー設定 (Business Process Container Settings)」見出しで、「**ビジネス・プロセス・コンテナー**」をクリックします。

保存キュー・メッセージ限度

保存キュー・メッセージ限度は、保存キューに入れることができるメッセージの最大数を定義します。保存キューがオーバーフローすると、システムは静止モードに入ります。 1 つのメッセージが失敗したら即時にシステムが静止モードに入るようにするには、値をゼロに設定します。ビジネス・プロセス・コンテナーのインフラストラクチャー障害に対する耐性を強化するには、値を増やします。

管理コンソールでこのパラメーターを見つけるには、「**サーバー**」 → 「**アプリケーション・サーバー**」 → ***server_name*** をクリックします。次に、「ビジネス・プロセス・コンテナー設定 (Business Process Container Settings)」見出しで、「**ビジネス・プロセス・コンテナー**」をクリックします。

保存キュー

保存キューは、障害を起こしたメッセージを保存します。これらは、ビジネス・プロセス・コンテナーの内部作業キューへ戻すことによって再生されます。メッセージは、3 回失敗したら保存キューに書き込まれます。メッセージが (3 * *RetryLimit*) + 2 回失敗した場合は、保留キューに書き込まれます。 (再試行限度について詳しくは、 148 ページの『再試行限度』を参照してください。) 保存キューが保存キュー・メッセージ限度で定義された限度に達してから別のメッセージが失敗すると、キューはオーバーフローし、システムは静止モードに入ります。管理者は、失敗したメッセージの照会と再生のタスクを実行して、このキュー内のメッセージを内部キューに戻すことができます。

保留キュー

保留キューには、 (3 * *RetryLimit*) + 2 回失敗したメッセージが含まれます。 (再試行限度について詳しくは、 148 ページの『再試行限度』を参照してください。) 管理者は、失敗したメッセージの照会と再生のタスクを実行して、このキュー内のメッセージを内部キューに戻すことができます。

メッセージの再生

管理者は、保留キューまたは保存キューから内部キューへメッセージを戻すことができます。この操作は、管理コンソールか管理コマンドを使用して実行できます。

静止モード

保存キューがオーバーフローすると、静止モードに入ります。これが起こると、場合によっては一時的だが重大なインフラストラクチャー障害が発生したと想定されます。静止モードの目的は、システムが多数のリソースを使用しないようにすることですが、インフラストラクチャー障害は、ほとんどのメッセージがいずれにせよ失敗する可能性があることを意味します。静止モードでは、次のメッセージの処理を試行する前に、2 秒間のシステム・スリープがあります。メッセージが正常に処理されると即時に、システムは標準メッセージ処理を再開します。

ヒューマン・タスクでの失敗したメッセージ処理

ヒューマン・タスク・コンテナーには、保存キューや再試行限度がありません。保留キューがあるだけで、失敗したメッセージはこのキューに置かれ、このキューから再生することができます。

管理コンソールを使用したスタッフ照会結果の最新表示

スタッフ照会の結果は静的です。管理コンソールを使用して、スタッフ照会を最新表示します。

Business Process Choreographer は、ランタイム・データベースで、Lightweight Directory Access Protocol (LDAP) サーバーなど、スタッフ・ディレクトリーに対して評価されたスタッフ割り当ての結果をキャッシュに入れます。スタッフ・ディレクトリーを変更する場合は、スタッフ割り当てを再評価するよう強制できます。

スタッフ照会を最新表示するには、以下の手順を実行します。

1. **「サーバー」** → **「アプリケーション・サーバー」** → ***server_name*** をクリックします。
2. **「構成」**タブの「コンテナーの設定」セクションで、**「ヒューマン・タスク・コンテナーの設定」** → **「ランタイム構成」** → **「スタッフ照会の更新」**をクリックします。 すべてのスタッフ照会は、更新されます。

注: セキュリティーが使用可能になっている場合は、オペレーター権限を持つユーザーに対してのみ更新ボタンが表示されます。

スタッフ照会結果をこのようにして最新表示すると、アプリケーションおよびデータベースに対して高い負荷が発生することがあります。以下にリストされた代替方式を使用することを考慮してください。

関連タスク

162 ページの『管理コマンドを使用したスタッフ照会結果の最新表示』
スタッフ照会の結果は静的です。管理コマンドを使用して、スタッフ照会を最新表示します。

164 ページの『更新デーモンを使用したスタッフ照会結果の更新』
期限切れのすべてのスタッフ照会結果を定期的に自動で更新するようにセットアップしたい場合は、 このメソッドを使用します。

Common Base Event および監査証跡の使用可能化

このタスクを使用して、Business Process Choreographer イベントを Common Event Infrastructure に Common Base Event として放出するか、監査証跡に保管するか、またはその両方を行えるようにします。

ビジネス・プロセス・コンテナーまたはヒューマン・タスク・コンテナーの状態監視者設定は、「構成」タブで永続的に、または「ランタイム」タブで一時的に変更することができます。これらの「構成」タブまたは「ランタイム」タブでの選択は、該当するコンテナーで実行されるすべてのアプリケーションに影響します。変更内容をビジネス・プロセス・コンテナーとヒューマン・タスク・コンテナーの両方に反映させるには、それぞれの設定を別々に変更する必要があります。

構成済みロギング・インフラストラクチャーの変更:

このタスクを使用して、監査ログの状態監視者ロギング、または構成の Common Event Infrastructure ロギングを変更します。

「構成」タブで行われた選択は、次にサーバーを開始したときにアクティブになります。選択した設定は、サーバーを開始するたびに適用されます。

以下のように構成を変更します。

1. 「ビジネス・プロセス・コンテナー」または「ヒューマン・タスク・コンテナー」ペインを表示します。

   「サーバー」 → **「アプリケーション・サーバー」** → ***server_name*** をクリックします。次に、**「コンテナー設定」**で、以下のいずれかの手順をクリックします。
   - ビジネス・プロセスの場合: **「ビジネス・プロセス・コンテナーの設定」** → **「ビジネス・プロセス・コンテナー」**
   - ヒューマン・タスクの場合: **「ヒューマン・タスク・コンテナーの設定」** → **「ヒューマン・タスク・コンテナー」**
2. 「一般プロパティー」セクションで、インプリメントするロギングを選択します。 状態監視者は、それぞれ独立しています。どちらかいずれか、またはその両方を使用可能にしたり、使用不可にすることができます。

   Common Event Infrastructure ロギングの使用可能化
   このチェック・ボックスを選択して、Common Event Infrastructure に基づくイベント放出を使用可能にします。

   監査ロギングの使用可能化
   このチェック・ボックスを選択して、リレーションシップ・データベースの監査証跡テーブルに、監査ログ・イベントを保管します。
3. 変更を受け入れます。
   a. **「適用」**をクリックします。
   b. メッセージ・ボックスで**「保管」**をクリックします。
   c. 「アプリケーション・サーバー」ペインで**「保管」**をクリックします。

状態監視者は、必要に応じて設定します。変更は、サーバーの再始動後に有効になります。

変更を有効にするには、コンテナーを再始動します。

セッション用ロギング・インフラストラクチャーの構成:

このタスクを使用して、監査ログの状態監視者ロギング、またはセッションの Common Event Infrastructure ロギングを変更します。

「ランタイム」タブで行った選択は、即時に有効になります。選択した設定は、次にサーバーを開始するまで有効です。

以下のように、セッション・インフラストラクチャーを変更します。

1. 「メッセージの再生」ペインを表示します。

   「サーバー」 → **「アプリケーション・サーバー」** → ***server_name*** をクリックします。次に、「コンテナー設定」で、以下のいずれかの手順をクリックします。
   - ビジネス・プロセスの場合: **「ビジネス・プロセス・コンテナーの設定」** → **「ランタイム構成」**
   - ヒューマン・タスクの場合: **「ヒューマン・タスク・コンテナーの設定」** → **「ランタイム構成」**
2. **「状態監視者ロギング」**セクションで、インプリメントされるロギングを選択します。状態監視者は、それぞれ独立しています。どちらかいずれか、またはその両方を使用可能にしたり、使用不可にすることができます。

   Common Event Infrastructure ロギングの使用可能化
   このチェック・ボックスを選択して、Common Event Infrastructure に基づくイベント放出を使用可能にします。

   監査ロギングの使用可能化
   このチェック・ボックスを選択して、リレーションシップ・データベースの監査証跡テーブルに、監査ログ・イベントを保管します。

   ランタイム変更も構成に保管
   このチェック・ボックスを選択して、このセッションで行った変更で構成を更新します。
3. 変更を受け入れます。

   「OK」をクリックします。

状態監視者は、必要に応じて設定します。

イベントの放出および保管:

状態変更のイベントは、ビジネス・プロセス、ヒューマン・タスク、またはその両方を実行する場合に生成される可能性があります。

イベントがアプリケーションによって検索されるように、2 つのインフラストラクチャーは、イベントの放出または保管を行います。アプリケーションでは、イベントを使用して、ビジネス・プロセスをモニターし、ビジネス・プロセスまたはヒューマン・タスクのヒストリーを分析することがあります。

例えばタスク・イベントは、ビジネス・プロセスが関係しなくても発生する可能性があります。これらのイベントは、監査証跡および Common Event Infrastructure (CEI) によってコンシュームされることもあります。これは、スタンドアロン・タスク、全くのヒューマン・タスク、およびビジネス・プロセス以外のアプリケーション・コンポーネントによって呼び出されるタスクに当てはまります。

イベントの生成はシステム・パフォーマンスに影響を与えるため、イベントの保管および放出を行うインフラストラクチャーを選択できます。

Common Event Infrastructure

イベントは、サブスクライブ・アプリケーションに保管および公表することができます。このイベント・インフラストラクチャーを使用するために、Common Event Infrastructure がインストールされ、構成されていることを確認してください。

Common Event Infrastructure に基づくイベントの放出を使用し、Common Event Infrastructure のアプリケーション・プログラミング・インターフェース (API) を経由して、Common Base Event 形式でイベントを検索します。サブスクリプション、または Common Event Infrastructure の照会指向インターフェースを使用して、利用中のアプリケーションに接続できます。

Common Event Infrastructure に基づくイベントの放出は、監査ログ・イベントよりも、システム・パフォーマンスに重大な影響を与えます。ただし、利用中のアプリケーションには、非常に大きな柔軟性が付与されます。

監査証跡

イベントは、リレーションシップ・データベース内にあるテーブルのレコードとして保管されます。

これは、パフォーマンスにほとんど影響を与えない高速イベント保管インフラストラクチャーです。利用中のアプリケーションがデータベースからイベントを検索するには、Structured Query Language (SQL) 照会が必要です。

インフラストラクチャーのいずれか、あるいは両方を選択することも、またはどちらも選択しないこともできます。インフラストラクチャーの選択では、イベントの保管または放出は必ずしも暗黙指定されません。選択によって、インフラストラクチャーが使用可能になります。後でメカニズムを追加することによって、実際のイベントの生成を制御できます。ただし、インフラストラクチャーを使用可能にすると、システム・パフォーマンスに影響する基本的なオーバーヘッドが発生します。

スクリプトによる Business Process Choreographer の管理

スクリプトを使用して実行可能な管理操作について説明します。

管理コマンドを使用した、監査ログ・エントリーの削除

管理コマンドを使用して、監査ログ・エントリーの一部またはすべてを削除します。

この手順を始める前に、次の条件が満たされている必要があります。

- 監査ログ・エントリーの削除に使用するアプリケーション・サーバーが稼働している必要があります。つまり、サーバー接続が必要であるため、wsadmin の `-conntype none` オプションは使用できません。
- セキュリティーが使用可能に設定されている場合は、使用するユーザー ID がオペレーター権限を持っている必要があります。

`deleteAuditLog.py` スクリプトを使用すると、データベースから監査ログ・エントリーを削除できます。

1. 管理スクリプトが置かれている Business Process Choreographer サブディレクトリーに移動します。

   以下のコマンドを入力します。

```
cd install_root/ProcessChoreographer/admin
```

2. 監査ログ・テーブルでエントリーを削除します。

   次のコマンドを 1 つ以上入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin –lang jython -f deleteAuditLog.py
                         -server serverName
                         [-profile profileName]
                         [options]

install_root/bin/wsadmin –lang jython -f deleteAuditLog.py
                         -node nodeName
                         -server serverName
                         [-profile profileName]
                         [options]

install_root/bin/wsadmin –lang jython -f deleteAuditLog.py
                         -cluster clusterName
                         [-profile profileName]
                         [options]
```

   各部の意味は、次のとおりです。

   -cluster *clusterName*
   クラスターの名前。ビジネス・プロセス・コンテナーが WebSphere クラスター用に構成されている場合は必須です。

   -node *nodeName*
   サーバー名を指定する場合のオプション。この名前は、ノードを示します。デフォルトはローカル・ノードです。

   -server *serverName*
   サーバーの名前。クラスター名が指定されていない場合は必須です。

   -profileName *profileName*
   ユーザー定義プロファイルの名前。デフォルト・プロファイルで作業していない場合、このオプションを指定します。

   使用可能なオプションは以下のとおりです。

   -all
   データベース内の監査ログ・エントリーをすべて削除します。複数のトランザクションで削除が実行されます。各トランザクションでは、slice パラメーターで指定されたエントリー数またはデフォルト数が削除されます。

   -time *timestamp*
   timestamp に指定した時刻より古い監査ログ・エントリーすべてが削除されます。使用される時刻は、協定世界時 (UTC) です。その時刻形式は、`YYYY-MM-DD['T'HH:MM:SS]` となっている必要があります。年月日だけを指定する場合、時間と分と秒は `00:00:00` に設定されます。

   -time および *-processtime* オプションは、相互に排他的です。

   -*processtime timestamp*
   timestamp で指定した時刻より前に完了したプロセスに属す監査ログ・エントリーがすべて削除されます。*-time* パラメーターの場合と同じ時刻形式を使用します。

-time および *-processtime* オプションは、相互に排他的です。

-slice *size*
-all パラメーターとともに使用され、*size* は各トランザクションに含まれるエントリー数を指定します。最適値は、データベース・システムで使用可能なログ・サイズによって決まります。値が高いほど、必要なトランザクションは少なくなりますが、データベース・ロギング・スペースを超過する可能性があります。値が低い場合は、スクリプトでの削除の完了に時間がかかる可能性があります。slice パラメーターのデフォルトのサイズは 250 です。

注: 未使用スタッフ照会のクリーンアップのスクリプトの jacl バージョン `deleteAuditLog.jacl` は、推奨されていません。これは、`ProcessChoreographer` ディレクトリーの `util` サブディレクトリーでは使用可能で、ここで説明しているのと同じパラメーターを使用しますが、`–lang jython` オプションは省略する必要があります。

無効になったプロセス・テンプレートおよびタスク・テンプレートの削除

管理コマンドを使用して、データベースから、無効になったプロセス・テンプレートおよびタスク・テンプレートを削除します。

この手順を始める前に、削除されるテンプレートが置かれているアプリケーション・サーバーが稼働している必要があります。つまり、サーバー接続が必要であるため、`wsadmin` の `-conntype none` オプションは使用できません。セキュリティーが使用可能になっている場合でも、このコマンドを実行するための特殊権限は不要です。

ここで説明するメソッドを使用して、データベースから、テンプレート、およびそのテンプレートに属するすべてのオブジェクト (WebSphere 構成リポジトリー内に、これらのオブジェクトを含んでいる有効な対応するアプリケーションがない場合) を除去します。この状態は、ユーザーによってアプリケーションのインストールが取り消されたか、Configuration Repository に保管されていなかった場合に発生します。これらのテンプレートは、通常影響はありません。これらのテンプレートは、Business Process Choreographer Explorer では表示されません。

これらのテンプレートがフィルタリングされない状況がまれに発生します。この場合には、以下のスクリプトでデータベースから除去される必要があります。

スクリプトを使用して、データベースから有効なアプリケーションのテンプレートを除去することはできません。対応するアプリケーションが有効な場合、この状態は検査され、ConfigurationError 例外がスローされます。

1. 管理スクリプトが置かれている Business Process Choreographer サブディレクトリーに移動します。

   以下のコマンドを入力します。

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. データベースから、無効になったビジネス・プロセス・テンプレート、またはヒューマン・タスク・テンプレートを削除します。

無効になったビジネス・プロセス・テンプレートを削除するには、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin.sh –lang jython -f deleteInvalidProcessTemplate.py
                        -server serverName
                        -template templateName
                        -validFrom validFromString
                        [-profileName profileName]

install_root/bin/wsadmin.sh –lang jython -f deleteInvalidProcessTemplate.py
                        -server serverName
                        -node nodeName
                        -template templateName
                        -validFrom validFromString
                        [-profileName profileName]

install_root/bin/wsadmin.sh –lang jython -f deleteInvalidProcessTemplate.py
                        -cluster clusterName
                        -template templateName
                        -validFrom validFromString
                        [-profileName profileName]
```

無効になったヒューマン・タスク・テンプレートを削除するには、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin.sh –lang jython -f deleteInvalidTaskTemplate.py
                        -server serverName
                        -template templateName
                        -validFrom validFromString
                        -nameSpace nameSpace
                        [-profileName profileName]

install_root/bin/wsadmin.sh –lang jython -f deleteInvalidTaskTemplate.py
                        -server serverName
                        -node nodeName
                        -template templateName
                        -validFrom validFromString
                        -nameSpace nameSpace
                        [-profileName profileName]

install_root/bin/wsadmin.sh –lang jython -f deleteInvalidTaskTemplate.py
                        -cluster clusterName
                        -template templateName
                        -validFrom validFromString
                        -nameSpace nameSpace
                        [-profileName profileName]
```

各部の意味は、次のとおりです。

cluster *clusterName*
クラスターの名前。ビジネス・プロセス・コンテナーが WebSphere クラスター用に構成されている場合は必須です。クラスター名、またはサーバー名およびノード名を指定できます。

node *nodeName*
サーバー名を指定する場合のオプション。この名前は、ノードを示します。デフォルトはローカル・ノードです。サーバー名およびノード名、またはクラスター名を指定できます。

server *serverName*
サーバーの名前。クラスター名が指定されていない場合は必須です。サーバー名およびノード名、またはクラスター名を指定できます。

template *templateName*
除去するプロセス・テンプレートまたはタスク・テンプレートの名前。

validFrom *validFromString*
管理コンソールの表示どおりにテンプレートが有効になった日付 (UTC 形式)。ストリングの形式は、「yyyy-MM-ddThh:mm:ss」(年、月、日、T、時、分、秒) です。例えば、2005-01-31T13:40:50 のようになります。

nameSpace *nameSpace*
タスク・テンプレートのターゲット・ネーム・スペース。

profileName *profileName*
ユーザー定義プロファイルの名前。デフォルト・プロファイルで作業していない場合、このオプションを指定します。

注: 未使用スタッフ照会のクリーンアップのスクリプトの jacl バージョン deleteInvalidTaskTemplate.jacl は、推奨されていません。これは、ProcessChoreographer ディレクトリーの util サブディレクトリーでは使用可能で、ここで説明しているのと同じパラメーターを使用しますが、–lang jython オプションは省略する必要があります。

完了したプロセス・インスタンスの削除

Business Process Choreographer データベースから、終了状態 (終了、強制終了、または失敗) になった最上位のプロセス・インスタンスを選択して削除するには、管理コマンドを使用します。

この手順を始める前に、削除されるプロセス・インスタンスが置かれているアプリケーション・サーバーが稼働している必要があります。つまり、サーバー接続が必要であるため、wsadmin の -conntype none オプションは使用できません。セキュリティーが使用可能になっている場合でも、このコマンドを実行するための特殊権限は不要です。

最上位のプロセス・インスタンスは、終了状態が終了、強制終了、失敗のいずれかであれば、完了したと見なされます。最上位のプロセス・インスタンスと、そのすべての関連データ (アクティビティー・インスタンス、子プロセス・インスタンス、インライン・タスク・インスタンスなど) をデータベースから選択削除する場合の基準を指定します。

1. 管理スクリプトが置かれている Business Process Choreographer サブディレクトリーに移動します。

   以下のコマンドを入力します。

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. データベースからプロセス・インスタンスを削除します。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin –lang jython –f deleteCompletedProcessInstances.py
     [([-node nodeName] -server serverName) | (-cluster clusterName)]
     (-all | -finished | -terminated | -failed )
     [-templateName templateName [-validFrom timestamp]]
     [-startedBy userID ]
     [-completedBefore timestamp]
     [-profileName profileName]
   ```

各部の意味は、次のとおりです。

-node *nodeName*
サーバー名を指定する場合のオプション。この名前は、ノードを示します。デフォルトはローカル・ノードです。サーバー名およびノード名、またはクラスター名を指定できます。

-server *serverName*
サーバーの名前。クラスター名が指定されていない場合は必須です。サーバー名およびノード名、またはクラスター名を指定できます。

-cluster *clusterName*
クラスターの名前。ビジネス・プロセス・コンテナーが WebSphere クラスター用に構成されている場合は必須です。クラスター名、またはサーバー名およびノード名を指定できます。

-all|-finished|-terminated|-failed
状態によってどのプロセス・インスタンスを削除するかを指定します。終了、強制終了、失敗、すべてを組み合わせて指定できます。

-templateName *templateName*
(オプション) 削除するプロセス・テンプレートまたはヒューマン・タスク・テンプレートの名前を指定します。このオプションを指定すると、validFrom も指定できます。

-validFrom *timestamp*
管理コンソールの表示どおりにテンプレートが有効になった日付 (UTC 形式)。このオプションは、templateName オプションと一緒にしか使用できません。 *timestamp* ストリングの形式は、「yyyy-MM-ddThh:mm:ss」(年、月、日、T、時、分、秒) です。例えば、`2006-11-20T12:00:00` のようになります。

startedBy *userID*
(オプション) 特定のユーザー ID が開始した完了済みプロセス・インスタンスのみを削除します。

-completedBefore *timestamp*
(オプション) 指定の時刻より前に完了したプロセス・インスタンスを削除します。*timestamp* ストリングの形式は、「yyyy-MM-ddThh:mm:ss」(年、月、日、T、時、分、秒) です。例えば、`2006-07-20T12:00:00` のようになります。

profileName *profileName*
ユーザー定義プロファイルの名前。デフォルト・プロファイルで作業していない場合、このオプションを指定します。

例えば、サーバー *myServer* のノード *myNode* で実行されるプロセス・インスタンスのうち、状態が終了で、ユーザー Antje が開始したものをすべて削除するには、次のコマンドを実行します。

```
wsadmin –lang jython –f deleteCompletedProcessInstances.py
        -node myNode -server myServer
        -finished
        -startedBy Antje
```

完了済みのプロセス・インスタンスがデータベースから削除されました。

Observer データベースからのデータの削除

Business Process Choreographer Observer データベースから、指定の条件に一致したプロセス・インスタンスのデータをすべて選択して、選択的に削除するには、管理コマンドを使用します。

プロセス・インスタンスの監視者情報を削除するには、3 つの方法があります。

- 指定の時刻より前に終了状態が削除になったプロセス・インスタンスの監視者データを削除する。
- 特定のプロセス・テンプレート・バージョンのプロセス・インスタンスの監視者データを削除する。
- 指定の時刻より前に最後のイベントが受信された場合は、プロセス・インスタンスの状態に関係なく、その監視者データを削除する。

1. 管理スクリプトが置かれている Business Process Choreographer サブディレクトリーに移動します。

   以下のコマンドを入力します。

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. データベースから、プロセス・インスタンスの監視者データを削除します。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin -lang jython
        -f observerDeleteCompletedProcessInstances.py
        [([-node nodeName] -server serverName) | (-cluster clusterName)]
        -dataSource dataSourceJNDIName
        ( -templateName templateName  -validFrom timestamp)
        | -completedBefore timestamp
        | -force -state state -reachedBefore timestamp
        [-profileName profileName]
   ```

   各部の意味は、次のとおりです。

   -node *nodeName*
   この名前は、ノードを示します。デフォルトはローカル・ノードです。このパラメーターはオプションです。

   -server *serverName*
   サーバーの名前。このパラメーターはオプションです。デフォルトでは、デフォルトのサーバーです。

   -cluster *clusterName*
   クラスターの名前。このパラメーターはオプションです。

   -datasource *datasourceJNDIName*
   このコマンドで操作するデータベースを指定します。サーバーまたはクラスターには複数の Observer データベースを入れることができるため、このパラメーターは必須です。

   -templateName *templateName***-validFrom** *timestamp*
   (オプション) 監視者データを削除するプロセス・テンプレートの名前を指定します。このオプションを指定すると、-validFrom オプションも指定する必要があります。

   timestamp ストリングでは、管理コンソールの表示どおりにテンプレートが

有効になった日付 (UTC 形式) を指定します。形式は、「yyyy-MM-ddThh:mm:ss」(年、月、日、T、時、分、秒) です。例えば、2006-11-20T12:00:00 のようになります。

-completedBefore *timestamp*
(オプション) 指定の時刻より前に完了したプロセス・インスタンスの監視者データを削除します。*timestamp* ストリングの形式は、「yyyy-MM-ddThh:mm:ss」(年、月、日、T、時、分、秒) です。例えば、2006-07-20T12:00:00 のようになります。

-force -state *state* **-reachedBefore** *timestamp*
(オプション) 指定の時刻より前に指定の状態になったプロセス・インスタンスの監視者データを強制的に削除します。*timestamp* ストリングの形式は、「yyyy-MM-ddThh:mm:ss」(年、月、日、T、時、分、秒) です。例えば、2006-07-20T12:00:00 のようになります。

profileName *profileName*
ユーザー定義プロファイルの名前。デフォルト・プロファイルで作業していない場合、このオプションを指定します。

例えば、サーバー *myServer* のノード *myNode* で実行されるプロセス・インスタンスのうち、2008 年 5 月 16 日の正午より前に終了状態になったものをすべて削除するには、次のコマンドを実行します。

```
wsadmin –lang jython –f observerDeleteCompletedProcessInstances.py
        -node myNode -server myServer
        -force -state finished -reachedBefore 2008-05-16T12:00:00
```

正常に実行されれば、ツールから、監視者データが削除されたインスタンスの数と、データベースから削除された表項目の数が報告されます。そうでない場合は、エラー情報が報告され、データベースは変更されません。

指定したプロセス・インスタンスの監視者データが Observer データベースから削除されました。

管理コマンドを使用した、失敗したメッセージの照会と再生

管理コマンドを使用して、ビジネス・プロセスまたはヒューマン・タスクの失敗したメッセージが存在するかどうかを判別し、失敗したメッセージが存在する場合は、そのメッセージの処理を再試行します。

この手順を始める前に、次の条件が満たされている必要があります。

- メッセージが照会または再生されるアプリケーション・サーバーが稼働している必要があります。つまり、サーバー接続が必要であるため、wsadmin スクリプトの -conntype none オプションは使用できません。
- セキュリティーが使用可能に設定されている場合は、オペレーター権限を持っている必要があります。

内部メッセージの処理中に問題が発生すると、このメッセージが最終的に保存キューまたは保留キューに入れられます。失敗したメッセージが存在するかどうかを判別する方法、およびそれらのメッセージを内部キューへ再び送信する方法は、以下のとおりです。

1. 管理スクリプトが置かれている Business Process Choreographer サブディレクトリーに移動します。

   以下のコマンドを入力します。

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. 保存キューと保留キューの両方で、失敗したメッセージの数を照会します。 以下のいずれかのコマンドを入力します。コマンド間の相違を強調して表示しています。

   ```
   install_root/bin/wsadmin –lang jython -f queryNumberOfFailedMessages.py
                            -cluster clusterName
                            [ -bfm | -htm ]
                            [-profile profileName]

   install_root/bin/wsadmin –lang jython -f queryNumberOfFailedMessages.py
                            -node nodeName
                            -server serverName
                            [ -bfm | -htm ]
                            [-profile profileName]
   ```

   各部の意味は、次のとおりです。

   cluster *clusterName*
   クラスターの名前。ビジネス・プロセス・コンテナーが WebSphere クラスター用に構成されている場合は必須です。

   node *nodeName*
   サーバー名を指定する場合のオプション。この名前は、ノードを示します。デフォルトはローカル・ノードです。

   server *serverName*
   サーバーの名前。クラスター名が指定されていない場合は必須です。

   bfm|htm
   これらのキーワードはオプションです。デフォルトでは、どちらのオプションも指定されていなければ、ビジネス・プロセスとヒューマン・タスク両方の失敗したメッセージすべてが表示されます。ビジネス・プロセス・コンテナーの保留キューと保存キューにあるメッセージ数のみを表示する場合は、-bfm オプションを指定します。ヒューマン・タスク・コンテナーの保留キューにあるメッセージ数のみを表示する場合は、-htm オプションを指定します。

   profile *profileName*
   ユーザー定義プロファイルの名前。デフォルト・プロファイルで作業していない場合、このオプションを指定します。

3. 保留キュー、保存キュー、または両方のキューにある失敗したメッセージをすべて再生します。 以下のいずれかのコマンドを入力します。

   ```
   install_root/bin/wsadmin –lang jython -f replayFailedMessages.py
                            -cluster clusterName
                            -queue replayQueue
                            profile profileName
   install_root/bin/wsadmin –lang jython -f replayFailedMessages.py
                            -node nodeName
                            -server serverName
                            -queue replayQueue
                            profile profileName
   install_root/bin/wsadmin –lang jython -f replayFailedMessages.py
   ```

```
-server serverName
-queue replayQueue
profile profileName
```

各部の意味は、次のとおりです。

queue *replayQueue*
以下のいずれか 1 つの値を持っている必要があります。

`holdQueue`

`retentionQueue`

両方

cluster *clusterName*
クラスターの名前。ビジネス・プロセス・コンテナーが WebSphere クラスター用に構成されている場合は必須です。

node *nodeName*
サーバー名を指定する場合のオプション。この名前は、ノードを示します。デフォルトはローカル・ノードです。

server *serverName*
サーバーの名前。クラスター名が指定されていない場合は必須です。

bfm|htm
これらのキーワードはオプションであり、相互に排他的です。デフォルトでは、どちらのオプションも指定されていなければ、ビジネス・プロセスとヒューマン・タスク両方の失敗したメッセージが再生されます。ビジネス・プロセスのメッセージのみを再生する場合は、-bfm オプションを指定します。ヒューマン・タスクのメッセージのみを再生する場合は、-htm を指定します。

profile *profileName*
ユーザー定義プロファイルの名前。デフォルト・プロファイルで作業していない場合、このオプションを指定します。

注: 未使用スタッフ照会のクリーンアップのスクリプトの jacl バージョン、`replayFailedMessages.jacl` は推奨されません。これは、`ProcessChoreographer` ディレクトリーの `util` サブディレクトリーでは使用可能で、ここで説明しているのと同じパラメーターを使用しますが、`–lang jython` オプションは省略する必要があります。

管理コマンドを使用したスタッフ照会結果の最新表示

スタッフ照会の結果は静的です。管理コマンドを使用して、スタッフ照会を最新表示します。

この手順を始める前に、次の条件が満たされている必要があります。

- メッセージが照会または再生されるアプリケーション・サーバーが稼働している必要があります。つまり、サーバー接続が必要であるため、wsadmin の `-conntype none` オプションは使用できません。
- セキュリティーが使用可能に設定されている場合は、オペレーター権限を持っている必要があります。

Business Process Choreographer は、ランタイム・データベースで、Lightweight Directory Access Protocol (LDAP) サーバーなど、スタッフ・ディレクトリーに対して評価されたスタッフ割り当ての結果をキャッシュに入れます。スタッフ・ディレクトリーを変更する場合は、スタッフ割り当てを再評価するよう強制できます。

1. 管理スクリプトが置かれている Business Process Choreographer サブディレクトリーに移動します。

   以下のコマンドを入力します。

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. スタッフ割り当てを再評価するよう強制します。

   以下のいずれかのコマンドを入力します。コマンド間の相違を強調して表示しています。

   ```
   install_root/bin/wsadmin –lang jython -f refreshStaffQuery.py
           -server serverName
          [-processTemplate templateName |
          (-taskTemplate templateName [-nameSpace nameSpace]) |
            -userList username{,username}...]
           [-profile profileName]

   install_root/bin/wsadmin –lang jython -f refreshStaffQuery.py
           -node nodeName
           -server serverName
          [-processTemplate templateName |
          (-taskTemplate templateName [-nameSpace nameSpace]) |
           -userList username{,username}...]
                           [-profile profileName]

   install_root/bin/wsadmin –lang jython -f refreshStaffQuery.py
           -cluster clusterName
          [-processTemplate templateName |
          (-taskTemplate templateName [-nameSpace nameSpace]) |
           -userList username{,username}...]
          [-profile profileName]
   ```

   各部の意味は、次のとおりです。

   cluster *clusterName*
   クラスターの名前。ビジネス・プロセス・コンテナーが WebSphere クラスター用に構成されている場合は必須です。

   node *nodeName*
   サーバー名を指定する場合のオプション。この名前は、ノードを示します。デフォルトはローカル・ノードです。

   server *serverName*
   サーバーの名前。クラスター名が指定されていない場合は必須です。

   processTemplate *templateName*
   プロセス・テンプレートの名前。このプロセス・テンプレートに属すスタッフ割り当てが更新されます。

   taskTemplate *templateName*
   タスク・テンプレートの名前。このタスク・テンプレートに属すスタッフ割り当てが更新されます。

   nameSpace *nameSpace*
   タスク・テンプレートのネーム・スペース。

userList *userName*
コンマで区切られたユーザー名のリスト。指定された名前を含むスタッフ割り当てが更新されます。

profileName *profileName*
ユーザー定義プロファイルの名前。デフォルト・プロファイルで作業していない場合、このオプションを指定します。

注: *templateName* も *userList* も指定しない場合は、データベースに保管されているすべてのスタッフ照会が更新されます。パフォーマンス上の理由により、この処理は避けた方がよいでしょう。

注: スタッフ照会の更新スクリプトの jacl バージョン `refreshStaffQuery.jacl` は、推奨されていません。これは、`ProcessChoreographer` ディレクトリーの `util` サブディレクトリーでは使用可能で、ここで説明しているのと同じパラメーターを使用しますが、`-lang jython` オプションは省略する必要があります。

更新デーモンを使用したスタッフ照会結果の更新

期限切れのすべてのスタッフ照会結果を定期的に自動で更新するようにセットアップしたい場合は、このメソッドを使用します。

スタッフ照会は、指定されたスタッフ・プラグイン・プロバイダー・リポジトリーによって解決されます。結果は Business Process Choreographer データベースに保管されます。許可のパフォーマンスを最適化するため、取得された照会結果はキャッシュされます。キャッシュの内容の現行性は、スタッフ照会更新デーモンを呼び出したときにチェックされます。

スタッフ照会結果を最新に保つために、期限切れのスタッフ照会結果を定期的に更新するデーモンが提供されます。このデーモンは、キャッシュしたスタッフ照会結果のうち有効期限が切れたものをすべて更新します。

1. ヒューマン・タスク・コンテナーのカスタム・プロパティー・ページに移動するには、次のようにします。

   「**サーバー**」 → 「**アプリケーション・サーバー**」 → ***Server_Name*** をクリックし、「**コンテナー設定**」セクションの「**構成**」タブで、「**ヒューマン・タスク・コンテナーの設定**」 → 「**ランタイム構成**」をクリックします。
2. 「**スタッフ照会リフレッシュ・スケジュール**」フィールドに、WebSphere CRON カレンダーがサポートする構文でスケジュールを入力します。この値は、いつデーモンが有効期限が切れたスタッフ照会結果を更新するかを決定します。デフォルト値は「`0 0 1 * * ?`」で、これで毎日午前 1 時に更新が実行されます。
3. 「**スタッフ照会結果のタイムアウト**」フィールドに、新しい値 (秒) を入力します。この値は、スタッフ照会結果が有効であるとみなされる期間を決定します。この期間が経過するとスタッフ照会結果は無効になったとみなされ、デーモンが次に実行されるときに、このスタッフ照会は更新されます。デフォルトは 1 時間です。
4. 「**OK**」をクリックします。
5. 変更を保管し、ヒューマン・タスク・コンテナー・アプリケーションを再始動して変更を有効にします。

新規の有効期限の値は、新規のスタッフ照会にのみ適用され、既存のスタッフ照会には適用されません。

管理コマンドを使用した、未使用のスタッフ照会結果の除去

管理コマンドを使用して、データベースから未使用のスタッフ照会結果を削除します。

この手順を始める前に、次の条件が満たされている必要があります。

- 未使用のスタッフ照会の削除に使用するアプリケーション・サーバーが稼働している必要があります。つまり、サーバー接続が必要であるため、wsadmin の -conntype none オプションは使用できません。
- セキュリティーが使用可能に設定されている場合は、オペレーター権限を持っている必要があります。

Business Process Choreographer は、評価されたスタッフ式のランタイム・データベースでユーザー名の名前を維持します。このスタッフ式を使用したプロセス・インスタンスおよびヒューマン・タスクは完了していますが、ユーザー名のリストは、対応するビジネス・プロセス・アプリケーションがアンインストールされるまでデータベース内に残ります。

データベースのサイズがパフォーマンスに影響する場合は、データベース表にキャッシュされた未使用のスタッフ・リストを削除できます。

1. 管理スクリプトが置かれている Business Process Choreographer サブディレクトリーに移動します。

   以下のコマンドを入力します。

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. 未使用のスタッフ・リストを除去します。

   以下のいずれかのコマンドを入力します。コマンド間の相違を強調して表示しています。

   ```
   install_root/bin/wsadmin –lang jython -f cleanupUnusedStaffQueryInstances.py
                            -server serverName
                            [-profile profileName]

   install_root/bin/wsadmin –lang jython -f cleanupUnusedStaffQueryInstances.py
                            -node nodeName
                            -server serverName
                            [-profile profileName]

   install_root/bin/wsadmin –lang jython -f cleanupUnusedStaffQueryInstances.py
                            -cluster clusterName
                            [-profile profileName]
   ```

   各部の意味は、次のとおりです。

   cluster *clusterName*
   クラスターの名前。ビジネス・プロセス・コンテナーが WebSphere クラスター用に構成されている場合は必須です。

   node *nodeName*
   サーバー名を指定する場合のオプション。この名前は、ノードを示します。デフォルトはローカル・ノードです。

server *serverName*
サーバーの名前。クラスター名が指定されていない場合は必須です。

profileName *profileName*
ユーザー定義プロファイルの名前。デフォルト・プロファイルで作業していない場合、このオプションを指定します。

データベースから削除されるエントリーの数が表示されます。

注: 未使用スタッフ照会のクリーンアップのスクリプトの jacl バージョン `cleanupUnusedStaffQueryInstances.jacl` は、推奨されていません。これは、`ProcessChoreographer` ディレクトリーの `util` サブディレクトリーでは使用可能で、ここで説明しているのと同じパラメーターを使用しますが、`–lang jython` オプションは省略する必要があります。

ビジネス・プロセスおよびヒューマン・タスクの管理

ビジネス・プロセスおよびヒューマン・タスクは、エンタープライズ・アプリケーションの一部として配置およびインストールされます。管理コンソールまたは管理コマンドを使用してプロセス・テンプレートおよびタスク・テンプレートを管理し、Business Process Choreographer Explorer を使用してプロセス・インスタンスおよびタスク・インスタンスを操作できます。 Business Process Choreographer Observer を使用して、ビジネス・プロセスおよびヒューマン・タスクについて報告します。

ビジネス・プロセスについて

ビジネス・プロセスは、ビジネス・ゴールを達成するために特定のシーケンスで呼び出される、ビジネス関連の一連のアクティビティーです。

WS-BPEL (Web Services Business Process Execution Language) で定義されたプロセスには、以下の構成要素があります。

- プロセス内の個々のビジネス・ステップであるアクティビティー。アクティビティーは、いくつかの異なるタイプのうちの 1 つを持つことができます。また、アクティビティーは、基本アクティビティーまたは構造化アクティビティーのいずれかに分類することができます。
  - 基本アクティビティーとは、構造を持たない、他のアクティビティーを含まないアクティビティーです。
  - 構造化アクティビティーとは、他のアクティビティーを含むアクティビティーです。
- WSDL インターフェースを使用してプロセスと対話する (またはその逆も可能) 外部エンティティーおよびパートナーを指定するパートナー・リンク。インターフェース・パートナーまたはリファレンス・パートナーとも呼ばれます。
- アクティビティー間で渡されるメッセージを格納する変数。この変数は、ビジネス・プロセス・インスタンスの状態を表します。
- 複数のサービス要求または応答メッセージを同じビジネス・プロセス・インスタンスに相互に関連付けるための相関セット。相関セットは、プロセスによって交換されるメッセージに含まれているアプリケーション・データに基づいています。

- ビジネス・プロセスの実行時に発生する可能性のある例外的な状態に対処する障害ハンドラー。
- 非送信請求メッセージの受信と処理を通常の実行処理と並行して行うイベント・ハンドラー。
- 単一のアクティビティーまたはアクティビティー・グループの補正ロジックを指定する補正ハンドラー。

これらの構成要素について詳しくは、『BPEL specification』を参照してください。

Business Process Choreographer では、以下に示すような BPEL 言語の IBM® 拡張もサポートされます。

- 人間との対話のためのヒューマン・タスク・アクティビティー。フォームの入力、文書や描画の承認、レターの作成などの、人が関わるほぼすべてのビジネス・プロセスのステップがインラインの参加タスクとなります。
- インライン Java コードを実行するためのスクリプト・アクティビティー。Java コードは、プロセス・コンテキストおよびアクティビティー・コンテキストだけでなく、すべての BPEL 変数、相関プロパティー、パートナー・リンクにアクセスすることができます。
- WebSphere Information Server およびリレーショナル・データベースに直接アクセスするための情報サービス・アクティビティー。
- プロセス・モデルのバージョン管理のための有効開始タイム・スタンプ。
- Common Event Infrastructure (CEI) のロギング。
- 1 つのトランザクションで複数のアクティビティーをサポートするための明示的チェックポイント機能。
- アクティビティーのタイムアウト。

ビジネス・プロセスのタイプ

ビジネス・プロセスは、長期実行または Microflow のいずれかになります。

長期実行プロセス

長期実行ビジネス・プロセスは割り込み可能です。このプロセスの各ステップは、それぞれ固有の物理トランザクションで実行できます。長期実行ビジネス・プロセスは、外部刺激を待機できます。外部刺激の例として、企業間の対話の中で別のビジネス・プロセスによって送信されるイベント、非同期の起動に対する応答、またはヒューマン・タスクの完了などがあります。

長期実行のビジネス・プロセスには、次の特性があります。

- いくつかのトランザクションとして実行します。
- 同期サービスと非同期サービスから成ります。
- それぞれの中間プロセス状態を格納します。これにより、プロセスの順方向リカバリーが可能になります。

Microflow

Microflow は、最初から最後まで割り込みなしに 1 つの物理スレッドで実行されます。Microflow は、中断不可能なビジネス・プロセスと呼ばれる場合もあります。Microflow は、それぞれ異なるトランザクション機能を持つことができます。

Microflow は、グローバル・トランザクション内で実行したり、アクティビティー・セッションの一部として実行したりできます。

Microflow には、次の特性があります。
- 1 つのトランザクションの中で実行します。
- 通常は短時間で実行します。
- ランタイム値をデータベースに保管しません。
- 同期サービスおよび中断不可能なサブプロセスのみで構成されます。つまり、Microflow は以下のものを含むことができません。
  - ヒューマン・タスク
  - wait アクティビティー
  - 複数の開始 receive アクティビティー
  - 開始していない receive アクティビティー

  microflow では、以下のサービスまたはアクティビティーを呼び出すことは推奨できません。
  - 長期実行サービス
  - 非同期プロトコルに結合されているアクティビティー。

追加の BPEL アクティビティー

Business Process Choreographer には、Web Services Business Process Execution Language (BPEL) invoke アクティビティーへの拡張である追加のアクティビティーのサポートが含まれています。

これらの追加アクティビティーには、 Java 断片アクティビティーおよび情報サービス・アクティビティーが含まれます。

Java 断片アクティビティー

Java 断片アクティビティー (BPEL invoke アクティビティーのスクリプト拡張) により、Java コードをプロセス・インプリメンテーションの一部として指定できます。この Java コードは、エンクロージング BPEL 環境にアクセスできます。例えば、BPEL 変数、パートナー・リンク、相関セット、およびカスタム・プロパティーを処理できます。これらのオブジェクトは、単純なタイプを表すデータ・オブジェクト、または Java オブジェクトのいずれかです。包含する Java メソッド内でローカル Java 変数を使用できるのと同じように、 Java 断片内で BPEL を使用できます。

情報サービス・アクティビティー

情報サービス・アクティビティーを使用すると、 IBM Information Server およびリレーショナル・データベースと直接対話することができます。以下の種類の情報サービス・アクティビティーが使用可能です。

情報サーバー
この種類のアクティビティーでは、Information Server で作成された情報サービスをビジネス・プロセスから呼び出すことができます。

SQL 断片
SQL 断片により、SQL ステートメント (データ定義言語 (DDL) ステートメントを含む) をビジネス・プロセスから処理できます。例えば、SQL

select ステートメントは照会を発行し、照会結果を参照 (設定参照) 別にプロセス変数に割り当てることができます。プロセス内のその他のアクティビティーは、すべての関連データをプロセス・スペースに移動したりする必要もなく、これらの設定参照を使用できます。

取得セット
取得セットにより、設定参照で定義されたデータをプロセス変数に読み込むことができます。データはビジネス・オブジェクトとして戻されます。

アトミック SQL シーケンス
アトミック SQL シーケンスにより、複数の SQL 断片および取得セット・ステートメントを情報サーバー・アクティビティーで定義できます。ステートメントは、定義された順序で 1 回のトランザクションで処理されます。

サブプロセスのライフ・サイクル管理とバージョン管理の振る舞い

別のプロセスによって開始されたプロセスをサブプロセスといいます。サブプロセスのライフ・サイクルの管理が可能な方法と、サブプロセスのバージョン管理の振る舞いは、プロセスのモデル化方法によって異なります。

モジュール性と再利用性を高めるため、多くの場合、カプセル化のプログラミング概念をビジネス・プロセス・モデリングに適用することには意味があります。すなわち、ビジネス・ロジックの 1 つ以上のステップを個別のプロセスとしてインプリメントし、このプロセスをメイン・プロセスから呼び出すのです。サブプロセスもまた、別のプロセスを開始することができます。これにより、不特定な深さのプロセス・インスタンスの階層が生じます。これらのプロセスをデプロイする場合は、プロセス間リレーションシップのプロセス・テンプレートのすべてを、同じ Business Process Choreographer データベースにデプロイする必要があります。

ライフ・サイクル管理

サブプロセスは、呼び出しプロセスと、対等リレーションシップか親子リレーションシップにあります。このリレーションシップにより、呼び出しプロセスのプロセス・ライフ・サイクルを管理するアクションが呼び出されたときのサブプロセスの振る舞いが決まります。ライフ・サイクルのアクションは、中断、再開、終了、削除、および補正で構成されます。プロセスのライフ・サイクルを管理するアクションは、トップレベルのプロセス・インスタンスでのみ実行可能です。

呼び出しプロセス対サブプロセスのリレーションシップは、サブプロセスの autonomy 属性によって判別されます。この属性の値は、次の値のいずれかになります。

peer 対等プロセスは、*トップレベル・プロセス*と見なされます。トップレベル・プロセスは、別のプロセス・インスタンスによって呼び出されていないか、別のプロセス・インスタンスによって呼び出されたプロセス・インスタンスですが、autonomy 属性の値は peer です。サブプロセスが対等リレーションシップの一部になっている場合、呼び出しプロセス・インスタンスのライフ・サイクル・アクションは、サブプロセス・インスタンスに適用されません。

ただし、片方向インターフェースをインプリメントする作成オペレーションを伴う長期実行プロセスの場合は、実行時に autonomy 属性の値が自動的に peer に設定されます。autonomy 属性を child に設定すると、この値は実行時に無視されます。

child サブプロセスが親子リレーションシップの一部になっている場合、親プロセス・インスタンスのライフ・サイクル・アクションは、サブプロセス・インスタンスに適用されます。例えば、親プロセス・インスタンスが中断されると、autonomy 属性が child のサブプロセス・インスタンスもすべて中断されます。

Microflow はいつでも子プロセスとして実行されます。ただし、2 つのプロセスの間に別のコンポーネントがある場合、例えば、2 つのプロセス・コンポーネントの間を接続しているインターフェース・マップ・コンポーネントなどは、親子関係の確立を妨げることがあります。

バージョン管理の振る舞い

使用されるプロセスのバージョンは、*早期バインディング*のシナリオと*実行時バインディング*のシナリオのどちらでプロセスが使用されるかによって決まります。

早期バインディング

早期バインディングのシナリオでは、呼び出されるサブプロセスのバージョンはデプロイメント中に決定されます。呼び出しプロセスは、Service Component Architecture (SCA) ワイヤリングに従って、静的にバインドされた専用のサブプロセスを呼び出します。プロセスのバージョン管理は無視されます。

早期バインディングの例は SCA ワイヤーです。例えば、スタンドアロン参照をプロセス・コンポーネントに結びつけると、この参照を使用するプロセスの呼び出しはすべて、プロセス・コンポーネントによって表される特定のバージョンを対象とします。

実行時バインディング

実行時バインディングのシナリオでは、呼び出されるサブプロセス・テンプレートの決定は、呼び出しプロセス・インスタンスでサブプロセスを呼び出す必要が生じたときに行われます。この場合は、現行で有効なサブプロセスのバージョンが使用されます。プロセスの新しいバージョンが、そのプロセスの以前のバージョンのどれよりも優先されます。既存のプロセス・インスタンスは、開始時に関連付けられたプロセス・テンプレートを使って、継続して実行されます。これにより、プロセス・テンプレートは次のカテゴリーに分けられます。

- 既に最新ではなくても、既存の長期実行プロセス・インスタンスにとっては有効なプロセス・テンプレート
- 新規プロセス・インスタンスで使用される現行のプロセス・テンプレート
- 有効開始日時に従って将来的に有効になるプロセス・テンプレート

サブプロセスの呼び出し時に実行時バインディングを適用するため、親プロセスは、参照パートナーで有効なサブプロセスが選択されるときの選択元となるサブプロセス・テンプレートの名前を指定する必要があります。プロセスの有効開始属性を使用して、現行で有効なサブプロセス・テンプレートが判別されます。 SCA ワイヤリングはどれも無視されます。

実行時バインディングの例は、Business Process Choreographer Explorer で新規プロセスが呼び出される場合です。作成されるインスタンスはいつでも、有効開始日付が将来ではないプロセス・テンプレートの最新バージョンに基づいています。

新規バージョンのプロセス・モデルが作成されて、既存のプロセス・モデルが実行時バインディング・シナリオで使用されるとき、変更を行うことは避けてください。変更を行うと、新規バージョンのプロセスが有効になり、例えば、親プロセスが新規バージョンのサブプロセスのインスタンスを起動したときに、互換性の問題をもたらすことがあります。以下は、避ける必要のある非互換の変更です。

- 相関セットの変更
- 親プロセスがサブプロセスと通信するために使用するインターフェースの変更

ビジネス・プロセスとサービスの間のデータ交換

ビジネス・プロセスがサービス・コンポーネント・アーキテクチャー (SCA) サービスを消費したり、ビジネス・プロセスが他の SCA サービスによって消費されたりすることができます。 Web サービス記述言語 (WSDL) メッセージ・データが SCA サービスとプロセスの間で交換される方法は、プロセスがどのようにモデル化されたかによって異なります。

ビジネス・プロセスがサービスを消費する

ビジネス・プロセスでのサービスの消費は、プロセス・モデルの Business Process Execution Language (BPEL) invoke アクティビティーを使用してインプリメントされます。 SCA サービスに渡されるデータは、1 つ以上の BPEL 変数から検索されます。通常、データは値によって受け渡されます。これは、呼び出されたサービスがデータのコピーを処理することを意味します。

特定の環境では、データを参照によって受け渡すことができます。参照によるデータの受け渡しにより、ビジネス・プロセスのパフォーマンスを改善することができます。

以下の条件が**すべて**満たされる場合、データは参照によってビジネス・プロセスに受け渡されます。

- サービスの呼び出しが同期である。
- BPEL プロセスと呼び出されるサービスが同じモジュールにある。
- データが以下のいずれかの方法で交換される。
  - 1 つ以上の BPEL 変数が XML スキーマ・タイプまたはエレメントを使用して宣言される。 WSDL メッセージ部分は、サービスの呼び出しと変数の間で個々にマップされます。

    ```
    <variable name="inputPart1Var" type="ws:inputPart1Type">
    <variable name="inputPart2Var" type="ws:inputPart2Type">
    ```

    Web サービス・アクティビティーは、パラメーターの拡張を使用して、BPEL 変数を参照します。 SCA 対話では、WSDL は参照によって受け渡されるデータのラッパーとして扱われます。

    ```
    <invoke ....>
      <wpc:input>
        <wpc:parameter name="ws:inputPart1" variable="inputPart1Var"/>
    ```

```
    <wpc:parameter name="ws:inputPart2" variable="inputPart2Var"/>
    ...
  </wpc:input>
</invoke ....>
```

– 1 つ以上の BPEL 変数が XML スキーマ・タイプまたはエレメントを使用して宣言される。 Web サービス対話は、文書リテラルのラップされたスタイルに準拠します。パラメーター・エレメントはラッパー文書と変数の間でマップされます。

```
<variable name="inputParm1Var" type="ws:inputParm1ElemType">
<variable name="inputParm2Var" type="ws:inputParm2ElemType">
```

Web サービス・アクティビティーは、パラメーターの拡張を使用して、BPEL 変数を参照します。これが、WebSphere Integration Developer で作成されるプロセスのデフォルトの振る舞いです。 SCA 対話では、ラッパーは参照によって受け渡されるパラメーターを保持します。

```
<invoke ....>
  <wpc:input>
    <wpc:parameter name="ws:inputParm1" variable="inputParm1Var"/>
    <wpc:parameter name="ws:inputParm2" variable="inputParm2Var"/>
    ...
  </wpc:input>
</invoke ....>
```

呼び出されたサービスがデータを変更する場合、これらの変更は対応する BPEL 変数に適用されます。ただし、最良実例として、呼び出されたサービスはデータを更新すべきではありません。その理由は、データに対して行われる変更は永続的ではないからです。長期間のプロセスの場合、現行のトランザクションがコミットすると変更は破棄され、microflow の場合、プロセスが終了すると変更が破棄されます。さらに、呼び出されたサービスによって変数が更新されると、イベントは生成されません。

ビジネス・プロセスがサービスによって消費される

他のサービスによって消費されるビジネス・プロセスでは、プロセス・モデル内に、receive アクティビティー、pick アクティビティー、またはイベント・ハンドラーが含まれています。プロセスに受け渡されるデータは 1 つ以上の BPEL 変数に書き込まれます。通常、データは値によって受け渡されます。これは、プロセスがデータのコピーを処理することを意味します。

ただし、以下の条件が**すべて**満たされる場合、データは参照によって受け渡されます。

- ビジネス・プロセスの起動が同期である。
- サービスと起動されるビジネス・プロセスが同じモジュールにある。
- データが以下のいずれかの方法で交換される。
  – 1 つ以上の BPEL 変数が XML スキーマ・タイプまたはエレメントを使用して宣言される。 WSDL メッセージ部分は、サービスの呼び出しと変数の間で個々にマップされます。

```
<variable name="outputPart1Var" type="ws:outputPart1Type">
<variable name="outputPart2Var" type="ws:outputPart2Type">
```

アクティビティーは、パラメーターの拡張を使用して、BPEL 変数を参照します。 SCA 対話では、WSDL は参照によって受け渡されるデータのラッパーとして扱われます。 receive アクティビティーの場合、対応する BPEL 断片は、以下の例のようになります。

```
<receive ....>
  <wpc:output>
    <wpc:parameter name="ws:outputPart1" variable="outputPart1Var"/>
    <wpc:parameter name="ws:outputPart2" variable="outputPart2Var"/>
    ...
  </wpc:output>
</receive ....>
```

– 1 つ以上の BPEL 変数が XML スキーマ・タイプまたはエレメントを使用して宣言される。 Web サービス対話は、文書リテラルのラップされたスタイルに準拠します。パラメーター・エレメントはラッパー文書と変数の間でマップされます。

```
<variable name="outputParm1Var" type="ws:outputParm1ElemType">
<variable name="outputParm2Var" type="ws:outputParm2ElemType">
```

アクティビティーは、パラメーターの拡張を使用して、BPEL 変数を参照します。これが、WebSphere Integration Developer で作成されるプロセスのデフォルトの振る舞いです。 SCA 対話では、ラッパーは参照によって受け渡されるパラメーターを保持します。 receive アクティビティーの場合、対応する BPEL 断片は、以下の例のようになります。

```
<receive ....>
  <wpc:output>
    <wpc:parameter name="ws:outputParm1" variable="outputParm1Var"/>
    <wpc:parameter name="ws:outputParm2" variable="outputParm2Var"/>
    ...
  </wpc:output>
</receive ....>
```

起動されたプロセスが BPEL 変数を変更する場合、呼び出しサービスからの入力データも変更されます。

ヒューマン・タスクについて

ヒューマン・タスクとは、サービスや他の人間と対話する人間に関与するコンポーネントです。

対話は、人または自動化サービスのいずれかによって開始することができます。人によって開始されるサービスは、自動化インプリメンテーション、または別の人によって提供されるサービスのいずれかです。自動化サービスによって起動されるヒューマン・タスクは、簡単に自動化インプリメンテーションと置き換えることができます。また、その逆も可能です。

タスクを使用して、例外の手動処理や承認などの、人間による対話が必要なビジネス・プロセスのスタッフ・アクティビティーをインプリメントすることができます。それ以外の例外処理では、障害や障害ハンドラー、または補正を使用して、Web Services Business Process Execution Language (WS-BPEL、略称 BPEL) によるネイティブのモデル化が行われます。

誰がタスクと対話できるかは、サポートされるいずれかのスタッフ・ディレクトリーを使用して決定することができます。作業項目は、タスクを表示するかタスクと対話する理由のあるユーザーのために作成されます。

Business Process Choreographer は、以下のタイプのスタッフ・ディレクトリーをサポートします。

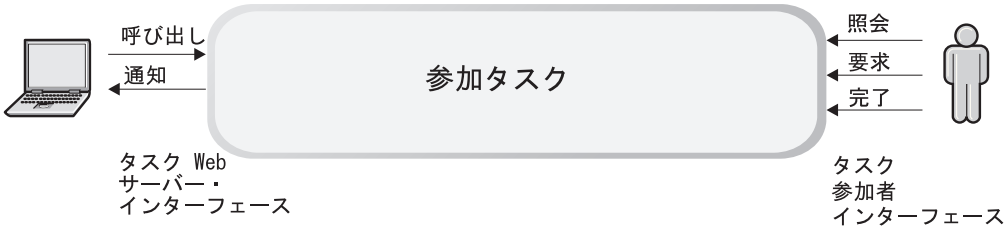
- Lightweight Directory Access Protocol (LDAP)
- WebSphere ユーザー・レジストリー

ヒューマン・タスクの種類

ヒューマン・タスクの種類には以下のものがあります。

参加タスク

Web サービスと人との対話をサポートします。これにより、人がサービスをインプリメントできます。例えば、ビジネス・プロセス内のヒューマン・タスク・アクティビティーを参加タスクとすることができます。



管理用タスク

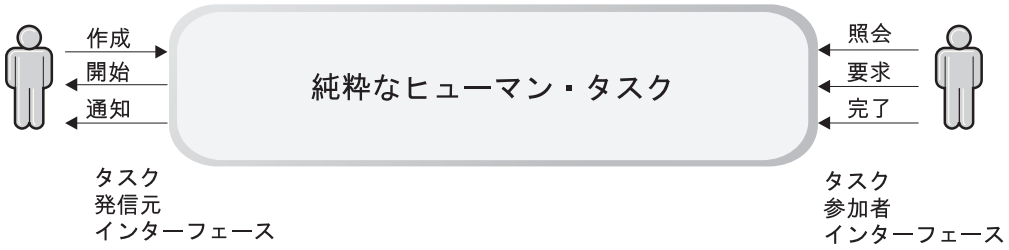
管理用タスクは参加タスクと似ていますが、管理用タスクはさまざまな処理中に発生する技術的な問題を解決するために管理者が使用する点が異なります。現在、ビジネス・プロセスにのみ管理タスクを使用できます。

親タスク

人とコンピューターとの対話をサポートします。これにより、人がグラフィカル・ユーザー・インターフェースを使用してサービスを作成し、始動することができます。例えば、ユーザーがビジネス・プロセスを開始し、そのプロセスに親タスクを使用してイベントを送信することができます。

純粋なヒューマン・タスク

人と人との対話をサポートします。これにより、構造化され制御された方法で、個人が他の個人と作業を共用できます。純粋なヒューマン・タスクはビジネス・プロセスまたは他の Web サービスとは対話しません。



ヒューマン・タスクとビジネス・プロセスの関係

ヒューマン・タスクは、以下のいずれかの方法でビジネス・プロセスと関連付けることができます。

インライン・タスク

インライン・タスクは、ビジネス・プロセスの一部として定義されます。これは Service Component Architecture (SCA) コンポーネントとして表示されませんが、コンテキスト・データをプロセスと共用できます。

スタンドアロン・タスク

スタンドアロン・タスクは、人との対話をサービスとしてインプリメントしたり (参加タスク)、グラフィカル・ユーザー・インターフェースによる人とサービスとの対話を活用したり (親タスク)、人と人との構造化されたコラボレーションをサポートしたりする (純粋なヒューマン・タスク)、SCA コンポーネントです。タスク・コンポーネントは、ビジネス・プロセスなどのその他のサービスと結合できます。

以下の表は、これらの 2 つのインプリメンテーション・タイプの違いを示しています。

インライン・タスク	スタンドアロン・タスク
ビジネス・プロセスの一部。	ビジネス・プロセスから独立。このインプリメンテーションは、ビジネス・プロセスを含まないシナリオでも使用できます。
タスクのライフ・サイクルは、通常、プロセスによって制御されます。	ライフ・サイクルはプロセスから独立しています。
参加タスクは、プロセス内のヒューマン・タスク・アクティビティーです。	参加タスクは、プロセス内の invoke アクティビティーです。
インライン・タスクは、プロセス・コンテキスト・データ (例えば、変数、スタッフ割り当て、またはカスタム・プロパティー) にアクセスできます。	スタンドアロン・タスクはプロセス・コンテキスト・データにアクセスできません。
参加タスクおよび親タスクのタスク記述、表示名、および資料は 1 つの言語のみサポートします。	参加タスクおよび親タスクのタスク記述、表示名、および資料は複数の言語をサポートします。
インライン・タスクは SCA コンポーネントとして表示されないため、再利用できません (関連付けできません)。	スタンドアロン・タスクは再利用可能です。参加タスクおよび親タスクは SCA コンポーネントとして表示されます (関連付けできます)。

インライン・タスク	スタンドアロン・タスク
サポートされるタスクの種類: 参加タスク、親タスク、および管理タスク。	サポートされるタスクの種類: 参加タスク、親タスク、および純粋なヒューマン・タスク。

サブタスク

サブタスクは、親タスクから分割される追加の作業単位です。サブタスク・モデルは、テンプレートから選択するか、実行時に定義することができます。入力データは、サブタスクを作成または開始する個人によって提供されます。親タスクは、すべてのサブタスクが終了するまで待機します。親タスクの所有者または編集者は、サブタスク出力データを統合してから、親タスクを完了します。

サブタスクが指定された時間内に完了できない場合、親タスクをエスカレーションできます。エスカレーションは、親タスクがまだサブタスクの完了を待っていることを示します。

サブタスクの対象は、純粋なヒューマン・タスクまたは親タスクです。

後続のタスク

後続のタスクは、既存のタスクを完了するために作成されるタスクです。後続のタスク・モデルは、テンプレートから選択するか、実行時に定義することができます。入力データは、後続のタスクを作成または開始する個人によって提供されます。後続のタスクの出力および障害メッセージ・タイプは、前のタスクのタイプと同じでなければなりません。前のタスクは処理進行状態になり、それを呼び出したサービスまたは個人に対して完了を報告しません。

後続のタスクが終了すると、その出力または障害データを、元のタスクを呼び出したサービスまたは個人に報告します。前のタスクのエスカレーションは、実行およびエスカレーションを続行します。後続のタスクには、独自のエスカレーションがあります。

後続のタスクの対象は、純粋なヒューマン・タスクのみです。

エスカレーション

エスカレーションとは、指定した期間内にタスクが正常に完了しない場合に実行されるアクションの過程のことです。例えば、タスクが要求されない場合、または定義した制限時間内に完了しない場合を考えてみましょう。この場合、タスク用に 1 つまたは複数のエスカレーションを指定できます。これらのエスカレーションを並行して開始したり、一連のエスカレーションとして開始することができます。

エスカレーションは、関連タスクがそのライフ・サイクルの一定の状態に到達すると初期化されます。タスクの状態は定義期間後に検査され、その状態がモデル化された期待値を満足しない場合は、エスカレーション・アクションが起動されます。以下のエスカレーション・アクションがサポートされます。

- ユーザーのセット用に作業項目を作成する
- 指定された受信者に E メールを送信する
- 登録済みコンシューマーに通知イベントを送信する

プロセス・テンプレートおよびプロセス・インスタンスの管理

管理コンソールおよび管理コマンドを使用して、プロセス・テンプレートを管理します。プロセス・インスタンスの処理には Business Process Choreographer Explorer を使用します。

プロセス・テンプレートは、エンタープライズ・アプリケーション内でビジネス・プロセスを定義します。プロセス・テンプレートを含むエンタープライズ・アプリケーションがインストールされ、配置され、開始されると、プロセス・テンプレートが開始状態になります。管理コンソールまたは管理コマンドを使用して、プロセス・テンプレートの停止および開始を行います。開始されたプロセス・テンプレートが Business Process Choreographer Explorer に表示されます。

プロセス・インスタンスは、長期実行プロセスの場合と、Microflow の場合があります。Business Process Choreographer Explorer を使用すると、プロセス・テンプレートやプロセス・インスタンスに関する情報を表示したり、プロセス・インスタンスに対してアクションを実行したりできます。例えば、アクションにはプロセス・インスタンスの開始、および、長期実行プロセスの場合、プロセス・インスタンスの中断や再開、あるいは終了などその他のプロセス・ライフ・サイクル・アクション、アクティビティーの修復などがあります。

ビジネス・プロセス管理 - よくある質問

ビジネス・プロセスの管理に関するよくある質問への一連の回答。

- 『プロセス・テンプレートが開始済み状態で、それが属しているアプリケーションが停止状態である場合は、どうなりますか?』
- 『プロセス・インスタンスの作成を停止するにはどうすればよいでしょうか?』
- 178 ページの『より新しいプロセス・テンプレートが有効になった場合、実行中のインスタンスはどうなりますか?』
- 178 ページの『作成に使用されたテンプレートが停止されると、実行中のインスタンスはどうなりますか?』
- 178 ページの『プロセス・インスタンスがまだ実行中であることは、 どのようにして分かりますか?』
- 178 ページの『ビジネス・プロセス・アプリケーションにプロセス・インスタンスが ある場合、アプリケーションを停止できないのはなぜですか?』

プロセス・テンプレートが開始済み状態で、それが属しているアプリケーションが停止状態である場合は、どうなりますか?

現在有効なプロセス・テンプレートが開始済み状態で、アプリケーションが停止状態である場合、新規プロセス・インスタンスはテンプレートから作成されません。既存プロセス・インスタンスは、アプリケーションが停止状態の場合にはナビゲートできません。

プロセス・インスタンスの作成を停止するにはどうすればよいでしょうか?

管理コンソールを使用して、プロセス・テンプレートを選択し、「**停止**」をクリックします。このアクションは、プロセス・テンプレートを停止状態にし、テンプレートからはこれ以上のインスタンスは作成されません。テンプレートの停止後は、テンプレートからプロセス・インスタンスを作成するすべての試行は、

EngineProcessModelStoppedException エラーになります。

より新しいプロセス・テンプレートが有効になった場合、実行中のインスタンスはどうなりますか?

プロセス・テンプレートが無効な場合、この状態は、テンプレートからインスタンス化されたインスタンスの実行に影響を与えません。既存プロセス・インスタンスの実行は、完了するまで続行します。古いインスタンスと新規インスタンスは、古いインスタンスがすべて完了するか、強制終了されるまで、並行して実行します。

作成に使用されたテンプレートが停止されると、実行中のインスタンスはどうなりますか?

プロセス・テンプレートの状態を「stopped」に変更した場合、停止されるのは作成中の新規インスタンスだけです。既存のプロセス・インスタンスは、通常どおり、完了するまで実行を継続します。

プロセス・インスタンスがまだ実行中であることは、どのようにして分かりますか?

Business Process Choreographer Explorer にプロセス管理者としてログオンし、「自分で管理するプロセス・インスタンス」ページに移動します。このページには、実行中のプロセス・インスタンスが表示されます。必要な場合は、これらのプロセス・インスタンスを強制終了して削除することができます。

ビジネス・プロセス・アプリケーションにプロセス・インスタンスがある場合、アプリケーションを停止できないのはなぜですか?

プロセス・インスタンスを実行させるには、対応するアプリケーションも稼動している必要があります。アプリケーションが停止している場合、プロセス・インスタンスのナビゲーションは継続できません。そのため、ビジネス・プロセス・アプリケーションは、プロセス・インスタンスがない場合にしか停止できません。

ビジネス・プロセスのための許可のロール

ビジネス・プロセスで実行できるアクションは、ご使用の許可のロールに依存します。このロールは、J2EE のロールまたはインスタンス・ベースのロールとすることができます。

ロールとは、同じ権限レベルを共用する従業員の集合です。Java 2 Platform, Enterprise Edition (J2EE) のロールは、ビジネス・プロセス・コンテナーが構成されたときにセットアップされます。インスタンス・ベースのロールは、プロセスがモデル化されたときにプロセスおよびアクティビティーに割り当てられます。ロール・ベースの許可については、グローバル・セキュリティーが WebSphere Application Server で使用可能になっていることが必要です。

J2EE のロール

以下の J2EE のロールがサポートされています。

- J2EE BPESystemAdministrator。このロールを割り当てられたユーザーは、すべての特権を持ちます。また、このロールは、ビジネス・プロセスのシステム管理者とも言われます。

- J2EE BPESystemMonitor。このロールを割り当てられたユーザーは、すべてのビジネス・プロセス・オブジェクトのプロパティーを表示できます。また、このロールは、ビジネス・プロセスのシステム・モニターとも言われます。

これらのロールへのユーザーおよびグループの割り当てを変更するには、管理コンソールを使用できます。

RACF セキュリティーによるロールの設定: この RACF® 許可は、以下のセキュリティー・フィールドを指定した場合に適用されます。

- **com.ibm.security.SAF.authorization= true**

```
RDEFINE EJBROLE BPESystemAdministrator  UACC(NONE)
PERMIT BPESystemAdministrator CLASS(EJBROLE)  ID(userid) ACCESS(READ)

RDEFINE EJBROLE BPESystemMonitor  UACC(NONE)
PERMIT BPESystemMonitor  CLASS(EJBROLE)  ID(userid) ACCESS(READ)
```

- **com.ibm.security.SAF.delegation= true**

```
RDEFINE EJBROLE JMSAPIUser  UACC(NONE) APPLDATA(' userid')
```

Security Authorization Facility (SAF) ベースの許可 (RACF EJBROLE プロファイルの使用など) を使用して、ビジネス・プロセス・コンテナーなど、EJB およびエンタープライズ・アプリケーションにおける Java 2 Platform、 Enterprise Edition (J2EE) ロールに対するクライアントのアクセスを制御できます。 SAF の使用についての詳細は、 WebSphere Application Server for z/OS インフォメーション・センターの『役割ベースの許可の System Authorization Facility』を参照してください。

インスタンス・ベースのロール

プロセス・インスタンスまたはアクティビティーはプロセス・モデルではスタッフ・メンバーには直接割り当てられておらず、その代わり、使用可能なロールの 1 つに割り当てられます。インスタンス・ベースのロールに割り当てられたスタッフ・メンバーは、そのロールに応じたアクションを実行できます。インスタンス・ベースのロールとユーザーの関連付けは、スタッフの解決を使用して実行時に決定されます。

以下のインスタンス・ベースのロールがサポートされています。
- プロセスに対して: 読者、スターター、管理者
- アクティビティーに対して: 読者、編集者、可能なスターター、可能な所有者、所有者、管理者

これらのロールは、以下のアクションの実行が許可されています。

ロール	許可されたアクション
アクティビティー・リーダー	関連したアクティビティー・インスタンスのプロパティーおよび入出力メッセージを表示します。
アクティビティー編集者	アクティビティー・リーダーに許可されたアクションと、アクティビティーに関連したメッセージおよびその他のデータへの書き込みアクセス権限。
可能なアクティビティー・スターター	アクティビティー・リーダーに許可されたアクション。このロールのメンバーは、 receive アクティビティーまたは pick アクティビティーにメッセージを送信できます。

ロール	許可されたアクション
可能なアクティビティー所有者	アクティビティー・リーダーに許可されたアクション。このロールのメンバーはアクティビティーを要求できます。
アクティビティー所有者	アクティビティーを処理し、完了します。このロールのメンバーは、所有された作業項目を管理者または可能な所有者に転送できます。
アクティビティー管理者	予期しないエラーで停止したアクティビティーを修理し、長期実行アクティビティーを強制終了します。
プロセス・スターター	関連したプロセス・インスタンスのプロパティーおよび入出力メッセージを表示します。
プロセス・リーダー	関連したプロセス・インスタンスのプロパティーおよび入出力メッセージを表示します。また、プロセス・リーダーは、プロセス・インスタンスに含まれるアクティビティーのプロパティー、および入出力メッセージを表示できますが、そのサブプロセスに関する情報は表示できません。
プロセス管理者	このロールのメンバーは、プロセス・インスタンスを管理し、開始済みプロセスに介入して、作業項目の作成、削除、および転送ができます。このロールのメンバーには、アクティビティー管理者権限もあります。

プロセス・スターターのユーザー ID は、プロセス・インスタンスが存在している場合、ユーザー・レジストリーから削除しないでください。これを行うと、このプロセスのナビゲーションが継続できません。システム・ログ・ファイルから以下の例外を受け取ります。

```
no unique ID for: <user ID>
```

管理コンソールによるプロセス・テンプレートの停止および開始

管理コンソールを使用して、それぞれのプロセス・テンプレートを個々に開始および停止することができます。

グローバル・セキュリティーが使用可能に設定されている場合は、使用するユーザー ID にオペレーター権限があることを確認します。アプリケーションがインストールされているサーバーが稼働している必要があります。

例えば、プロセス・テンプレートが属しているビジネス・プロセス・アプリケーションをアンインストールするには、まずそのビジネス・テンプレートを停止する必要があります。以下のステップでは、管理コンソールを使用して、プロセス・テンプレートを停止および開始する方法について説明します。

1. 管理するアプリケーションを選択します。

   管理コンソールのナビゲーション・ペインで、「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」をクリックしてから、管理するアプリケーションをクリックします。
2. 「**関連項目**」の下のエンタープライズ・アプリケーションの「構成」ページで、「**EJB モジュール**」をクリックしてから、Enterprise JavaBeans モジュールをクリックします。

3. 「**追加プロパティー**」の下の EJB モジュールの「構成」ページで、「**ビジネス・プロセス**」をクリックしてからプロセス・テンプレートをクリックします。
4. プロセス・テンプレートを停止します。

   プロセス・テンプレートの既存のインスタンスは、正常に終了するまで稼働を続けます。ただし、停止したテンプレートからプロセス・インスタンスを作成することはできません。
5. 停止状態のプロセス・テンプレートを開始します。

管理コマンドによるプロセス・テンプレートの停止および開始

管理コマンドは、プロセス・テンプレートを停止および開始するため管理コンソールの代わりを提供します。

グローバル・セキュリティーが使用可能に設定されている場合は、使用するユーザー ID にオペレーター権限があることを確認します。

例えば、ビジネス・テンプレートが属しているビジネス・プロセス・アプリケーションをアンインストールするには、まずそのビジネス・テンプレートを停止する必要があります。以下のステップでは、管理コマンドを使用して、プロセス・テンプレートを停止および開始する方法について説明します。

1. Business Process Choreographer サンプル・ディレクトリーに変更します。 次のように入力します。

   ```
   cd install_root/ProcessChoreographer/sample
   ```
2. プロセス・テンプレートを停止します。

   ```
   install_root/bin/wsadmin -f bpcTemplates.jacl
                                -stop application_name
   ```

   ここで *application_name* は、そのテンプレートが所属するアプリケーションの名前です。

   プロセス・テンプレートの既存のインスタンスは、正常に終了するまで稼働を続けます。アプリケーションが停止している場合、停止したテンプレートからプロセス・インスタンスを作成することはできません。
3. プロセス・テンプレートを開始します。

   ```
   install_root/bin/wsadmin -f bpcTemplates.jacl
                                -start application_name
   ```

   プロセス・テンプレートが開始されます。Business Process Choreographer Explorer を使用して、プロセス・テンプレートからプロセス・インスタンスを開始することができます。

プロセス・ライフ・サイクルの管理

プロセスは、開始してから終了までにさまざまな状態を通過します。プロセス管理者は、プロセスのライフ・サイクル中にさまざまなアクションをとることができます。

新規プロセス・インスタンスの開始:

新規プロセス・インスタンスは、使用を許可されている任意のプロセス・テンプレートから開始できます。

インストールされたプロセス・テンプレートは、すべて Business Process Choreographer Explorer でプロセス・テンプレートのリストに表示されます。新規プロセス・インスタンスを開始するには、次のステップを実行します。

1. 使用を許可されているプロセス・テンプレートを表示します。

   ナビゲーション・ペインの「プロセス・テンプレート」の下で、「**自分のプロセス・テンプレート**」をクリックします。
2. プロセス・テンプレートの横にあるチェック・ボックスを選択し、「**インスタンスの開始**」をクリックします。

   このアクションにより、「プロセス入力メッセージ」ページが表示されます。

   プロセスに複数の操作が存在する場合、このアクションによって、すべての使用可能な操作を含むページが表示されます。プロセス・インスタンスを開始するための操作を選択します。
3. プロセス・インスタンスを開始するための入力データを提供します。

   プロセスが長期実行プロセスである場合は、プロセス・インスタンス名に入力できます。名前を指定しない場合、新規プロセス・インスタンスには、システムによって生成された名前が割り当てられます。

   プロセス入力メッセージに対する入力を完了します。
4. プロセスを開始するには、「**実行依頼**」をクリックします。

プロセス・インスタンスが開始されます。ビジネス・プロセスに、人間による対話を必要とするアクティビティーが含まれている場合は、潜在的な所有者すべてに対してタスクが生成されます。潜在的な所有者である場合は、このタスクが「自分のタスク」ページのリストに表示されます。

プロセスが長期実行プロセスである場合は、そのプロセスが終了すると即時にプロセス出力メッセージが表示されます。すべてのプロセスに出力メッセージがあるわけではありません。例えば、プロセスが片方向操作をインプリメントしている場合、出力メッセージは表示されません。

プロセス・インスタンスの進行状況のモニター:

プロセス・インスタンスの進行状況をモニターすると、プロセスが最後まで実行できるように何らかのアクションを起こす必要があるかどうかを判断できます。

プロセス・インスタンスをモニターするには、 Business Process Choreographer Explorer で次のステップを実行します。

1. プロセス・インスタンスのリストを表示します。

   例えば、ナビゲーション・ペインの「プロセス・インスタンス」の下で、「**自分で管理**」をクリックします。
2. プロセス・インスタンスの横にあるチェック・ボックスを選択し、「**プロセス状態の表示**」をクリックします。

このアクションにより、「プロセス状態」ページが表示されます。このページには、アクティビティー、遷移を含むリンクとそのリンクの結合条件、障害ハンドラー、補正ハンドラー、およびプロセスに定義されるイベント・ハンドラーが表示されます。太字で表示されるアクティビティーは、プロセス・モデル内でビジネス関連として定義されています。これらのアクティビティーに対しては、状態情報が表示されます。

3. アクティビティーを操作するには、そのアクティビティーをクリックします。

   このアクションで「アクティビティー」ページが表示され、これらのアクションはこのページで実行できます。

プロセス・インスタンスの中断と再開:

長期間にわたって実行するトップレベルのプロセス・インスタンスを中断することができます。これは、例えば、プロセスの後半で使用するバックエンド・システムへのアクセスの構成、あるいはプロセス・インスタンスの失敗の原因となっている問題の修正を行えるよう実行することができます。プロセスの前提条件が満たされている場合は、プロセス・インスタンスの実行を再開できます。

プロセス・インスタンスを中断して再開するには、プロセス管理者権限が必要です。

プロセス・インスタンスを中断するには、プロセス・インスタンスが実行中または失敗している状態である必要があります。プロセスを再開するには、プロセス・インスタンスが中断された状態である必要があります。

プロセス・インスタンスを中断または再開するには、 Business Process Choreographer Explorer で次のステップを完了します。

1. プロセス・インスタンスのリストを表示します。

   例えば、ナビゲーション・ペインの「プロセス・インスタンス」の下で、「**自分で管理**」をクリックします。

2. プロセスを中断します。

   プロセス・インスタンスの横にあるチェック・ボックスを選択し、「**中断**」をクリックします。

   このアクションにより、指定されたトップレベルのプロセス・インスタンスが中断されます。プロセス・インスタンスは中断状態になります。autonomy 属性が `child` に設定されたサブプロセスも、状態が実行中、失敗、終了、または補正になっていれば中断されます。ただし、その場合も、プロセス・インスタンスに属するアクティブなアクティビティーおよびタスクは完了できます。

3. プロセス・インスタンスを再開します。

   中断状態のプロセス・インスタンスを選択し、「**再開**」をクリックします。 プロセス・インスタンスとそのサブプロセスは、中断される前の状態 (例えば、実行中) になります。プロセス・インスタンスおよびそのサブプロセスが再開します。

プロセス・インスタンスの終了:

例えば、プロセス・インスタンスが示している作業または文書が必要なくなった場合や、プロセス・インスタンスを完了できるユーザーがいない場合、プロセス・テンプレートに問題が発生して再設計が必要な場合などは、プロセス・インスタンスを終了することができます。

プロセス・インスタンスを終了するには、プロセス管理者権限が必要です。

プロセス・インスタンスを終了するには、 Business Process Choreographer Explorer で次のステップを実行します。ビジネス・プロセス・モデルに補正が定義されている場合、補正を持つプロセス・インスタンスの終了を選択できます。

1. 管理できるプロセス・インスタンスを表示します。

   ナビゲーション・ペインの「プロセス・インスタンス」の下で、「**自分で管理**」をクリックします。
2. 停止するプロセス・インスタンスの横にあるチェック・ボックスを選択します。
   - 補正を持つプロセス・インスタンスを終了させるには、「**補正**」をクリックします。

     このアクションで、プロセス・インスタンスは終了し、補正処理が開始します。
   - 補正を持たないプロセス・インスタンスを終了させるには、「**終了**」をクリックします。

     このアクションにより、プロセス・インスタンスは未解決のアクティビティーまたはタスクを待たず、即時に停止します。終了したプロセス・インスタンスは補正されません。

プロセス・インスタンスの削除:

プロセス・インスタンスが完了時に自動的に削除されないように、プロセス・テンプレートをモデル化することができます。これらのプロセス・インスタンスは、完了後は明示的に削除できます。

プロセス・インスタンスを削除するには、プロセス管理者権限が必要です。プロセス・インスタンスは、終了または強制終了の状態になっている必要があります。

対応するプロパティーが、プロセス・モデルのプロセス・テンプレート用に設定されている場合、完了したプロセス・インスタンスは自動的に Business Process Choreographer データベースから削除されます。

データベースにプロセス・インスタンスを保持する必要がある場合があります。例えば、監査ログに書き込まれていないプロセス・インスタンスからのデータを照会する場合、またはオフピーク時に、プロセスの削除を延期したい場合です。ただし、不要になった以前のプロセス・インスタンスのデータが、ディスク・スペースおよびパフォーマンスに影響を与える可能性があります。したがって、不要になった、または保持する必要がなくなったプロセス・インスタンスのデータは、定期的に削除する必要があります。この保守タスクは、オフピーク時に実行するようにしてください。

完了したプロセス・インスタンスを削除する場合は、例えば個々のプロセス・インスタンスを削除するには Business Process Choreographer Explorer を使用し、複数のプロセス・インスタンスを一度に削除するには deleteCompletedProcessInstances 管理スクリプトを使用します。

プロセス・インスタンスを削除するには、 Business Process Choreographer Explorer で次のステップを実行します。

1. 管理するプロセス・インスタンスを表示します。

   ナビゲーション・ペインの「プロセス・インスタンス」の下で、「**自分で管理**」をクリックします。
2. 削除するプロセス・インスタンスを選択し、「**削除**」をクリックします。

このアクションによって、選択したプロセス・インスタンスがデータベースから削除されます。

プロセスおよびアクティビティーの修復

プロセスに問題が発生した場合は、そのプロセスを分析し、次にアクティビティーを修復することができます。

Business Process Choreographer Explorer では、プロセス管理者が現在実行中のプロセスをモニターするためのさまざまなビューが提供されます。

- アクティビティーが停止状態にあるプロセス・インスタンスを表示するには、ナビゲーション・ペインの「プロセス・インスタンス」で、「**重大なプロセス**」をクリックします。
- 特定のプロセス・インスタンスの進行状況をモニターするには、プロセス・インスタンスのリストを表示する任意のビューで、「**プロセス状態の表示**」をクリックします。

これで、保留アクティビティーを修復できます。

アクティビティーの再開:

アクティビティーを修復した場合は、新しい入力データでアクティビティーを再開することができます。

アクティビティーは停止状態、関連付けられたプロセス・インスタンスは実行状態である必要があります。

アクティビティーを再開するには、Business Process Choreographer Explorer で次のステップを実行します。

1. アクティビティーの「アクティビティー」ページに移動して、「**再始動**」をクリックします。
2. アクティビティーの再開に必要な入力データを指定します。

   アクティビティーの再開時にエラーが発生してもプロセスを続行する場合は、「**エラーの継続**」を選択します。
3. 「**再始動**」をクリックします。

アクティビティーの強制完了:

例えば呼び出されたサービスが使用できなくなったなどの理由でアクティビティーが予定どおりに完了しないことがわかった場合、アクティビティーを強制完了してプロセス・フローを続行することができます。

一般的に、アクティビティーは停止状態である必要があります。ただし、アクティビティーがスタッフ・アクティビティーである場合、アクティビティーは作動可能状態または要求済み状態のどちらかになることもあります。関連付けられたプロセス・インスタンスは実行状態である必要があります。

アクティビティーを強制完了するには、Business Process Choreographer Explorer で次のステップを実行します。

1. アクティビティーの「アクティビティー」ページに移動して、「**強制完了**」をクリックします。
2. アクティビティーの完了に必要なデータを指定します。
3. もう一度「**強制完了**」をクリックします。

Microflow の補正の管理:

Microflow の実行時に、問題が発生する場合があります。そのような場合、プロセス・モデルでプロセスに対して補正が定義されている可能性があります。補正によって、直前に完了したステップを元に戻すことができます。例えば、データおよび状態をリセットして、これらの問題からリカバリーできます。

Microflow を補正するには、管理コンソールで補正サービスが開始されている必要があります。

Microflow の補正アクションが失敗した場合は、プロセス管理者が問題を解決するために介入する必要があります。

失敗した補正アクションを管理するには、Business Process Choreographer Explorer で次のステップを実行します。

1. 失敗した補正アクションのリストを表示します。

   ナビゲーション・ペインの「プロセス・インスタンス」の下で、「**失敗した補正**」をクリックします。

   「失敗した補正」ページが表示されます。このページには、名前付き補正アクションが失敗した理由に関する情報が含まれています。この情報は、失敗した補正を訂正するために実行すべきアクションを判別するうえで役立ちます。
2. アクティビティーの横にあるチェック・ボックスを選択してから、使用可能なアクションのいずれかをクリックします。

   次の管理アクションが使用可能です。

   スキップ
   現在の補正アクションをスキップし、Microflow の補正を続行します。補正されていないアクティビティーが発生することになります。

   再試行 失敗した補正アクションを訂正するためのアクションを実行するには、「**再試行**」をクリックして、補正アクションを再試行します。

   停止 補正処理を停止します。

ビジネス・プロセスの補正:

補正は、正常に完了したプロセス内の操作を元に戻すための手段です。

補正処理は、プロセス・モデルで補正が定義された実行中のプロセス・インスタンスでエラーが発生した場合に開始されます。補正は、エラーが発生した時点までにコミットされた操作の影響をリバースし、整合した状態に戻します。

プロセス・モデルで、長期実行プロセスおよび Microflow に対する補正を定義できます。

長期実行プロセスの補正

長期実行プロセスに対する補正は、*ビジネス・レベル補正*とも呼ばれます。このタイプの補正は、スコープ・レベルで定義されています。これは、プロセスの一部、またはプロセス全体が補正可能ということです。

補正は、障害ハンドラー、スコープまたはプロセスの補正ハンドラーによって起動されます。補正は、プロセスのもう一つのナビゲーション・パスです。

長期実行プロセスは、子プロセスを囲む親スコープが補正されたときに、正常終了した子プロセスを自動的に補正します。プロセス内では、正常に完了した起動およびスコープ・アクティビティーのみ補正されます。

Microflow の補正

Microflow の補正は、*テクニカル補正*としても知られています。このタイプの補正は、Microflow が含まれている作業単位 (トランザクションまたはアクティビティー・セッション) がロールバックされた場合に起動されます。したがって、通常、取り消しアクションは、作業単位のロールバックによってリバースすることのできないアクティビティーに対して指定されます。プロセス・インスタンスが実行されると、補正可能アクティビティーに対する取り消しアクションが、それを囲む作業単位に登録されます。この作業単位 (ロールバックまたはコミット) の結果によって、補正が開始されます。

Microflow が補正可能な長期実行プロセスの子である場合は、Microflow の完了時に、親プロセスに対して Microflow の取り消しアクションが使用可能になります。したがって、Microflow は親プロセスの補正に参加できる可能性があります。このようなタイプの Microflow では、プロセス・モデルを定義する際にプロセス内のすべてのアクティビティーに対して取り消しアクションを指定することをお勧めします。

補正処理中にエラーが発生した場合、補正アクションでは、エラーを手動で解決する必要があります。Business Process Choreographer Explorer を使用して、これらの補正アクションを修復することができます。

タスク・テンプレートとタスク・インスタンスの管理

管理コンソールおよび管理コマンドを使用して、タスク・テンプレートを管理します。タスク・インスタンスの処理には Business Process Choreographer Explorer を使用します。

ヒューマン・タスクのための許可のロール

ヒューマン・タスクで実行できるアクションは、ご使用の許可のロールに依存します。このロールは、J2EE のロールまたはインスタンス・ベースのロールとすることができます。

ロールとは、同じ権限レベルを共用する従業員の集合です。Java 2 Platform, Enterprise Edition (J2EE) のロールは、ヒューマン・タスク・コンテナーが構成されたときにセットアップされます。インスタンス・ベースのロールは、タスクがモデル化されたときにヒューマン・タスクおよびエスカレーションに割り当てられます。ロール・ベースの許可については、グローバル・セキュリティーが WebSphere Application Server で使用可能になっていることが必要です。

J2EE のロール

以下の J2EE のロールがサポートされています。

- J2EE TaskSystemAdministrator。このロールを割り当てられたユーザーは、すべての特権を持ちます。また、このロールは、ヒューマン・タスクのシステム管理者とも言われます。
- J2EE TaskSystemMonitor。このロールを割り当てられたユーザーは、すべてのタスク・オブジェクトのプロパティーを表示できます。また、このロールは、ヒューマン・タスクのシステム・モニターとも言われます。

これらのロールへのユーザーおよびグループの割り当てを変更するには、管理コンソールを使用できます。

RACF セキュリティーによるロールの設定: この RACF 許可は、以下のセキュリティー・フィールドを指定した場合に適用されます。

- **com.ibm.security.SAF.authorization= true**

```
RDEFINE EJBROLE TaskSystemAdministrator  UACC(NONE)
PERMIT TaskSystemAdministrator CLASS(EJBROLE)  ID(userid) ACCESS(READ)

RDEFINE EJBROLE TaskSystemMonitor  UACC(NONE)
PERMIT TaskSystemMonitor  CLASS(EJBROLE)  ID(userid) ACCESS(READ)
```

- **com.ibm.security.SAF.delegation= true**

```
RDEFINE EJBROLE JMSAPIUser  UACC(NONE) APPLDATA(' userid')
```

Security Authorization Facility (SAF) ベースの許可 (RACF EJBROLE プロファイルの使用など) を使用して、WebSphere Application Server 管理コンソール・アプリケーションなど、EJB および Web アプリケーションにおける Java 2 Platform、Enterprise Edition (J2EE) ロールに対するクライアントのアクセスを制御できます。詳細については、 WebSphere Application Server for z/OS インフォメーション・センターの『役割ベースの許可の System Authorization Facility』を参照してください。

インスタンス・ベースのロール

タスク・インスタンスまたはエスカレーション・インスタンスはタスク・モデルではスタッフ・メンバーには直接割り当てられておらず、その代わり、使用可能なロールの 1 つに割り当てられます。インスタンス・ベースのロールに割り当てられた

スタッフ・メンバーは、そのロールに応じたアクションを実行できます。インスタンス・ベースのロールとユーザーの関連付けは、スタッフの解決を使用して実行時に決定されます。

以下のインスタンス・ベースのロールがサポートされています。
- タスクに対して: 可能なインスタンス作成者、オリジネーター、可能なスターター、スターター、可能な所有者、所有者、読者、編集者、管理者
- エスカレーションに対して: エスカレーション受信者

これらのロールは、以下のアクションの実行が許可されています。

ロール	許可されたアクション
潜在的インスタンス作成者	このロールのメンバーは、タスクのインスタンスを作成できます。タスク・テンプレートまたはアプリケーション・コンポーネントに対して可能なインスタンス作成者が定義されていない場合、すべてのユーザーがこのロールのメンバーとして考慮されます。
オリジネーター	このロールのメンバーは、タスクが開始されるまで管理権限を持ちます。タスクが開始されると、オリジネーターは読者の権限を持ち、タスクの中断や再開、および作業項目の転送などの一部の管理アクションを実行できます。
可能なスターター	このロールのメンバーは、既存のタスク・インスタンスを開始できます。可能なスターターが指定されない場合、オリジネーターが可能なスターターになります。可能なスターターがない場合のインライン・タスクについては、デフォルトは全員となります。
スターター	このロールのメンバーは読者の権限を持ち、作業項目の転送などの一部の管理アクションを実行できます。
可能な所有者	このロールのメンバーはタスクを要求できます。タスク・テンプレートまたはアプリケーション・コンポーネントに対して可能な所有者が定義されていない場合、すべてのユーザーがこのロールのメンバーとして考慮されます。このロールのスタッフの解決が失敗する場合、管理者は潜在的な所有者として割り当てられます。
所有者	タスクを処理し、完了します。
読者	すべてのタスク・オブジェクトのプロパティーを表示できますが、操作はできません。
編集者	このロールのメンバーはタスクの内容を扱うことができますが、要求または完了することはできません。
管理者	このロールのメンバーは、タスク、タスク・テンプレート、およびエスカレーションを管理できます。
エスカレーション受信者	このロールのメンバーは、エスカレーションおよびエスカレートされたタスクの読者権限を持ちます。

管理コンソールによるタスク・テンプレートの停止および開始

管理コンソールを使用して、タスク・テンプレートを開始および停止します。

グローバル・セキュリティーが使用可能に設定されている場合は、ユーザー ID にオペレーター権限があることを確認します。

タスク・テンプレートは、エンタープライズ・アプリケーション内でスタンドアロン・タスクと表される Service Component Architecture (SCA) サービスを定義します。タスク・テンプレートを含むエンタープライズ・アプリケーションがインストールされ、配置され、開始されると、タスク・テンプレートが開始状態になります。

1. 管理するアプリケーションを選択します。

   管理コンソールのナビゲーション・ペインで、「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」をクリックしてから、管理するアプリケーションをクリックします。
2. 「**関連項目**」の下のエンタープライズ・アプリケーションの「構成」ページで、「**EJB モジュール**」をクリックしてから、Enterprise JavaBeans モジュールをクリックします。
3. 「**追加プロパティー**」の下の EJB モジュールの「構成」ページで、「**ヒューマン・タスク**」をクリックしてからプロセス・テンプレートをクリックします。
4. タスク・テンプレートを停止するには、「**停止**」をクリックします。
5. タスク・テンプレートを開始するには、「**開始**」をクリックします。

管理コマンドによるタスク・テンプレートの停止および開始

管理コマンドは、タスク・テンプレートを停止および開始するため管理コンソールの代わりを提供します。

グローバル・セキュリティーが使用可能に設定されている場合は、オペレーター権限を持つユーザー ID でログインしていることを確認します。

タスク・テンプレートは、エンタープライズ・アプリケーション内でスタンドアロン・タスクと表される Service Component Architecture (SCA) サービスを定義します。タスク・テンプレートを含むエンタープライズ・アプリケーションがインストールされ、配置され、開始されると、タスク・テンプレートが開始状態になります。

1. Business Process Choreographer サンプル・ディレクトリーに変更します。 次のように入力します。

   ```
   cd install_root/ProcessChoreographer/sample
   ```
2. タスク・テンプレートを停止します。

   ```
   install_root/bin/wsadmin -f bpcTemplates.jacl
                                -stop application_name
   ```

   ここで *application_name* は、そのテンプレートが所属するアプリケーションの名前です。 タスク・テンプレートの既存のインスタンスは、正常に終了するまで稼働を続けます。
3. タスク・テンプレートを開始します。

   ```
   install_root/bin/wsadmin -f bpcTemplates.jacl
                                -start application_name
   ```

   タスク・テンプレートが開始されます。Business Process Choreographer Explorer を使用して、タスク・テンプレートに関連付けられているタスク・インスタンスを処理できます。

タスク・インスタンスの作成と開始

タスク・インスタンスは、使用を許可されているタスク・テンプレートのどれからでも作成して開始することができます。

インストールされたタスク・テンプレートは、すべて Business Process Choreographer Explorer でタスク・テンプレートのリストに表示されます。タスク・テンプレートからタスク・インスタンスを作成して開始するには、次のステップを実行します。

1. 使用を許可されているタスク・テンプレートを表示します。

   ナビゲーション・ペインの「タスク・テンプレート」の下で、「**自分のタスク・テンプレート**」をクリックします。
2. タスク・テンプレートの横にあるチェック・ボックスを選択し、「**インスタンスの開始**」をクリックします。

   このアクションにより、「タスク入力メッセージ」ページが表示されます。
3. タスク・インスタンスを開始するための入力データを提供します。
4. タスク・インスタンスを開始するには、「**実行依頼**」をクリックします。

タスク・インスタンスで作業する準備が整いました。

タスクの操作

タスクを操作するには、タスクを要求してから、それを完了するために必要なアクションを実行します。

タスクが作動可能状態である場合、タスクの潜在的な所有者または管理者は、そのタスクを要求できます。タスクを要求したユーザーは、そのタスクの所有者になり、タスクの完了に責任を負います。

タスクのリストには、ユーザーが読者または編集者のロールを持っているタスクも表示されます。

Business Process Choreographer Explorer を使用してタスクを要求し完了するには、次のステップを実行します。

1. 自分に割り当てられているタスクを表示します。

   「**タスク・インスタンス**」 → 「**自分のタスク**」をクリックします。

   このアクションにより、「自分のタスク」ページが表示され、割り当てられているタスクがリストされます。
2. 操作するタスクを要求します。

   タスクの横にあるチェック・ボックスを選択し、「**処理**」をクリックします。

   このアクションにより、「タスク・メッセージ」ページが表示されます。
3. タスクを完了する情報を提供します。

   例えばタスクを完了するために同僚からの情報が必要な場合など、操作を中断する必要がある場合は、「**保管**」をクリックして変更を保管します。

4. 「**完了**」をクリックして、提供した情報でタスクを完了します。

完了したタスクは、完了状態になります。タスクを完了しないと、そのタスクは要求済み状態のままです。

作業割り当ての管理

タスクの開始後に、そのタスクに対する作業割り当ての管理が必要になる場合があります (例えば、ワークグループのメンバー間でワークロードを効率よく分散するため)。

*作業項目*は、特定の理由でのユーザーまたはユーザー・グループに対する、タスクまたはプロセス・インスタンスなどのビジネス・エンティティーの割り当てです。割り当ての理由により、ユーザーは、可能な所有者、編集者、または管理者など、ビジネス・プロセス・シナリオでのさまざまなロールを演じることができます。

さまざまなユーザーがさまざまなロールを持つことができるので、タスク・インスタンスには幾つかの作業項目を関連付けることができます。例えば、John、Sarah、Mike はすべてタスク・インスタンスの潜在的な所有者であり、Anne は管理者です。作業項目は、この 4 人全員に対して生成されます。 John、Sarah、Mike のタスク・リストには、それぞれ自分の作業項目だけがタスクとして表示されます。Anne は管理者なので、自分のタスクの作業項目を取得し、さらに John、Sarah、Mike に対して生成された作業項目を管理できます。

場合によっては、タスクの開始後にタスク割り当てを変更する必要があります。例えば、元の所有者から別のユーザーへ作業項目を転送します。追加の作業項目の作成や、必要なくなった作業項目の削除を実行しなければならない場合もあります。

所有するタスクの転送:

タスクの所有者は、タスクを別のユーザーに転送しなければならない場合があります (例えば、他の誰かがタスクを完了するための情報を提供する必要がある場合など)。

Business Process Choreographer Explorer で、次のステップを実行し、所有するタスクを転送します。

1. 所有するタスクを表示します。

   ナビゲーション・ペインの「タスク・インスタンス」の下で、「**ユーザーのタスク**」をクリックします。
2. 転送するタスクの横にあるチェック・ボックスを選択し、「**転送**」をクリックします。
3. タスクを転送します。

   「**新規所有者**」フィールドで、新規タスク所有者のユーザー ID を指定し、「**転送**」をクリックします。タスクは、別の潜在的所有者またはタスク管理者にのみ転送できます。

転送済みのタスクは、新規タスク所有者に属するタスクのリストに表示されます。

自分がスターター、オリジネーター、またはタスクの管理者である作業項目の転送:

タスクで作業を始めた後に作業割り当てを変更する必要がでてくる場合があります。例えば、タスクの所有者が休暇中で、その所有者が出社する前にタスクを完了しなければならない場合、作業項目を別のユーザーに転送する必要があります。所有しているロールとタスクの状態によって作業項目の転送方法は変わります。

作業項目を転送するには、次のいずれかのロールを持っている必要があり、タスクは次の状態のいずれかである必要があります。

ロール	**タスク状態**	**作業項目は、次のユーザー・ロールに転送可能です。**
所有者	要求	潜在的所有者、管理者。
スターター	強制終了、期限切れ、終了、失敗、または実行中	潜在的スターター、管理者。
オリジネーター	任意のタスク状態	潜在的インスタンス作成者、管理者。タスクがアクティブ状態の場合は、任意のユーザー・ロールに転送できます。
管理者	作動可能、要求済み、強制終了、期限切れ、終了、失敗、または実行中	任意のユーザー・ロール。

作業項目を転送するには、 Business Process Choreographer Explorer で次のステップを実行します。

1. 管理できるタスク・インスタンスを表示します。

   ナビゲーション・ペインの「タスク・インスタンス」の下で、「**自分で管理**」をクリックします。
2. タスク・インスタンスの作業項目を表示します。

   「自分で管理するタスク・インスタンス」ページで、タスク・インスタンスの横にあるチェック・ボックスを選択して「**作業項目**」をクリックします。
3. 作業項目を転送します。
   a. 「**新規所有者**」フィールドで、新規作業項目所有者のユーザー ID を指定します。
   b. 1 つ以上の作業項目を選択し、「**転送**」をクリックします。

新規の作業項目所有者を持つ転送済み作業項目が、作業項目のリストに表示されます。

作業項目の作成:

例えば現在の潜在的な所有者が追加作業を受け入れられない場合などに、新規の潜在的な所有者の作業項目を作成できます。また、スタッフ・リポジトリーに対する照会が潜在的な所有者を戻さない場合に、作業項目を作成することもできます。これは、例えば、プロセスの開始以降に組織が変更した場合に、長期実行プロセスで発生します。

タスク・インスタンスの作業項目を作成するには、タスクに対する適切なロールを持っている必要があります。タスク管理者は、タスク・インスタンスの状態が、作動可能、要求済み、実行中、終了、または失敗のいずれかになっていれば、そのタスク・インスタンスの作業項目を作成できます。タスク・インスタンスがタスク・テンプレートから派生している場合は、タスクが強制終了または期限切れ状態のときにも作業項目を作成できます。

作業項目を作成するには、 Business Process Choreographer Explorer で次のステップを実行します。

1. 管理するタスク・インスタンスを表示します。

   ナビゲーション・ペインの「タスク・インスタンス」の下で、**「自分で管理」**をクリックします。
2. 作業項目を作成する対象のタスク・インスタンスの横にあるチェック・ボックスを選択し、**「作業項目の作成」**をクリックします。 「作業項目の作成」ページが表示されます。
3. 作業項目を作成します。
   a. **「新規所有者」**フィールドで、新規作業項目所有者のユーザー ID を指定します。
   b. **「理由」**リストから 1 つ以上のロールを選択します。

      これらのロールにより、割り当てられたユーザーが新規作業項目で実行できるアクションが決まります。
   c. **「作成」**をクリックします。

作業項目は、新規所有者に対して指定する各ロールに対して作成されます。新規タスクは、このユーザーに割り当てられるタスクのリストに表示されます。

作業項目の削除:

例えば、エラーの作業項目を作成した場合や、もう会社で働いていない人に対して作業項目が生成されている場合は、作業項目を削除することができます。

タスク・インスタンスの作業項目を削除するには、タスクに対する適切なロールを持っている必要があります。タスク管理者は、タスク・インスタンスの状態が、作動可能、要求済み、実行中、終了、または失敗のいずれかになっていれば、そのタスク・インスタンスを削除できます。タスク・インスタンスがタスク・テンプレートから派生している場合は、強制終了または期限切れ状態のタスクも削除できます。

作業項目を削除するには、 Business Process Choreographer Explorer で次のステップを実行します。

1. 管理するタスク・インスタンスを表示します。

   ナビゲーション・ペインの「タスク・インスタンス」の下で、**「自分で管理」**をクリックします。
2. タスク・インスタンスの作業項目を表示します。

「自分で管理するタスク・インスタンス」ページで、タスク・インスタンスを選択して「**作業項目**」をクリックします。

3. 作業項目を削除します。

   1 つ以上の作業項目を選択し、「**削除**」をクリックします。

作業項目が削除されます。

タスク・エスカレーションの表示

エスカレーションは、割り当てられたタスクをユーザーが時間どおりに完了できない可能性があることをエスカレーション受信者に通知します。

タスクが期限切れになると、エスカレーションが発生する場合があります。エスカレーションの結果として、次のアクションが発生します。

- 例えば、マネージャーが問題の解決をサポートするための処置をとれるように、新規の作業項目が作成されます。
- ヒューマン・タスク・コンテナーの構成時に E メール設定を指定した場合は、エスカレートされたタスクについて知らせるために、指定された担当者に E メールが送信されます。
- イベント通知ハンドラーが呼び出されます。

エスカレーションを表示するには、「タスク・インスタンス」の下にある「**自分のエスカレーション**」をクリックします。

- エスカレーションに関する情報を表示するには、エスカレーション ID をクリックします。
- エスカレートされたタスクに関する情報を表示するには、タスク名をクリックします。

エスカレーションを知らせる E メールの送信:

タスクが期限切れになると、エスカレーションが発生する場合があります。指定のユーザーにエスカレーションについて知らせる E メールを送信するように、システムをセットアップすることができます。

Lightweight Directory Access Protocol (LDAP) 構成に、エスカレーションの通知が必要なユーザーの E メール・アドレスが含まれていることを確認します。

1. WebSphere Integration Developer のヒューマン・タスク・エディターで、タスクに以下のアクションを実行します。
   a. プロパティー領域の「**詳細**」タブのタスク設定の下で、「**スタッフ・プラグイン構成の JNDI 名 (JNDI name of staff plugin configuration)**」フィールドの値を編集します。

      Java Naming and Directory Interface (JNDI) 名を `bpe/staff/sampleldapconfiguration`、または任意の LDAP プロバイダー構成 JNDI 名に設定します。

   b. プロパティー領域の「**詳細**」タブのエスカレーション設定で、「**通知タイプ (Notification type)**フィールドの値を `E-mail` に設定します。

c. エスカレーション通知のために送信される E メールの本文用テキストを指定します。

テキストを指定しない場合は、デフォルトのメッセージ・テキストが使用されます。

2. WebSphere Process Server で、以下のアクションを実行します。
   a. Simple Mail Transfer Protocol (smtp) ホストが設定されていることを確認します。

      管理コンソールで、「**メール・プロバイダー (Mail Providers)**」 → 「**ビルトイン・メール・プロバイダー (Built-in Mail Provider)**」 → 「**メール・セッション**」 → **HTMMailSession_*server*** と進み、この設定をチェックします。smtp ホストはセル・レベルで定義されます。
   b. ヒューマン・タスク・コンテナーの構成時に指定した送信側の E メール・アドレスが有効であることを確認します。

エスカレーション E メールで問題が発生した場合は、`SystemOut.log` ファイルでエラー・メッセージを確認してください。

ビジネス・プロセスおよびアクティビティーについての報告

ビジネス・プロセスおよびアクティビティーの処理中には、プロセス、アクティビティー、タスクのいずれかの状態が変更されると、イベントが生成されます。これらのイベントを保管し、Business Process Choreographer Observer によるレポート作成に利用できます。例えば、プロセス・ボトルネックの分析レポートや、アクティビティーから呼び出されるサービスの信頼性評価のレポートなどです。

事前定義のレポートを使用することも、プロセスおよびアクティビティーのユーザー定義レポートを作成することもできます。

事前定義レポート

事前定義のリストや図表では、イベント情報および状態情報を入手するためにドリルダウン・アプローチが採用されています。例えば、日付などのフィルター基準を指定して、アクティビティー・インスタンスのデータを棒グラフで表示することができます。

プロセスおよびアクティビティーのユーザー定義レポート

ユーザー定義のプロセス・レポートとアクティビティー・レポートは、事前定義のリストおよび図表に比べると柔軟性に優れています。また、レポート定義を格納したり取り出したりすることができます。プロセス・レポートでは、プロセス・インスタンスに属するアクティビティーの情報を取得できます。アクティビティー・レポートでは、関連するプロセス・インスタンスの情報も取得できます。

第 5 章 開発

ビジネス・プロセスおよびタスク用クライアント・アプリケーションの開発

モデル化ツールを使用して、ビジネス・プロセスやタスクを作成しデプロイすることができます。そのようなプロセスとタスクは、実行時に相互作用を受けます。例えば、プロセスが開始され、タスクが要求され完了します。プロセスおよびタスクとは、Business Process Choreographer Explorer を使用して対話できますが、Business Process Choreographer API を使用して、このような対話用にカスタマイズしたクライアントを開発することもできます。

このクライアントは、Enterprise JavaBeans™ (EJB) クライアント、Web サービス・クライアント、または Business Process Choreographer Explorer JavaServer Faces (JSF) コンポーネントを利用する Web クライアントのいずれかです。これらのクライアントを開発するために、Business Process Choreographer は、Enterprise JavaBeans (EJB) API と Web サービス用インターフェースを提供しています。EJB API には、任意の Java アプリケーション (別の EJB アプリケーションを含む) からアクセスできます。 Web サービス用インターフェースには、Java 環境または Microsoft® .Net 環境のいずれかからアクセスできます。

ビジネス・プロセスおよびヒューマン・タスク用 EJB クライアント・アプリケーションの開発

EJB API は、WebSphere Process Server 上にインストールされているビジネス・プロセスやヒューマン・タスクを処理する EJB クライアント・アプリケーションを開発するための汎用的な方法をいくつか提供します。

この Enterprise JavaBeans (EJB) API を使用すれば、以下を実行するためのクライアント・アプリケーションを作成できます。

- プロセスやタスクのライフ・サイクルの、開始から完了後の削除までの管理
- アクティビティーやプロセスの修復
- ワークグループのメンバーに対するワークロードの管理および配布

EJB API は、次の 2 種類のステートレス・セッション・エンタープライズ Bean として提供されます。

- BusinessFlowManagerService インターフェースは、ビジネス・プロセス・アプリケーション用のメソッドを備えています。
- HumanTaskManagerService インターフェースは、タスク・ベースのアプリケーション用のメソッドを備えています。

EJB API の詳細は、com.ibm.bpe.api パッケージおよび com.ibm.task.api パッケージの中の Javadoc を参照してください。

以下のステップは、 EJB クライアント・アプリケーションの開発に必要なアクションの概要です。

1. アプリケーションが提供する機能を決定します。

2. 使用するセッション Bean を決定します。

   アプリケーションでインプリメントするシナリオに応じて、2 つのセッション Bean のうちの 1 つ、または両方を使用することができます。
3. アプリケーションのユーザーが必要とする許可権限を判別します。

   アプリケーションのユーザーは、アプリケーションに組み込まれたメソッドを呼び出すこと、およびそれらのメソッドが戻すオブジェクトとそのオブジェクトの属性を表示するのに適した権限ロールを割り当てられていなければなりません。該当するセッション Bean のインスタンスを作成するときに、WebSphere Application Server がコンテキストとそのインスタンスを関連付けます。コンテキストには呼び出し元のプリンシパル ID、グループ・メンバーシップ・リスト、およびロールに関する情報が含まれています。この情報は、それぞれの呼び出しごとに、呼び出し元の権限を確認するために使用されます。

   Javadoc には、各メソッドの許可情報が含まれています。
4. アプリケーションをレンダリングする方法を決めます。

   EJB API は、ローカル側でもリモート側でも呼び出すことができます。
5. アプリケーションを開発します。
   a. EJB API にアクセスします。
   b. EJB API を使用して、プロセスまたはタスクと対話します。
      - データを照会します。
      - データで作業を行います。

EJB API へのアクセス

Enterprise JavaBeans (EJB) API は、次の 2 種類のステートレス・セッション・エンタープライズ Bean として提供されます。ビジネス・プロセス・アプリケーションおよびタスク・アプリケーションは、 Bean のホーム・インターフェースを介して、適切なセッション・エンタープライズ Bean にアクセスします。

BusinessFlowManagerService インターフェースは、ビジネス・プロセス・アプリケーション用のメソッドを提供します。HumanTaskManagerService インターフェースは、タスク・ベースのアプリケーション用のメソッドを提供します。このアプリケーションは、別の Enterprise JavaBeans (EJB) アプリケーションを含む任意の Java アプリケーションです。

リモート・セッション Bean へのアクセス:

EJB クライアント・アプリケーションは、Bean のホーム・インターフェースを介して、適切なリモート・セッション Bean にアクセスします。

セッション Bean は、プロセス・アプリケーションに対しては BusinessFlowManager セッション Bean、タスク・アプリケーションに対しては HumanTaskManager セッション Bean のいずれかである可能性があります。

1. リモート・セッション Bean への参照をアプリケーション・デプロイメント記述子に追加します。 参照を以下のファイルの 1 つに追加します。
   - Java 2 Platform Enterprise Edition (J2EE) クライアント・アプリケーションの場合は、`application-client.xml` ファイル

- Web アプリケーションの場合は、web.xml ファイル
- Enterprise JavaBeans (EJB) アプリケーションの場合は、ejb-jar.xml ファイル

プロセス・アプリケーションの場合のリモート・ホーム・インターフェースへの参照は、以下の例で示されます。

```
<ejb-ref>
 <ejb-ref-name>ejb/BusinessFlowManagerHome</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <home>com.ibm.bpe.api.BusinessFlowManagerHome</home>
 <remote>com.ibm.bpe.api.BusinessFlowManager</remote>
</ejb-ref>
```

タスク・アプリケーションの場合のリモート・ホーム・インターフェースへの参照は、以下の例で示されます。

```
<ejb-ref>
 <ejb-ref-name>ejb/HumanTaskManagerHome</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <home>com.ibm.task.api.HumanTaskManagerHome</home>
 <remote>com.ibm.task.api.HumanTaskManager</remote>
</ejb-ref>
```

WebSphere Integration Developer を使用して EJB 参照をデプロイメント記述子に追加する場合、EJB 参照のバインディングが、アプリケーションのデプロイ時に自動的に作成されます。EJB 参照の追加について詳しくは、WebSphere Integration Developer の文書を参照してください。

2. 生成されたスタブをアプリケーションにパッケージします。

   ご使用のアプリケーションが、BPEContainer アプリケーションまたは TaskContainer アプリケーションを実行しているのと異なる Java 仮想マシン (JVM) で実行されている場合、以下のアクションを完了します。

   a. プロセス・アプリケーションの場合、*<install_root>*/ProcessChoreographer/client/bpe137650.jar ファイルを、ご使用のアプリケーションのエンタープライズ・アーカイブ (EAR) ファイルにパッケージします。
   b. タスク・アプリケーションの場合、*<install_root>*/ProcessChoreographer/client/task137650.jar ファイルを、ご使用のアプリケーションの EAR ファイルにパッケージします。
   c. ビジネス・プロセスまたはヒューマン・タスクの複合データ・タイプを使用し、クライアントが EJB アプリケーションまたは Web アプリケーションで実行しない場合は、対応する XSD または WSDL ファイルを、ご使用のアプリケーションの EAR ファイルにパッケージします。
   d. アプリケーション・モジュールのマニフェスト・ファイル内の **Classpath** パラメーターを、JAR ファイルを含めるように設定します。

      アプリケーション・モジュールは、J2EE アプリケーション、Web アプリケーション、または EJB アプリケーションの可能性があります。

3. Java Naming and Directory Interface (JNDI) からリモート・セッション Bean のホーム・インターフェースへの参照を取得します。

   以下の例では、プロセス・アプリケーションでのこのステップを示します。

```
// Obtain the default initial JNDI context
InitialContext initialContext = new InitialContext();

  // Lookup the remote home interface of the BusinessFlowManager bean
  Object result =
         initialContext.lookup("java:comp/env/ejb/BusinessFlowManagerHome");

// Convert the lookup result to the proper type
   BusinessFlowManagerHome processHome =
           (BusinessFlowManagerHome)javax.rmi.PortableRemoteObject.narrow
           (result,BusinessFlowManagerHome.class);
```

セッション Bean のホーム・インターフェースには、EJB オブジェクトの create メソッドが含まれます。このメソッドは、セッション Bean のリモート・インターフェースを戻します。

4. セッション Bean のリモート・インターフェースにアクセスします。

   以下の例では、プロセス・アプリケーションでのこのステップを示します。

   ```
   BusinessFlowManager process = processHome.create();
   ```

   セッション Bean へのアクセス権は、呼び出し元が Bean が提供するすべてのアクションを実行できることを保証するものではありません。呼び出し元には、そのアクションに対する許可も必要になります。セッション Bean のインスタンスを作成すると、コンテキストとそのセッション Bean のインスタンスが関連付けられます。コンテキストには呼び出し元のプリンシパル ID、グループ・メンバーシップ・リストが含まれ、その呼び出し元に Business Process Choreographer の J2EE ロールのいずれかがあるかどうかを示します。コンテキストは、グローバル・セキュリティーが設定されていなくても、それぞれの呼び出しごとに、呼び出し元の権限を確認するために使用されます。グローバル・セキュリティーが設定されていない場合、呼び出し元のプリンシパル ID の値は UNAUTHENTICATED になります。

5. サービス・インターフェースによって公開されたビジネス関数を呼び出します。

   以下の例では、プロセス・アプリケーションでのこのステップを示します。

   ```
   process.initiate("MyProcessModel",input);
   ```

   アプリケーションからの呼び出しは、トランザクションとして実行されます。トランザクションは、以下のいずれかの方法で確立されて終了します。
   - WebSphere Application Server から自動的に (デプロイメント記述子が TX_REQUIRED を指定)。
   - アプリケーションから明示的に。アプリケーションの呼び出しを 1 つのトランザクションにバンドルすることができます。

     ```
     // Obtain user transaction interface
     UserTransaction transaction=
            (UserTransaction)initialContext.lookup("jta/usertransaction");

     // Begin a transaction
     transaction.begin();

     // Applications calls ...

     // On successful return, commit the transaction
     transaction.commit();
     ```

ヒント: データベース・デッドロックを防ぐには、並列トランザクションで以下のようなステートメントの実行を避けるようにします。

```
// Obtain user transaction interface
UserTransaction transaction=
       (UserTransaction)initialContext.lookup("jta/usertransaction");

transaction.begin();

//read lock on the activity instance
process.getActivityInstance(aiid);
//write lock on the activity instance
process.claim(aiid);

transaction.commit();
```

例

以下に、ステップ 3 から 5 でタスク・アプリケーションを探す方法の例を示します。

```
// Obtain the default initial JNDI context
InitialContext initialContext = new InitialContext();

// Lookup the remote home interface of the HumanTaskManager bean
Object result =
         initialContext.lookup("java:comp/env/ejb/HumanTaskManagerHome");

// Convert the lookup result to the proper type
HumanTaskManagerHome taskHome =
           (HumanTaskManagerHome)javax.rmi.PortableRemoteObject.narrow
           (result,HumanTaskManagerHome.class);


...
//Access the remote interface of the session bean.
HumanTaskManager task = taskHome.create();

...
//Call the business functions exposed by the service interface
task.callTask(tkiid,input);
```

ローカル・セッション Bean へのアクセス:

EJB クライアント・アプリケーションは、Bean のホーム・インターフェースを介して、適切なローカル・セッション Bean にアクセスします。

セッション Bean は、プロセス・アプリケーションの場合は LocalBusinessFlowManager セッション Bean、ヒューマン・タスク・アプリケーションの場合は LocalHumanTaskManager セッション Bean になります。

1. ローカル・セッション Bean への参照をアプリケーション・デプロイメント記述子に追加します。 参照を以下のファイルの 1 つに追加します。
   - Java 2 Platform Enterprise Edition (J2EE) クライアント・アプリケーションの場合は、`application-client.xml` ファイル
   - Web アプリケーションの場合は、`web.xml` ファイル
   - Enterprise JavaBeans (EJB) アプリケーションの場合は、`ejb-jar.xml` ファイル

   プロセス・アプリケーションの場合のローカル・ホーム・インターフェースへの参照は、以下の例で示されます。

```
<ejb-local-ref>
 <ejb-ref-name>ejb/LocalBusinessFlowManagerHome</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <local-home>com.ibm.bpe.api.LocalBusinessFlowManagerHome</local-home>
 <local>com.ibm.bpe.api.LocalBusinessFlowManager</local>
</ejb-local-ref>
```

タスク・アプリケーションの場合のローカル・ホーム・インターフェースへの参照は、以下の例で示されます。

```
<ejb-local-ref>
 <ejb-ref-name>ejb/LocalHumanTaskManagerHome</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <local-home>com.ibm.task.api.LocalHumanTaskManagerHome</local-home>
 <local>com.ibm.task.api.LocalHumanTaskManager</local>
</ejb-local-ref>
```

WebSphere Integration Developer を使用して EJB 参照をデプロイメント記述子に追加する場合、EJB 参照のバインディングが、アプリケーションのデプロイ時に自動的に作成されます。EJB 参照の追加について詳しくは、WebSphere Integration Developer の文書を参照してください。

2. Java Naming and Directory Interface (JNDI) からローカル・セッション Bean のローカル・ホーム・インターフェースへの参照を取得します。

   以下の例では、プロセス・アプリケーションでのこのステップを示します。

   ```
   // Obtain the default initial JNDI context
   InitialContext initialContext = new InitialContext();

     // Lookup the local home interface of the LocalBusinessFlowManager bean

      LocalBusinessFlowManagerHome processHome =
           (LocalBusinessFlowManagerHome)initialContext.lookup
           ("java:comp/env/ejb/LocalBusinessFlowManagerHome");
   ```

   ローカル・セッション Bean のホーム・インターフェースには、EJB オブジェクトの create メソッドが含まれます。このメソッドは、セッション Bean のローカル・インターフェースを戻します。

3. ローカル・セッション Bean のローカル・インターフェースにアクセスします。

   以下の例では、プロセス・アプリケーションでのこのステップを示します。

   ```
   LocalBusinessFlowManager process = processHome.create();
   ```

   セッション Bean へのアクセス権は、呼び出し元が Bean が提供するすべてのアクションを実行できることを保証するものではありません。呼び出し元には、そのアクションに対する許可も必要になります。セッション Bean のインスタンスを作成すると、コンテキストとそのセッション Bean のインスタンスが関連付けられます。コンテキストには呼び出し元のプリンシパル ID、グループ・メンバーシップ・リストが含まれ、その呼び出し元に Business Process Choreographer の J2EE ロールのいずれかがあるかどうかを示します。コンテキストは、グローバル・セキュリティーが設定されていなくても、それぞれの呼び出しごとに、呼び出し元の権限を確認するために使用されます。グローバル・セキュリティーが設定されていない場合、呼び出し元のプリンシパル ID の値は UNAUTHENTICATED になります。

4. サービス・インターフェースによって公開されたビジネス関数を呼び出します。

以下の例では、プロセス・アプリケーションでのこのステップを示します。

```
process.initiate("MyProcessModel",input);
```

アプリケーションからの呼び出しは、トランザクションとして実行されます。トランザクションは、以下のいずれかの方法で確立されて終了します。

- WebSphere Application Server から自動的に (デプロイメント記述子が TX_REQUIRED を指定)。
- アプリケーションから明示的に。アプリケーションの呼び出しを 1 つのトランザクションにバンドルすることができます。

```
// Obtain user transaction interface
UserTransaction transaction=
       (UserTransaction)initialContext.lookup("jta/usertransaction");

// Begin a transaction
transaction.begin();

// Applications calls ...

// On successful return, commit the transaction
transaction.commit();
```

ヒント: データベース・デッドロックを防ぐには、並列トランザクションで以下のようなステートメントの実行を避けるようにします。

```
// Obtain user transaction interface
UserTransaction transaction=
       (UserTransaction)initialContext.lookup("jta/usertransaction");

transaction.begin();

//read lock on the activity instance
process.getActivityInstance(aiid);
//write lock on the activity instance
process.claim(aiid);

transaction.commit();
```

例

以下に、ステップ 2 から 4 でタスク・アプリケーションを探す方法の例を示します。

```
// Obtain the default initial JNDI context
InitialContext initialContext = new InitialContext();

// Lookup the local home interface of the LocalHumanTaskManager bean
LocalHumanTaskManagerHome taskHome =
        (LocalHumanTaskManagerHome)initialContext.lookup
        ("java:comp/env/ejb/LocalHumanTaskManagerHome");

...
//Access the local interface of the local session bean
LocalHumanTaskManager task = taskHome.create();

...
//Call the business functions exposed by the service interface
task.callTask(tkiid,input);
```

ビジネス・プロセスおよびタスク関連のオブジェクトの照会

クライアント・アプリケーションは、ビジネス・プロセスとタスク関連オブジェクトを操作します。データベース内のビジネス・プロセス・オブジェクトおよびタスク関連のオブジェクトを照会して、これらのオブジェクトの特定のプロパティーを取得することができます。

Business Process Choreographer を構成すると、リレーショナル・データベースは、ビジネス・プロセス・コンテナーおよびタスク・コンテナーの両方に関連付けられます。このデータベースは、ビジネス・プロセスとタスクの管理用のすべてのテンプレート (モデル) とインスタンス (ランタイム) のデータを保管します。そのデータを照会するには、SQL 形式の構文を使用します。

単発の照会を実行して、オブジェクトの特定のプロパティーを取得することができます。また、頻繁に使用する照会を保管しておいて、この保管照会文をアプリケーションに組み込むこともできます。

ビジネス・プロセスおよびタスク関連オブジェクトに対する照会:

サービス API の query メソッドまたは queryAll メソッドを使用して、ビジネス・プロセスおよびタスクに関する保管情報を取得します。

query メソッドは、呼び出し元の権限に応じてオブジェクトを戻します。照会結果セットには、呼び出し元が関連した作業項目を持つオブジェクトのプロパティーのみが含まれます。 queryAll メソッドは、データベース内のすべてのオブジェクトについて選択されたデータを戻します。

オブジェクトのプロパティーを照会するために、事前定義データベース・ビューが提供されています。

照会は、以下のもので構成されます。
- select 文節
- where 文節
- order-by 文節
- スキップ・タプル・パラメーター
- しきい値パラメーター
- 時間帯パラメーター

照会の構文はオブジェクト・タイプによって異なります。以下の表は、異なるオブジェクト・タイプごとの構文を示しています。

表 2.

オブジェクト	構文
プロセス・テンプレート	`ProcessTemplateData[] queryProcessTemplates (java.lang.String whereClause, java.lang.String orderByClause, java.lang.Integer threshold, java.util.TimeZone timezone);`
タスク・テンプレート	`TaskTemplate[] queryTaskTemplates (java.lang.String whereClause, java.lang.String orderByClause, java.lang.Integer threshold, java.util.TimeZone timezone);`

表 2. (続き)

オブジェクト	構文

ビジネス・プロセス・データおよびタスク関連データ	`QueryResultSet query (java.lang.String selectClause,`
	`java.lang.String whereClause,`
	`java.lang.String orderByClause,`
	`java.lang.Integer skipTuples`
	`java.lang.Integer threshold,`
	`java.util.TimeZone timezone);`

例えば、関数の呼び出し元にアクセス可能な作業項目 ID のリストは、次のようにして取得します。

```
QueryResultSet result = process.query("DISTINCT WORK_ITEM.WIID",
                                      null, null, null, null, null);
```

QUERY インターフェースには、queryAll メソッドも含まれています。このメソッドを使用して、データベースに保管されているすべてのオブジェクトのデータを、モニターなどの目的で検索できます。queryAll メソッドの呼び出し元には、 Java 2 Platform Enterprise Edition (J2EE) ロールの、BPESystemAdministrator、BPESystemMonitor、TaskSystemAdministrator、または TaskSystemMonitor のいずれかが必要です。オブジェクトの対応する作業項目を使用した許可検査は適用されません。

照会には、カスタム・プロパティーと変数プロパティーの両方を含めることができます。複数のカスタム・プロパティーまたは変数プロパティーを照会に含める場合、対応するデータベース表で自己結合が行われます。データベース・システムによっては、これらの `query()` 呼び出しによりパフォーマンスへの影響が出ることがあります。

createStoredQuery メソッドを使用して、照会を Business Process Choreographer データベースに保管することもできます。保管照会文を定義する際には、照会の基準を指定します。基準は保管照会文が実行されるときに動的に適用されます。つまり、データは実行時にアセンブルされます。保管照会文にパラメーターが含まれている場合、照会が実行されるときにこれらのパラメーターも解決されます。

Business Process Choreographer API について詳しくは、プロセス関連メソッドの com.ibm.bpe.api パッケージおよびタスク関連メソッドの com.ibm.task.api パッケージ内にある Javadoc を参照してください。

select 文節:

照会関数の select 文節は、照会によって戻されるオブジェクト・プロパティーを示します。

select 文節は、照会結果を記述します。これは、戻すオブジェクト・プロパティー (結果の列) を識別する名前のリストを指定します。構文は SQL select 文節と同じで、コンマを使用して文節のパーツを分離します。文節の各パーツは、事前定義されているビューのいずれか 1 つのプロパティーを指定する必要があります。QueryResultSet オブジェクトで戻される列は、select 文節で指定されているプロパティーと同じ順序で表示されます。

select 文節は、AVG()、SUM()、MIN()、または MAX() などの SQL 集約関数はサポートしていません。

複数の名前と値のペアのプロパティー (カスタム・プロパティーおよび照会できる変数のプロパティーなど) を選択する場合は、ビュー名に 1 桁のカウンターを追加します。このカウンターは 1 から 9 の値を取ることができます。

select 文節の例

- ″WORK_ITEM.OBJECT_TYPE, WORK_ITEM.REASON″

  関連オブジェクトのオブジェクト・タイプ、および作業項目の割り当て理由を取得します。

- ″DISTINCT WORK_ITEM.OBJECT_ID″

  呼び出し元が作業項目を所有しているオブジェクトの ID すべてを重複なしで取得します。

- ″ACTIVITY.TEMPLATE_NAME, WORK_ITEM.REASON″

  呼び出し元が作業項目を所有しているアクティビティーの名前、およびその割り当て理由を取得します。

- ″ACTIVITY.STATE, PROCESS_INSTANCE.STARTER″

  アクティビティーの状態、およびその関連プロセス・インスタンスのスターターを取得します。

- ″DISTINCT TASK.TKIID, TASK.NAME″

  呼び出し元が作業項目を所有しているタスクの ID と名前すべてを重複なしで取得します。

- ″TASK_CPROP1.STRING_VALUE, TASK_CPROP2.STRING_VALUE″

  さらに where 文節でも指定されているカスタム・プロパティーの値を取得します。

- ″QUERY_PROPERTY1.STRING_VALUE, QUERY_PROPERTY2.INT_VALUE

  照会できる変数のプロパティーの値を取得します。これらの部分は、さらに where 文節でも指定されています。

- ″COUNT(DISTINCT TASK.TKIID)″

  where 文節の条件を満たす固有のタスクの作業項目の数を数えます。

where 文節:

照会関数の中の where 文節は、照会ドメインに適用するフィルター基準を記述します。

where 文節の構文は、SQL の where 文節と同じです。文節から明示的に SQL を追加したり、where 文節に述部を結合したりする必要はありません。これらの構成要素は照会の実行時に自動的に追加されます。フィルター基準を適用しない場合は、where 文節に `null` を指定する必要があります。

where 文節構文は以下のものをサポートします。

- キーワード: AND, OR, NOT
- 比較演算子: =、<=、<、<>、>、>=、LIKE
- 設定操作: IN

  LIKE 操作では、照会されるデータベースに定義されているワイルドカード文字がサポートされます。

以下の規則も適用されます。

- オブジェクト ID 定数を `ID('string-rep-of-oid')` に指定します。
- `BIN('UTF-8 string')` としてバイナリー定数を指定します。
- 整数列挙型の代わりにシンボリック定数を使用します。例えば、アクティビティー状態式 `ACTIVITY.STATE=2` を指定する代わりに、`ACTIVITY.STATE=ACTIVITY.STATE.STATE_READY` を指定します。
- 比較ステートメント内のプロパティーの値に単一引用符 (') が含まれる場合、例えば `"TASK_CPROP.STRING_VALUE='d''automatisation'"` のように、引用符を二重にしてください。
- ビュー名に 1 桁のサフィックスを追加して、複数の名前と値のペアのプロパティー (カスタム・プロパティーなど) を参照します。例:
  `"TASK_CPROP1.NAME='prop1' AND "TASK_CPROP2.NAME='prop2'"`
- タイム・スタンプ定数を `TS('yyyy-mm-ddThh:mm:ss')` に指定します。現在日付を参照するには、タイム・スタンプを `CURRENT_DATE` に指定します。

  タイム・スタンプには、最低でも日付または時間の値を指定する必要があります。
  - 日付のみを指定すると、時間値はゼロに設定されます。
  - 時間のみを指定すると、日付は現在の日付に設定されます。
  - 日付を指定する場合、年は 4 桁の定数で構成する必要があります。月および日の値はオプションです。欠落している月および日の値は、01 に設定されます。例えば、`TS('2003')` は `TS('2003-01-01T00:00:00')` と同じです。
  - 日付を指定する場合、この値は 24 時間制で記述されます。例えば、現在日付が 2003 年 1 月 1 日の場合、`TS('T16:04')` または `TS('16:04')` は、`TS('2003-01-01T16:04:00')` と同じです。

where 文節の例

- オブジェクト ID と既存の ID の比較

  ```
  "WORK_ITEM.WIID = ID('_WI:800c00ed.df8d7e7c.feffff80.38')"
  ```

  この型の where 文節は、通常、直前の呼び出しの既存オブジェクト ID を使用して、動的に作成されます。このオブジェクト ID が *wiid1* 変数に保管されている場合、文節の構文は次のようになります。

  ```
  "WORK_ITEM.WIID = ID('" + wiid1.toString() + "')"
  ```

- タイム・スタンプの使用

  ```
  "ACTIVITY.STARTED >= TS('2002-06-1T16.00.00')"
  ```

- シンボリック定数の使用

```
"WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER"
```

- ブール値 true および false の使用

```
"ACTIVITY.BUSINESS_RELEVANCE = TRUE"
```

- カスタム・プロパティーの使用

```
"TASK_CPROP1.NAME = 'prop1' AND " TASK_CPROP1.STRING_VALUE = 'v1' AND
 TASK_CPROP2.NAME = 'prop2' AND " TASK_CPROP2.STRING_VALUE = 'v2'"
```

order-by 文節:

照会関数内の order-by 文節は、照会結果セットのソート基準を指定します。

order-by 文節の構文は、SQL の order-by 文節と同じで、コンマを使用して文節の各パーツを分離します。文節の各パーツは、事前定義されているビューのいずれか 1 つのプロパティーを指定する必要があります。

ソート基準はサーバーに適用されます。つまり、ソートにサーバーのロケールが使用されます。複数のプロパティーを指定すると、照会結果セットはまず最初のプロパティーの値で順序付けされ、次に 2 番目のプロパティーの値で順序付けされる、という具合に続きます。

照会結果セットをソートしない場合は、order-by 文節で null を指定する必要があります。

order-by 文節の例

- ″PROCESS_TEMPLATE.NAME″

  照会結果を、プロセス・テンプレート名でアルファベット順にソートします。
- ″PROCESS_INSTANCE.CREATED, PROCESS_INSTANCE.NAME DESC″

  照会結果を作成日でソートし、特定の日付の場合はその結果を、プロセス・インスタンス名でアルファベット順の逆順にソートします。
- ″ACTIVITY.OWNER, ACTIVITY_TEMPLATE.NAME, ACTIVITY.STATE″

  照会結果を、アクティビティー所有者、アクティビティー・テンプレート名、アクティビティーの状態の順でソートします。

スキップ・タプル・パラメーター:

スキップ・タプル・パラメーターは、照会結果セットの先頭から数えた、無視して呼び出し元に戻さない、照会結果セット・タプルの数を指定します。

このパラメーターは、しきい値パラメーターと一緒に使用して、(最初に 20 項目を検索し、次に、その次の 20 項目を検索するなどのように、) クライアント・アプリケーションでのページングをインプリメントします。

このパラメーターを null に設定したときに、しきい値パラメーターを設定していないと、すべての適格なタプルが戻されます。

スキップ・タプル・パラメーターの例

- new Integer(5)

  最初の 5 つの適格なタプルを戻さないように指定します。

しきい値パラメーター:

照会関数のしきい値パラメーターは、照会の結果セットとしてサーバーからクライアントに戻されるオブジェクトの数を制限します。

実動シナリオにおける照会結果セットは数千から数百万の項目を含む場合さえあるため、常にしきい値を指定することがベスト・プラクティスとなります。しきい値パラメーターは、例えば、同時に少数の項目のみが表示されるグラフィカル・ユーザー・インターフェースなどで有用な場合があります。しきい値パラメーターを適宜設定すると、データベース照会が高速化し、サーバーからクライアントへ転送する必要のあるデータが少なくなります。

このパラメーターを `null` に設定したときに、スキップ・タプル・パラメーターを設定していないと、適格なオブジェクトがすべて戻されます。

しきい値パラメーターの例

- new Integer(50)

  50 個の適格なタプルを戻すように指定します。

時間帯パラメーター:

照会関数の時間帯パラメーターは、照会内のタイム・スタンプ定数の時間帯を定義します。

照会を開始するクライアントと照会を処理するサーバーの間で、時間帯が異なることがあります。時間帯パラメーターを使用して、where 文節で使用されるタイム・スタンプ定数の時間帯を、例えば地方時を指定するように指定します。照会の結果セットで戻される日付には、照会で指定したものと同じ時間帯が設定されます。

このパラメーターを `null` に設定すると、タイム・スタンプ定数は協定世界時 (UTC) と想定されます。

時間帯パラメーターの例

- ```
  process.query("ACTIVITY.AIID",
                "ACTIVITY.STARTED > TS('2005-01-01T17:40')",
                 null,
                 null,
                 java.util.TimeZone.getDefault() );
  ```

  2005 年 1 月 1 日の 17:40 地方時より後に開始されたアクティビティーのオブジェクト ID を戻します。
- ```
  process.query("ACTIVITY.AIID",
                "ACTIVITY.STARTED > TS('2005-01-01T17:40')",
                 null, null, null);
  ```

  2005 年 1 月 1 日の 17:40 UTC より後に開始されたアクティビティーのオブジェクト ID を戻します。この指定は、例えば東部標準時より 6 時間早い時間です。

保管照会文のパラメーター:

保管照会文は、データベースに保管され、名前で識別される照会のことです。照会が実行されると、修飾するタプルが動的にアセンブルされます。保管照会文を再使用可能にするために、実行時に解決されるパラメーターを照会定義で使用できます。

例えば、顧客名を保管するカスタム・プロパティーを定義したとします。特定の顧客 ACME Co. に関連したタスクを戻すように、照会を定義することができます。この情報を照会する場合、照会内の where 文節は以下の例のようになります。

```
String whereClause =
  "TASK.STATE = TASK.STATE.STATE_READY
  AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER
  AND TASK_CPROP.NAME = 'company' AND TASK_CPROP.STRING_VALUE = 'ACME Co.'";
```

顧客 BCME Ltd. も検索できるようにこの照会を再使用可能にするには、カスタム・プロパティーの値に対してパラメーターを使用できます。パラメーターをタスク照会に追加する場合、以下の例のようになります。

```
String whereClause =
  "TASK.STATE = TASK.STATE.STATE_READY
  AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER
  AND TASK_CPROP.NAME = 'company' AND TASK_CPROP.STRING_VALUE = '@param1'";
```

`@param1` パラメーターは、query メソッドに受け渡されるパラメーターのリストから、実行時に解決されます。以下の規則は、照会内でのパラメーターの使用に適用されます。

- パラメーターは where 文節でのみ使用できる。
- パラメーターはストリングである。
- パラメーターは実行時にストリングの置換を使用して置き換えられる。特殊文字が必要な場合、where 文節に指定するか、実行時にパラメーターの一部として受け渡す必要があります。
- パラメーター名は、ストリング `@param` を整数と連結したもので構成される。最小番号は 1 で、実行時に照会 API に受け渡されるパラメーターのリスト内の最初の項目を指します。
- パラメーターは where 文節内で複数回使用できます。出現するパラメーターはすべて同じ値で置き換えられます。

照会結果:

照会結果セットには、照会の結果が入ります。

結果セットのエレメントは、呼び出し元が表示を許可されているオブジェクトです。 エレメントは、次のメソッドを使用して相対的に読み取るか、あるいは最初と最後のメソッドを使用して絶対的に読み取ります。照会結果セットの暗黙カーソルは初めは最初のエレメントの前に配置されるため、エレメントを読み取る前に、最初または次のメソッドのいずれかを呼び出す必要があります。 size メソッドを使用して、セット内のエレメント数を判別することができます。

照会結果セットのエレメントは、作業項目とそれに関連する参照オブジェクト (アクティビティー・インスタンスやプロセス・インスタンスなど) の選択済み属性を構成します。 QueryResultSet エレメントの最初の属性 (列) は、照会要求の select

文節で指定されている最初の属性の値を指定します。 QueryResultSet エレメントの 2 番目の属性 (列) は、照会要求の select 文節で指定されている 2 番目の属性の値を指定する、という具合に続きます。

属性の値は、その属性タイプと互換性のあるメソッドを呼び出すことによって、また、適切な列インデックスを指定することによって、取得することができます。列インデックスの番号付けは 1 から始まります。

属性タイプ	メソッド
ストリング	getString
OID	getOID
タイム・スタンプ	getTimestamp getString
整数	getInteger getShort getLong getString getBoolean
ブール	getBoolean getShort getInteger getLong getString
byte[]	getBinary

例:

以下の照会が実行されます。

```
QueryResultSet resultSet = process.query("ACTIVITY.STARTED,
                                          ACTIVITY.TEMPLATE_NAME AS NAME,
                                          WORK_ITEM.WIID, WORK_ITEM.REASON",
                                          null, null, null, null);
```

戻される照会結果セットには、以下の 4 つの列があります。
- 列 1 はタイム・スタンプ
- 列 2 はストリング
- 列 3 はオブジェクト ID
- 列 4 は整数

以下のメソッドを使用して、属性値を取得することができます。

```
while (resultSet.next())
{
 java.util.Calendar activityStarted = resultSet.getTimestamp(1);
 String templateName = resultSet.getString(2);
 WIID wiid = (WIID) resultSet.getOID(3);
 Integer reason = resultSet.getInteger(4);
}
```

結果セットの表示名を、例えば、印刷されるテーブルの見出しなどに使用することができます。これらの名前は、ビューの列名、または照会の AS 文節で定義された名前です。以下のメソッドを使用して、例中の表示名を取得することができます。

```
resultSet.getColumnDisplayName(1) returns "STARTED"
resultSet.getColumnDisplayName(2) returns "NAME"
resultSet.getColumnDisplayName(3) returns "WIID"
resultSet.getColumnDisplayName(4) returns "REASON"
```

ビジネス・プロセス・オブジェクトおよびヒューマン・タスク・オブジェクトの照会のための事前定義ビュー:

ビジネス・プロセス・オブジェクトおよびヒューマン・タスク・オブジェクト用に、事前定義データベース・ビューが提供されています。これらのオブジェクトの参照データを照会する場合は、これらのビューを使用します。

定義済みビューを使用する場合は、ビューの列用に明示的に結合述部を追加する必要はありません。これらの構成要素は自動的に追加されます。このデータを照会するには、サービス API (BusinessFlowManagerService または HumanTaskManagerService) の汎用照会関数を使用することができます。HumanTaskManagerDelegate API の対応するメソッド、または ExecutableQuery インターフェースのインプリメンテーションによって提供される定義済み照会を使用することもできます。

ACTIVITY ビュー:

この定義済みデータベース・ビューは、アクティビティーの照会に使用します。

表 3. ACTIVITY ビュー内の列

列名	タイプ	コメント
PIID	ID	プロセス・インスタンス ID。
AIID	ID	アクティビティー・インスタンス ID。
PTID	ID	プロセス・テンプレート ID。
ATID	ID	アクティビティー・テンプレート ID。

表 3. ACTIVITY ビュー内の列 (続き)

列名	タイプ	コメント
KIND	整数	アクティビティーの種類。指定可能な値は、以下のとおりです。 KIND_INVOKE (21) KIND_RECEIVE (23) KIND_REPLY (24) KIND_THROW (25) KIND_RETHROW (46) KIND_TERMINATE (26) KIND_WAIT (27) KIND_COMPENSATE (29) KIND_SEQUENCE (30) KIND_EMPTY (3) KIND_SWITCH (32) KIND_WHILE (34) KIND_PICK (36) KIND_FLOW (38) KIND_SCOPE (40) KIND_SCRIPT (42) KIND_STAFF (43) KIND_ASSIGN (44) KIND_CUSTOM (45) KIND_FOR_EACH_PARALLEL (49) KIND_FOR_EACH_SERIAL (47)
COMPLETED	タイム・スタンプ	アクティビティーの完了時刻。
ACTIVATED	タイム・スタンプ	アクティビティーが活動化された時刻。
FIRST_ACTIVATED	タイム・スタンプ	そのアクティビティーが初めて活動化された時刻。
STARTED	タイム・スタンプ	アクティビティーの開始時刻。

表 3. ACTIVITY ビュー内の列 (続き)

列名	タイプ	コメント
STATE	整数	アクティビティーの状態。指定可能な値は、以下のとおりです。 STATE_INACTIVE (1) STATE_READY (2) STATE_RUNNING (3) STATE_PROCESSING_UNDO (14) STATE_SKIPPED (4) STATE_FINISHED (5) STATE_FAILED (6) STATE_TERMINATED (7) STATE_CLAIMED (8) STATE_TERMINATING (9) STATE_FAILING (10) STATE_WAITING (11) STATE_EXPIRED (12) STATE_STOPPED (13)
OWNER	ストリング	所有者のプリンシパル ID。
DESCRIPTION	ストリング	アクティビティー・テンプレートの説明にプレースホルダーが含まれている場合、この列には、解決済みのプレースホルダーを所有するアクティビティー・インスタンスの説明が入ります。
TEMPLATE_NAME	ストリング	関連するアクティビティー・テンプレートの名前。
TEMPLATE_DESCR	ストリング	関連するアクティビティー・テンプレートの説明。
BUSINESS_RELEVANCE	ブール	アクティビティーがビジネスと関係があるかどうかを指定します。指定可能な値は、以下のとおりです。 **TRUE** アクティビティーはビジネスに関係があります。Business Process Choreographer Explorer でアクティビティー状況を表示できます。 **FALSE** アクティビティーはビジネスに関係がありません。
EXPIRES	タイム・スタンプ	アクティビティーの期限が切れる日時。アクティビティーの期限が切れている場合は、このイベントが発生したときの日時。

ACTIVITY_ATTRIBUTE ビュー:

この定義済みデータベース・ビューは、アクティビティーのカスタム・プロパティーの照会に使用します。

表 4. ACTIVITY_ATTRIBUTE ビュー内の列

列名	タイプ	コメント
AIID	ID	カスタム・プロパティーを所有するアクティビティー・インスタンスの ID。
NAME	ストリング	カスタム・プロパティーの名前。
VALUE	ストリング	カスタム・プロパティーの値。

ACTIVITY_SERVICE ビュー:

この定義済みデータベース・ビューは、アクティビティー・サービスの照会に使用します。

表 5. ACTIVITY_SERVICE ビュー内の列

列名	タイプ	コメント
EIID	ID	イベント・インスタンスの ID。
AIID	ID	イベントを待機しているアクティビティーの ID。
PIID	ID	イベントが含まれているプロセス・インスタンスの ID。
VTID	ID	イベントを説明するサービス・テンプレートの ID。
PORT_TYPE	ストリング	ポート・タイプの名前。
NAME_SPACE_URI	ストリング	ネーム・スペースの URI。
OPERATION	ストリング	サービスのオペレーション名。

APPLICATION_COMP ビュー:

この定義済みデータベース・ビューは、アプリケーション・コンポーネント ID、およびタスクのデフォルト設定の照会に使用します。

表 6. APPLICATION_COMP ビュー内の列

列名	タイプ	コメント
ACOID	ストリング	アプリケーション・コンポーネントの ID。
BUSINESS_ RELEVANCE	ブール	コンポーネントのデフォルトのタスク・ビジネス関連ポリシー。この値は、タスク・テンプレートまたはタスクの定義によって上書きされる場合があります。属性は、監査証跡へのロギングに影響します。指定可能な値は、以下のとおりです。 **TRUE** タスクはビジネスと関係があり、監査されます。 **FALSE** タスクはビジネスとの関係がなく、監査されません。
NAME	ストリング	アプリケーション・コンポーネントの名前。

表 6. *APPLICATION_COMP ビュー内の列 (続き)*

列名	タイプ	コメント
SUPPORT_ AUTOCLAIM	ブール	コンポーネントのデフォルトの自動要求ポリシー。この属性が TRUE に設定されている場合、潜在的な所有者が単一のユーザーであれば、タスクを自動的に要求することができます。この値は、タスク・テンプレートまたはタスクの定義によって上書きされる場合があります。
SUPPORT_CLAIM_ SUSP	ブール	中断されているタスクを要求できるかどうかを判断するコンポーネントのデフォルト設定。この属性が TRUE に設定されている場合は、中断されているタスクを要求することができます。この値は、タスク・テンプレートまたはタスクの定義によって上書きされる場合があります。
SUPPORT_ DELEGATION	ブール	コンポーネントのデフォルトのタスク代行ポリシー。この属性が TRUE に設定されている場合は、タスクの作業項目割り当てを変更することができます。すなわち、作業項目の作成、削除、または転送が可能です。
SUPPORT_ FOLLOW_ON	ブール	コンポーネントのデフォルトの後続タスク・ポリシー。この属性が TRUE に設定されている場合は、タスクの後続タスクを作成できます。この値は、タスク・テンプレートまたはタスクの定義によって上書きされる場合があります。
SUPPORT_ SUB_TASK	ブール	コンポーネントのデフォルトのサブタスク・ポリシー。この属性が TRUE に設定されている場合は、タスクのサブタスクを作成できます。この値は、タスク・テンプレートまたはタスクの定義によって上書きされる場合があります。

ESCALATION ビュー:

この定義済みデータベース・ビューは、エスカレーションのデータの照会に使用します。

表 7. *ESCALATION ビュー内の列*

列名	タイプ	コメント
ESIID	ストリング	エスカレーション・インスタンスの ID。
ACTION	整数	エスカレーションによって起動されるアクション。指定可能な値は、以下のとおりです。 **ACTION_CREATE_WORK_ITEM (1)** それぞれのエスカレーションに対する受信側の作業項目を作成します。 **ACTION_SEND_EMAIL (2)** それぞれのエスカレーションの受信側に E メールを送信します。 **ACTION_CREATE_EVENT (3)** イベントを作成および公表します。

表 7. ESCALATION ビュー内の列 (続き)

列名	タイプ	コメント
ACTIVATION_STATE	整数	対応するタスクが以下のいずれかの状態になると、エスカレーション・インスタンスが作成されます。 **ACTIVATION_STATE_READY (2)** ヒューマン・タスクまたは参加タスクは、要求を受ける準備ができていることを示します。 **ACTIVATION_STATE_RUNNING (3)** 親タスクが開始され、実行中であることを示します。 **ACTIVATION_STATE_CLAIMED (8)** タスクが要求されることを明示します。 **ACTIVATION_STATE_WAITING_ FOR_SUBTASK (20)** タスクがサブタスクの完了を待つことを示します。
ACTIVATION_TIME	タイム・スタンプ	エスカレーションが活動化される時刻。
AT_LEAST_ EXP_STATE	整数	エスカレーションによって予期されるタスクの状態。タイムアウトが発生した場合、タスク状態がこの属性の値と比較されます。指定可能な値は、以下のとおりです。 **AT_LEAST_EXPECTED_STATE_CLAIMED (8)** タスクが要求されることを明示します。 **AT_LEAST_EXPECTED_STATE_ENDED (20)** タスクが最終状態 (FINISHED、FAILED、TERMINATED、または EXPIRED) にあることを明示します。 **AT_LEAST_EXPECTED_STATE_ SUBTASKS_COMPLETED (21)** タスクのすべてのサブタスクが完了していることを示します。
ESTID	ストリング	対応するエスカレーション・テンプレートの ID。
FIRST_ESIID	ストリング	チェーン内の最初のエスカレーションの ID。
INCREASE_PRIORITY	整数	タスクの優先度を増やす方法を示します。指定可能な値は、以下のとおりです。 **INCREASE_PRIORITY_NO (1)** タスクの優先度は増えません。 **INCREASE_PRIORITY_ONCE (2)** タスクの優先度は 1 度に 1 ずつ増えます。 **INCREASE_PRIORITY_REPEATED (3)** タスクの優先度は、エスカレーションが繰り返されるごとに 1 ずつ増えます。
NAME	ストリング	エスカレーションの名前。

表 7. ESCALATION ビュー内の列 (続き)

列名	タイプ	コメント
STATE	整数	エスカレーションの状態。指定可能な値は、以下のとおりです。 STATE_INACTIVE (1) STATE_WAITING (2) STATE_ESCALATED (3) STATE_SUPERFLUOUS (4)
TKIID	ストリング	エスカレーションが所属するタスク・インスタンス ID。

ESCALATION_CPROP ビュー:

この定義済みデータベース・ビューは、エスカレーションのカスタム・プロパティーを照会するために使用します。

表 8. ESCALATION_CPROP ビュー内の列

列名	タイプ	コメント
ESIID	ストリング	エスカレーション ID。
NAME	ストリング	プロパティーの名前。
DATA_TYPE	ストリング	非ストリング・カスタム・プロパティーのクラスのタイプ。
STRING_VALUE	ストリング	String 型のカスタム・プロパティーの値。

ESCALATION_DESC ビュー:

この定義済みデータベース・ビューは、エスカレーションのマルチリンガル記述データを照会するために使用します。

表 9. ESCALATION_DESC ビュー内の列

列名	タイプ	コメント
ESIID	ストリング	エスカレーション ID。
LOCALE	ストリング	説明または表示名に関連付けられているロケールの名前。
DESCRIPTION	ストリング	タスク・テンプレートの説明。
DISPLAY_NAME	ストリング	エスカレーションの記述名。

PROCESS_ATTRIBUTE ビュー:

この定義済みデータベース・ビューは、プロセスのカスタム・プロパティーの照会に使用します。

表 10. PROCESS_ATTRIBUTE ビュー内の列

列名	タイプ	コメント
PIID	ID	カスタム・プロパティーを所有するプロセス・インスタンスの ID。

表 10. PROCESS_ATTRIBUTE ビュー内の列 (続き)

列名	タイプ	コメント
NAME	ストリング	カスタム・プロパティーの名前。
VALUE	ストリング	カスタム・プロパティーの値。

PROCESS_INSTANCE ビュー:

この定義済みデータベース・ビューは、プロセス・インスタンスの照会に使用します。

表 11. PROCESS_INSTANCE ビュー内の列

列名	タイプ	コメント
PTID	ID	プロセス・テンプレート ID。
PIID	ID	プロセス・インスタンス ID。
NAME	ストリング	プロセス・インスタンスの名前。
STATE	整数	プロセス・インスタンスの状態。指定可能な値は、以下のとおりです。 STATE_READY (1) STATE_RUNNING (2) STATE_FINISHED (3) STATE_COMPENSATING (4) STATE_INDOUBT (10) STATE_FAILED (5) STATE_TERMINATED (6) STATE_COMPENSATED (7) STATE_COMPENSATION_FAILED (12) STATE_TERMINATING (8) STATE_FAILING (9) STATE_SUSPENDED (11)
CREATED	タイム・スタンプ	プロセス・インスタンスの作成時刻。
STARTED	タイム・スタンプ	プロセス・インスタンスの開始時刻。
COMPLETED	タイム・スタンプ	プロセス・インスタンスの完了時刻。
PARENT_NAME	ストリング	親プロセス・インスタンスの名前。
TOP_LEVEL_NAME	ストリング	トップレベル・プロセス・インスタンスの名前。トップレベルのプロセス・インスタンスがない場合、これは、現行プロセス・インスタンスの名前になります。
STARTER	ストリング	プロセス・インスタンスのスターターのプリンシパル ID。
DESCRIPTION	ストリング	プロセス・テンプレートの説明にプレースホルダーが含まれている場合、この列には、解決済みのプレースホルダーを所有するプロセス・インスタンスの説明が入ります。
TEMPLATE_NAME	ストリング	関連するプロセス・テンプレートの名前。

表 11. PROCESS_INSTANCE ビュー内の列 (続き)

列名	タイプ	コメント
TEMPLATE_DESCR	ストリング	関連するプロセス・テンプレートの説明。

PROCESS_TEMPLATE ビュー:

この定義済みデータベース・ビューは、プロセス・テンプレートの照会に使用します。

表 12. PROCESS_TEMPLATE ビュー内の列

列名	タイプ	コメント
PTID	ID	プロセス・テンプレート ID。
NAME	ストリング	プロセス・テンプレートの名前。
VALID_FROM	タイム・スタンプ	プロセス・テンプレートのインスタンス化が可能になる時刻。
TARGET_NAMESPACE	ストリング	プロセス・テンプレートのターゲット・ネーム・スペース。
APPLICATION_NAME	ストリング	プロセス・テンプレートが所属するエンタープライズ・アプリケーションの名前。
VERSION	ストリング	ユーザー定義のバージョン。
CREATED	タイム・スタンプ	プロセス・テンプレートがデータベース内に作成される時刻。
STATE	整数	プロセス・インスタンスの作成にプロセス・テンプレートを使用できるかどうかを指定します。指定可能な値は、以下のとおりです。 STATE_STARTED (1) STATE_STOPPED (2)
EXECUTION_MODE	整数	このプロセス・テンプレートから派生したプロセス・インスタンスの実行方法を指定します。指定可能な値は、以下のとおりです。 EXECUTION_MODE_MICROFLOW (1) EXECUTION_MODE_LONG_RUNNING (2)
DESCRIPTION	ストリング	プロセス・テンプレートの説明。
COMP_SPHERE	整数	プロセス・テンプレート内の microflow のインスタンスの補正の振る舞いを指定します。既存の補正範囲を結合するか、補正範囲を作成するかのいずれかです。 指定可能な値は、以下のとおりです。 COMP_SPHERE_REQUIRED (2) COMP_SPHERE_REQUIRES_NEW (3) COMP_SPHERE_SUPPORTS (4) COMP_SPHERE_NOT_SUPPORTED (1)

QUERY_PROPERTY ビュー:

この定義済みデータベース・ビューは、プロセス・レベル変数の照会に使用します。

表 13. QUERY_PROPERTY ビュー内の列

列名	タイプ	コメント
PIID	ID	プロセス・インスタンス ID。
VARIABLE_NAME	ストリング	プロセス・レベル変数の名前。
NAME	ストリング	照会プロパティーの名前。
NAMESPACE	ストリング	照会プロパティーのネームスペース。
GENERIC_VALUE	ストリング	次のいずれかの定義済みタイプにマップしないプロパティー・タイプのストリング表記。STRING_VALUE、NUMBER_VALUE、DECIMAL_VALUE、または TIMESTAMP_VALUE。
STRING_VALUE	ストリング	プロパティー・タイプがストリング・タイプにマップされる場合、これがストリングの値です。
NUMBER_VALUE	整数	プロパティー・タイプが整数タイプにマップされる場合、これが整数の値です。
DECIMAL_VALUE	10 進数	プロパティー・タイプが浮動小数点タイプにマップされる場合、これが 10 進数の値です。
TIMESTAMP_VALUE	タイム・スタンプ	プロパティー・タイプがタイム・スタンプ・タイプにマップされる場合、これがタイム・スタンプの値です。

TASK ビュー:

この定義済みデータベース・ビューは、タスク・オブジェクトの照会に使用します。

表 14. TASK ビュー内の列

列名	タイプ	コメント
TKIID	ID	タスク・インスタンスの ID。
ACTIVATED	タイム・スタンプ	タスクが活動化された時刻。
APPLIC_ DEFAULTS_ID	ID	タスクのデフォルト値を指定するアプリケーション・コンポーネントの ID。
APPLIC_NAME	ストリング	タスクが所属するエンタープライズ・アプリケーションの名前。

表 14. TASK ビュー内の列 (続き)

列名	タイプ	コメント
BUSINESS_ RELEVANCE	ブール	タスクがビジネスと関係があるかどうかを指定します。 属性は、監査証跡へのロギングに影響します。指定可能な値は、以下のとおりです。 **TRUE** タスクはビジネスと関係があり、監査されます。 **FALSE** タスクはビジネスとの関係がなく、監査されません。
COMPLETED	タイム・スタンプ	タスクが完了した時刻。
CONTAINMENT_ CTX_ID	ID	このタスクの包含コンテキスト。この属性は、タスクのライフ・サイクルを決定します。タスクの包含コンテキストを削除すると、タスク・テンプレートも削除されます。
CTX_ AUTHORIZATION	整数	タスクの所有者がタスク・コンテキストにアクセスできるようにします。指定可能な値は、以下のとおりです。 **AUTH_NONE** 関連コンテキスト・オブジェクトに対する許可権限はありません。 **AUTH_READER** 関連コンテキスト・オブジェクトに関する操作に、プロセス・インスタンスのプロパティーの読み取りなどのリーダー権限が必要です。
DUE	タイム・スタンプ	タスクの期限時刻。
EXPIRES	タイム・スタンプ	タスクの有効期限が切れる日付。
FIRST_ACTIVATED	タイム・スタンプ	タスクが初めて活動化された時刻。
FOLLOW_ON_TKIID	ID	後続のタスクのインスタンスの ID。
HIERARCHY_ POSITION	整数	指定可能な値は、以下のとおりです。 **HIERARCHY_POSITION_TOP_TASK (0)** タスク階層の最上位タスク。 **HIERARCHY_POSITION_SUB_TASK (1)** タスクはタスク階層のサブタスクです。 **HIERARCHY_POSITION_FOLLOW_ON_TASK (2)** タスクはタスク階層の後続タスクです。
IS_AD_HOC	ブール	このタスクが実行時に動的に作成されたのか、またはタスク・テンプレートから作成されたのかを示します。
IS_ESCALATED	ブール	このタスクのエスカレーションが発生済みかどうかを示します。

表 14. TASK ビュー内の列 (続き)

列名	タイプ	コメント
IS_INLINE	ブール	タスクがビジネス・プロセス内のインラインのタスクであるかどうかを示します。
IS_WAIT_FOR_SUB_TK	ブール	親タスクが、サブタスクが終了状態になるのを待機しているかどうかを示します。
KIND	整数	タスクの種類。指定可能な値は、以下のとおりです。 **KIND_HUMAN (101)** タスクが人の手で作成され、処理されることを示します。 **KIND_WPC_STAFF_ACTIVITY (102)** タスクが、WebSphere Business Integration Server Foundation バージョン 5 ビジネス・プロセスのスタッフ・アクティビティーであるヒューマン・タスクであることを示します。 **KIND_ORIGINATING (103)** 人からコンピューターへの対話をタスクがサポートし、人がサービスを作成、開始、および始動できることを示します。 **KIND_PARTICIPATING (105)** コンピューターから人への対話をタスクがサポート、人がサービスをインプリメントできることを示します。 **KIND_ADMINISTRATIVE (106)** タスクが管理用タスクであることを示します。
LAST_MODIFIED	タイム・スタンプ	タスクの最終変更時刻。
LAST_STATE_CHANGE	タイム・スタンプ	タスクの状態の最終変更時刻。
NAME	ストリング	タスクの名前。
NAME_SPACE	ストリング	タスクのカテゴリー化に使用されるネーム・スペース。
ORIGINATOR	ストリング	タスク・オリジネーターのプリンシパル ID。
OWNER	ストリング	タスクの所有者のプリンシパル ID。
PARENT_CONTEXT_ID	ストリング	このタスクの親コンテキスト。この属性は、呼び出し側アプリケーション・コンポーネント内の対応するコンテキストに対するキーを提供します。親コンテキストは、そのタスクを作成するアプリケーション・コンポーネントによって設定されます。
PRIORITY	整数	タスクの優先順位。
STARTED	タイム・スタンプ	タスクが開始された時刻 (STATE_RUNNING、STATE_CLAIMED)。
STARTER	ストリング	タスク・スターターのプリンシパル ID。

表 14. TASK ビュー内の列 (続き)

列名	タイプ	コメント
STATE	整数	タスクの状態。指定可能な値は、以下のとおりです。 **STATE_READY (2)** そのタスクは要求を受ける準備ができたことを示します。 **STATE_RUNNING (3)** タスクが開始され、実行中であることを示します。 **STATE_FINISHED (5)** タスクが正常に完了したことを示します。 **STATE_FAILED (6)** タスクが正常に完了しなかったことを示します。 **STATE_TERMINATED (7)** 外部要求または内部要求が原因で、タスクが終了したことを示します。 **STATE_CLAIMED (8)** タスクが要求されることを示します。 **STATE_EXPIRED (12)** 指定された期間を超えたため、タスクが終了したことを示します。 **STATE_FORWARDED (101)** タスクが完了して、後続タスクがあることを示します。
SUPPORT_ AUTOCLAIM	ブール	このタスクが単一ユーザーに割り当てられている場合に、自動的に要求されるかどうかを示します。
SUPPORT_CLAIM_ SUSP	ブール	このタスクが中断されている場合に、要求可能かどうかを示します。
SUPPORT_ DELEGATION	ブール	このタスクが、作業項目の作成、削除、または転送によって、作業代行をサポートするかどうかを示します。
SUPPORT_ FOLLOW_ON	ブール	このタスクが後続タスクの作成をサポートするかどうかを示します。
SUPPORT_SUB_TASK	ブール	このタスクがサブタスクの作成をサポートするかどうかを示します。
SUSPENDED	ブール	タスクが中断されているかどうかを示します。
TKTID	ID	タスク・テンプレート ID。
TOP_TKIID	ID	これがサブタスクの場合、親タスク上位インスタンス ID。
TYPE	ストリング	タスクのカテゴリー化に使用されるタイプ。

TASK_CPROP ビュー:

この定義済みデータベース・ビューは、タスク・オブジェクトのカスタム・プロパティーを照会するために使用します。

表 15. TASK_CPROP ビュー内の列

列名	タイプ	コメント
TKIID	ストリング	タスク・インスタンス ID。
NAME	ストリング	プロパティーの名前。
STRING_VALUE	ストリング	String 型のカスタム・プロパティーの値。

TASK_DESC ビュー:

この定義済みデータベース・ビューは、タスク・オブジェクトのマルチリンガル記述データを照会するために使用します。

表 16. TASK_DESC ビュー内の列

列名	タイプ	コメント
TKIID	ストリング	タスク・インスタンス ID。
LOCALE	ストリング	説明または表示名に関連付けられているロケールの名前。
DESCRIPTION	ストリング	タスクの説明。
DISPLAY_NAME	ストリング	タスクの記述名。

TASK_TEMPL ビュー:

この定義済みデータベース・ビューは、タスクのインスタンスを生成するために使用できるデータを保持します。

表 17. TASK_TEMPL ビュー内の列

列名	タイプ	コメント
TKTID	ストリング	タスク・テンプレート ID。
VALID_FROM	タイム・スタンプ	インスタンス化にタスク・テンプレートが使用できるようになる時刻。
APPLIC_ DEFAULTS_ID	ストリング	タスク・テンプレートのデフォルト値を指定するアプリケーション・コンポーネントの ID。
APPLIC_NAME	ストリング	タスク・テンプレートが所属するエンタープライズ・アプリケーションの名前。
BUSINESS_ RELEVANCE	ブール	タスク・テンプレートがビジネスと関係があるかどうかを指定します。 属性は、監査証跡へのロギングに影響します。指定可能な値は、以下のとおりです。 **TRUE** タスクはビジネスと関係があり、監査されます。 **FALSE** タスクはビジネスとの関係がなく、監査されません。

表 17. TASK_TEMPL ビュー内の列 (続き)

列名	タイプ	コメント
CONTAINMENT_ CTX_ID	ID	このタスク・テンプレートの包含コンテキスト。この属性は、タスク・テンプレートのライフ・サイクルを決定します。包含コンテキストを削除すると、タスク・テンプレートも削除されます。
CTX_ AUTHORIZATION	整数	タスクの所有者がタスク・コンテキストにアクセスできるようにします。指定可能な値は、以下のとおりです。 **AUTH_NONE** 関連コンテキスト・オブジェクトに対する許可権限はありません。 **AUTH_READER** 関連コンテキスト・オブジェクトに関する操作に、プロセス・インスタンスのプロパティーの読み取りなどのリーダー権限が必要です。
IS_AD_HOC	ブール	このタスク・テンプレートが、実行時に動的に作成されたのか、またはタスクが EAR ファイルの一部としてデプロイされたときに作成されたのかを示します。
IS_INLINE	ブール	このタスク・テンプレートがビジネス・プロセス内のタスクとしてモデル化されるかどうかを示します。
KIND	整数	このタスク・テンプレートから派生したタスクの種類。指定可能な値は、以下のとおりです。 **KIND_HUMAN (101)** タスクが人の手で作成され、処理されることを明示します。 **KIND_ORIGINATING (103)** 人によるコンピューターへのタスクの割り当てが可能であることを示します。この場合、人間が自動サービスを呼び出します。 **KIND_PARTICIPATING (105)** サービス・コンポーネント (ビジネス・プロセスなど) により、人にタスクが割り当てられることを示します。 **KIND_ADMINISTRATIVE (106)** タスクが管理用タスクであることを明示します。
NAME	ストリング	タスク・テンプレートの名前。
NAMESPACE	ストリング	タスク・テンプレートのカテゴリー化に使用されるネーム・スペース。
PRIORITY	整数	タスク・テンプレートの優先順位。

表 17. TASK_TEMPL ビュー内の列 (続き)

列名	タイプ	コメント
STATE	整数	タスク・テンプレートの状態。指定可能な値は、以下のとおりです。 **STATE_STARTED (1)** タスク・テンプレートをタスク・インスタンスの作成に使用できることを明示します。 **STATE_STOPPED (2)** タスク・テンプレートが停止されたことを明示します。この状態のタスク・テンプレートからタスク・インスタンスを作成することはできません。
SUPPORT_ AUTOCLAIM	ブール	このタスク・テンプレートから派生したタスクが 1 人のユーザーに割り当てられた場合、タスクを自動的に要求できるかどうかを示します。
SUPPORT_CLAIM_ SUSP	ブール	このタスク・テンプレートから派生したタスクが中断された場合、タスクを自動的に要求できるかどうかを示します。
SUPPORT_ DELEGATION	ブール	このタスク・テンプレートから派生したタスクが、作業項目の作成、削除、または転送を使用して作業代行をサポートするかどうかを示します。
SUPPORT_ FOLLOW_ON	ブール	タスク・テンプレートが後続タスクの作成をサポートするかどうかを示します。
SUPPORT_SUB_TASK	ブール	タスク・テンプレートがサブタスクの作成をサポートするかどうかを示します。
TYPE	ストリング	タスク・テンプレートのカテゴリー化に使用されるタイプ。

TASK_TEMPL_CPROP ビュー:

この定義済みデータベース・ビューは、タスク・テンプレートのカスタム・プロパティーを照会するために使用します。

表 18. TASK_TEMPL_CPROP ビュー内の列

列名	タイプ	コメント
TKTID	ストリング	タスク・テンプレート ID。
NAME	ストリング	プロパティーの名前。
DATA_TYPE	ストリング	非ストリング・カスタム・プロパティーのクラスのタイプ。
STRING_VALUE	ストリング	String 型のカスタム・プロパティーの値。

TASK_TEMPL_DESC ビュー:

この定義済みデータベース・ビューは、タスク・テンプレート・オブジェクトのマルチリンガル記述データを照会するために使用します。

表 19. TASK_TEMPL_DESC ビュー内の列

列名	タイプ	コメント
TKTID	ストリング	タスク・テンプレート ID。
LOCALE	ストリング	説明または表示名に関連付けられているロケールの名前。
DESCRIPTION	ストリング	タスク・テンプレートの説明。
DISPLAY_NAME	ストリング	タスク・テンプレートの記述名。

WORK_ITEM ビュー:

この定義済みデータベース・ビューは、作業項目の照会や、プロセス、タスクおよびエスカレーション用の許可データの照会に使用します。

表 20. WORK_ITEM ビュー内の列

列名	タイプ	コメント
WIID	ID	作業項目 ID。
OWNER_ID	ストリング	所有者のプリンシパル ID。
GROUP_NAME	ストリング	関連するグループ・ワーク・リストの名前。
EVERYBODY	ブール	この作業項目を全員が所有するかどうかを指定します。

表 20. WORK_ITEM ビュー内の列 (続き)

列名	タイプ	コメント
OBJECT_TYPE	整数	関連オブジェクトのタイプ。指定可能な値は、以下のとおりです。 **OBJECT_TYPE_ACTIVITY (1)** その作業項目がアクティビティー用に作成されたものであることを明示します。 **OBJECT_TYPE_PROCESS_INSTANCE (3)** その作業項目がプロセス・インスタンス用に作成されたものであることを明示します。 **OBJECT_TYPE_TASK_INSTANCE (5)** その作業項目がタスク用に作成されたものであることを明示します。 **OBJECT_TYPE_TASK_TEMPLATE (6)** その作業項目がタスク・テンプレート用に作成されたものであることを明示します。 **OBJECT_TYPE_ESCALATION_ INSTANCE (7)** その作業項目がエスカレーション・インスタンス用に作成されたものであることを明示します。 **OBJECT_TYPE_APPLICATION_ COMPONENT (9)** その作業項目がアプリケーション・コンポーネント用に作成されたものであることを示します。
OBJECT_ID	ID	関連オブジェクト (例えば、関連プロセスやタスクなど) の ID。
ASSOC_OBJECT_TYPE	整数	ASSOC_OID 属性によって参照されるオブジェクトのタイプ。例えば、タスク、プロセス、または外部オブジェクトなど。OBJECT_TYPE 属性の値を使用します。
ASSOC_OID	ID	作業項目のあるオブジェクト関連オブジェクトの ID。例えば、この作業項目が作成されたアクティビティー・インスタンスを含んでいるプロセス・インスタンスの、プロセス・インスタンス ID など。

表 20. *WORK_ITEM* ビュー内の列 (続き)

列名	タイプ	コメント
REASON	整数	作業項目の割り当て理由。指定可能な値は、以下のとおりです。 REASON_POTENTIAL_STARTER (5) REASON_POTENTIAL_INSTANCE_ CREATOR (11) REASON_POTENTIAL_STARTER (1) REASON_EDITOR (2) REASON_READER (3) REASON_ORIGINATOR (9) REASON_OWNER (4) REASON_STARTER (6) REASON_ESCALATION_RECEIVER (10) REASON_ADMINISTRATOR (7)
CREATION_TIME	タイム・スタンプ	作業項目が作成された日時。

照会で変数を使用することによるデータのフィルタリング:

照会結果は、照会基準に一致するオブジェクトを戻します。この結果を、変数の値でフィルタリングすることもできます。

実行時にプロセスが使用する変数を、そのプロセス・モデルで定義することができます。これらの変数で、照会可能なパートを宣言します。

例えば、John Smith が保険会社のサービス番号を呼び出して、損傷を受けた車に対する保険請求の進捗状況を問い合わせるとします。請求の管理者はカスタマー ID でその請求を検索します。

1. **オプション:** プロセス内の照会可能な変数のプロパティーをリストします。

   プロセス・テンプレート ID を使用して、プロセスを特定します。照会可能な変数がわかっている場合は、このステップはスキップしてください。

   ```
   List variableProperties = process.getQueryProperties(ptid);
   for (int i = 0; i < variableProperties.size(); i++)
   {
      QueryProperty queryData = (QueryProperty)variableProperties.get(i);
      String variableName = queryData.getVariableName();
      String name         = queryData.getName();
      int mappedType      = queryData.getMappedType();
      ...
   }
   ```

2. フィルター基準に一致する変数を持つプロセス・インスタンスをリストします。

   このプロセスでは、カスタマー ID は変数 customerClaim の一部としてモデル化され、照会可能です。そのため、カスタマー ID を使用すれば問題の請求を見つけることができます。

```
QueryResultSet result = process.query
    ("PROCESS_INSTANCE.NAME, QUERY_PROPERTY.STRING_VALUE",
     "QUERY_PROPERTY.VARIABLE_NAME = 'customerClaim' AND " +
     "QUERY_PROPERTY.NAME = 'customerID' AND " +
     "QUERY_PROPERTY.STRING_VALUE like 'Smith%'",
          null, null, null, null );
```

このアクションによって戻される照会結果セットには、プロセス・インスタンス名と、ID が Smith で始まる顧客のカスタマー ID の値が含まれています。

保管照会文の管理:

保管照会文は、頻繁に実行される照会を保管する方法を提供します。この保管照会文は、すべてのユーザーに対して使用できる照会 (共通照会) にすることも、特定のユーザーに属する照会 (専用照会) にすることもできます。

保管照会文は、データベースに保管され、名前で識別される照会のことです。専用の保管照会文と共通の保管照会文の名前を同じにすることができます。異なる複数の所有者の専用保管照会文を同じ名前にすることもできます。

保管照会文は、ビジネス・プロセス・オブジェクト、タスク・オブジェクト、またはこの 2 つのオブジェクト・タイプの組み合わせたものを対象とします。

共通保管照会文の管理:

共通保管照会文はシステム管理者によって作成されます。この照会は、全ユーザーが使用できます。

システム管理者は、共通保管照会文を作成、表示、および削除できます。API 呼び出しでユーザー ID を指定しないと、保管照会文は共通保管照会文であると想定されます。

1. 共通の保管照会文を作成します。

   例えば、以下のコードの断片は、プロセス・インスタンスの保管照会文を作成して、それを CustomerOrdersStartingWithA の名前を付けて保管します。

   ```
   process.createStoredQuery("CustomerOrdersStartingWithA",
               "DISTINCT PROCESS_INSTANCE.PIID, PROCESS_INSTANCE.NAME",
               "PROCESS_INSTANCE.NAME LIKE 'A%'",
               "PROCESS_INSTANCE.NAME",
                null,null);
   ```

   保管照会文による照会結果は、文字 A で始まるすべてのプロセス・インスタンス名、および関連したプロセス・インスタンス ID (PIID) をソートしたリストになります。

2. 保管照会文で定義された照会を実行します。

   ```
   QueryResultSet result = process.query("CustomerOrdersStartingWithA",
                new Integer(0));
   ```

   このアクションにより、基準を満たすオブジェクトが戻されます。この場合は、A で始まる顧客オーダー。

3. 使用可能な共通保管照会文の名前をリストします。

   以下のコードの断片では、戻される照会のリストを共通照会のみに限定する方法を示しています。

```
String[] storedQuery = process.getStoredQueryNames(StoredQueryData.KIND_PUBLIC);
```

4. **オプション:** 特定の保管照会文で定義された照会を検査します。

   専用の保管照会文には、共通の保管照会文と同じ名前を付けることができます。名前が同じである場合は、専用の保管照会文が戻されます。以下のコードの断片では、指定した名前の共通照会のみを戻す方法を示しています。タスク・ベース・オブジェクトでこの照会を実行する場合は、戻されるオブジェクト・タイプとして、StoredQueryData ではなく StoredQuery を指定してください。

```
StoredQueryData storedQuery = process.getStoredQuery
   (StoredQueryData.KIND_PUBLIC, "CustomerOrdersStartingWithA");
String selectClause = storedQuery.getSelectClause();
String whereClause = storedQuery.getWhereClause();
String orderByClause = storedQuery.getOrderByClause();
Integer threshold = storedQuery.getThreshold();String owner = storedQuery.getOwner();
```

5. 共通の保管照会文を削除します。

   以下のコードの断片では、ステップ 1 で作成した保管照会文の削除方法を示しています。

```
process.deleteStoredQuery("CustomerOrdersStartingWithA");
```

他のユーザーの専用保管照会文の管理:

専用照会はどのユーザーでも作成できます。この照会は、照会の所有者とシステム管理者しか使用できません。

システム管理者は、特定ユーザーに属する専用の保管照会文を管理できます。

1. ユーザー ID Smith の専用保管照会文を作成します。

   例えば、以下のコードの断片は、プロセス・インスタンスの保管照会文を作成して、それをユーザー ID Smith 用に CustomerOrdersStartingWithA の名前を付けて保管します。

```
process.createStoredQuery("Smith", "CustomerOrdersStartingWithA",
             "DISTINCT PROCESS_INSTANCE.PIID, PROCESS_INSTANCE.NAME",
             "PROCESS_INSTANCE.NAME LIKE 'A%'",
             "PROCESS_INSTANCE.NAME",
              null, null, null, null);
```

   保管照会文による照会結果は、文字 A で始まるすべてのプロセス・インスタンス名、および関連したプロセス・インスタンス ID (PIID) をソートしたリストになります。

2. 保管照会文で定義された照会を実行します。

```
QueryResultSet result = process.query("Smith", "CustomerOrdersStartingWithA",
                                      null,null, null, null);
               new Integer(0));
```

   このアクションにより、基準を満たすオブジェクトが戻されます。この場合は、A で始まる顧客オーダー。

3. 特定のユーザーに属する専用照会の名前のリストを取得します。

   例えば、以下のコードの断片では、ユーザー Smith に属する専用照会のリストを取得する方法を示しています。

```
String[] storedQuery = process.getStoredQueryNames("Smith");
```

4. 特定の照会の詳細を表示します。

以下のコードの断片では、ユーザー Smith が所有する照会 CustomerOrdersStartingWithA の詳細を表示する方法を示しています。

```
StoredQuery storedQuery = process.getStoredQuery
   ("Smith", "CustomerOrdersStartingWithA");
String selectClause = storedQuery.getSelectClause();
String whereClause = storedQuery.getWhereClause();
String orderByClause = storedQuery.getOrderByClause();
Integer threshold = storedQuery.getThreshold();String owner = storedQuery.getOwner();
```

5. 専用の保管照会文を削除します。

   以下のコードの断片では、ユーザー Smith が所有する専用照会を削除する方法を示しています。

   ```
   process.deleteStoredQuery("Smith", "CustomerOrdersStartingWithA");
   ```

専用保管照会文の操作:

システム管理者でなくても、自分専用の保管照会文は作成、実行、および削除できます。また、システム管理者が作成した共通の保管照会文を使用することもできます。

1. 専用の保管照会文を作成します。

   例えば、以下のコードの断片は、プロセス・インスタンスの保管照会文を作成して、固有の名前を付けて保管します。ユーザー ID を指定しないと、保管照会文はログオン・ユーザーの専用保管照会文であると想定されます。

   ```
   process.createStoredQuery("CustomerOrdersStartingWithA",
                "DISTINCT PROCESS_INSTANCE.PIID, PROCESS_INSTANCE.NAME",
                "PROCESS_INSTANCE.NAME LIKE 'A%'",
                "PROCESS_INSTANCE.NAME",
                 null,null);
   ```

   この照会は、文字 A で始まるプロセス・インスタンス名、および関連したプロセス・インスタンス ID (PIID) をすべてソートしたリストにして戻します。

2. 保管照会文で定義された照会を実行します。

   ```
   QueryResultSet result = process.query("CustomerOrdersStartingWithA",
                  new Integer(0));
   ```

   このアクションにより、基準を満たすオブジェクトが戻されます。この場合は、A で始まる顧客オーダー。

3. ログオン・ユーザーがアクセスできる保管照会文の名前のリストを取得します。

   以下のコードの断片では、ユーザーがアクセスできる共通の保管照会文と専用の保管照会文の両方を取得する方法を示しています。

   ```
   String[] storedQuery = process.getStoredQueryNames();
   ```

4. 特定の照会の詳細を表示します。

   以下のコードの断片では、ユーザー Smith が所有する照会 CustomerOrdersStartingWithA の詳細を表示する方法を示しています。

   ```
   StoredQuery storedQuery = process.getStoredQuery
      ("CustomerOrdersStartingWithA");
   String selectClause = storedQuery.getSelectClause();
   String whereClause = storedQuery.getWhereClause();
   String orderByClause = storedQuery.getOrderByClause();
   Integer threshold = storedQuery.getThreshold();String owner = storedQuery.getOwner();
   ```

5. 専用の保管照会文を削除します。

以下のコードの断片は、専用の保管照会文の削除方法を示します。

```
process.deleteStoredQuery("CustomerOrdersStartingWithA");
```

ビジネス・プロセス用のアプリケーションの開発

ビジネス・プロセスは、ビジネス・ゴールを達成するために特定のシーケンスで呼び出される、ビジネス関連の一連のアクティビティーです。プロセスに対する標準のアクションに対応したアプリケーションを開発する方法を示した例が提供されています。

ビジネス・プロセスは、microflow または長期にわたって実行するプロセスのいずれかです。

- microflow は短期で実行するビジネス・プロセスで、同期で実行されます。結果は即時に呼び出し元に戻されます。
- 長期実行の割り込み可能プロセスは、まとめてチェーニングされるアクティビティーのシーケンスとして実行されます。プロセス内で特定の構造を使用すると、プロセス・フローに割り込みが発生します (例えば、ヒューマン・タスクの呼び出し、同期バインディングを使用したサービスの呼び出し、またはタイマー駆動型アクティビティーの使用など)。

  プロセスの並列分岐は、通常非同期でナビゲートされます。すなわち、並列分岐のアクティビティーは並行して実行されます。アクティビティーのタイプとトランザクションの設定に応じて、アクティビティーを独自のトランザクションで実行することができます。

ビジネス・プロセスのための許可のロール:

ビジネス・プロセスで実行できるアクションは、ご使用の許可のロールに依存します。このロールは、J2EE のロールまたはインスタンス・ベースのロールとすることができます。

ロールとは、同じ権限レベルを共用する従業員の集合です。Java 2 Platform, Enterprise Edition (J2EE) のロールは、ビジネス・プロセス・コンテナーが構成されたときにセットアップされます。インスタンス・ベースのロールは、プロセスがモデル化されたときにプロセスおよびアクティビティーに割り当てられます。ロール・ベースの許可については、グローバル・セキュリティーが WebSphere Application Server で使用可能になっていることが必要です。

J2EE のロール

以下の J2EE のロールがサポートされています。

- J2EE BPESystemAdministrator。このロールを割り当てられたユーザーは、すべての特権を持ちます。また、このロールは、ビジネス・プロセスのシステム管理者とも言われます。
- J2EE BPESystemMonitor。このロールを割り当てられたユーザーは、すべてのビジネス・プロセス・オブジェクトのプロパティーを表示できます。また、このロールは、ビジネス・プロセスのシステム・モニターとも言われます。

これらのロールへのユーザーおよびグループの割り当てを変更するには、管理コンソールを使用できます。

RACF セキュリティーによるロールの設定: この RACF 許可は、以下のセキュリティー・フィールドを指定した場合に適用されます。

- **com.ibm.security.SAF.authorization= true**

```
RDEFINE EJBROLE BPESystemAdministrator  UACC(NONE)
PERMIT BPESystemAdministrator CLASS(EJBROLE)  ID(userid) ACCESS(READ)

RDEFINE EJBROLE BPESystemMonitor  UACC(NONE)
PERMIT BPESystemMonitor  CLASS(EJBROLE)  ID(userid) ACCESS(READ)
```

- **com.ibm.security.SAF.delegation= true**

```
RDEFINE EJBROLE JMSAPIUser  UACC(NONE) APPLDATA(' userid')
```

Security Authorization Facility (SAF) ベースの許可 (RACF EJBROLE プロファイルの使用など) を使用して、ビジネス・プロセス・コンテナーなど、EJB およびエンタープライズ・アプリケーションにおける Java 2 Platform、 Enterprise Edition (J2EE) ロールに対するクライアントのアクセスを制御できます。 SAF の使用についての詳細は、 WebSphere Application Server for z/OS インフォメーション・センターの『役割ベースの許可の System Authorization Facility』を参照してください。

インスタンス・ベースのロール

プロセス・インスタンスまたはアクティビティーはプロセス・モデルではスタッフ・メンバーには直接割り当てられておらず、その代わり、使用可能なロールの 1 つに割り当てられます。インスタンス・ベースのロールに割り当てられたスタッフ・メンバーは、そのロールに応じたアクションを実行できます。インスタンス・ベースのロールとユーザーの関連付けは、スタッフの解決を使用して実行時に決定されます。

以下のインスタンス・ベースのロールがサポートされています。
- プロセスに対して: 読者、スターター、管理者
- アクティビティーに対して: 読者、編集者、可能なスターター、可能な所有者、所有者、管理者

これらのロールは、以下のアクションの実行が許可されています。

ロール	許可されたアクション
アクティビティー・リーダー	関連したアクティビティー・インスタンスのプロパティーおよび入出力メッセージを表示します。
アクティビティー編集者	アクティビティー・リーダーに許可されたアクションと、アクティビティーに関連したメッセージおよびその他のデータへの書き込みアクセス権限。
可能なアクティビティー・スターター	アクティビティー・リーダーに許可されたアクション。このロールのメンバーは、 receive アクティビティーまたは pick アクティビティーにメッセージを送信できます。
可能なアクティビティー所有者	アクティビティー・リーダーに許可されたアクション。このロールのメンバーはアクティビティーを要求できます。
アクティビティー所有者	アクティビティーを処理し、完了します。このロールのメンバーは、所有された作業項目を管理者または可能な所有者に転送できます。
アクティビティー管理者	予期しないエラーで停止したアクティビティーを修理し、長期実行アクティビティーを強制終了します。

ロール	許可されたアクション
プロセス・スターター	関連したプロセス・インスタンスのプロパティーおよび入出力メッセージを表示します。
プロセス・リーダー	関連したプロセス・インスタンスのプロパティーおよび入出力メッセージを表示します。また、プロセス・リーダーは、プロセス・インスタンスに含まれるアクティビティーのプロパティー、および入出力メッセージを表示できますが、そのサブプロセスに関する情報は表示できません。
プロセス管理者	このロールのメンバーは、プロセス・インスタンスを管理し、開始済みプロセスに介入して、作業項目の作成、削除、および転送ができます。このロールのメンバーには、アクティビティー管理者権限もあります。

プロセス・スターターのユーザー ID は、プロセス・インスタンスが存在している場合、ユーザー・レジストリーから削除しないでください。これを行うと、このプロセスのナビゲーションが継続できません。システム・ログ・ファイルから以下の例外を受け取ります。

```
no unique ID for: <user ID>
```

プロセス・インスタンスに対するアクションに必要なロール:

LocalBusinessFlowManager または BusinessFlowManager インターフェースへのアクセス権は、呼び出し元がプロセスに対するすべてのアクションを実行できることは保証しません。呼び出し元は、アクションを実行する許可が与えられているロールを使用して、クライアント・アプリケーションにログオンする必要があります。

次の表に、それぞれのロールで実行できるプロセス・インスタンス上のアクションを示します。

アクション	呼び出し元のプリンシパルのロール		
	リーダー	スターター	管理者
createMessage	x	x	x
createWorkItem			x
delete			x
deleteWorkItem			x
forceTerminate			x
getActiveHandlers	x	x	x
getAllActivities	x		x
getAllWorkItems	x		x
getClientUISettings	x		x
getCustomProperties	x	x	x
getCustomProperty	x	x	x
getCustomPropertyNames	x	x	x
getFaultMessage	x	x	x
getInputClientUISettings	x		x
getInputMessage	x	x	x

アクション	呼び出し元のプリンシパルのロール		
	リーダー	スターター	管理者
getOutputClientUISettings	x		x
getOutputMessage	x	x	x
getProcessInstance	x	x	x
getVariable	x	x	x
getWaitingActivities	x	x	x
getWorkItems	x		x
resume			x
restart			x
setCustomProperty		x	x
setVariable			x
suspend			x
transferWorkItem			x

ビジネス・プロセス・アクティビティーのアクションに必要なロール:

LocalBusinessFlowManager または BusinessFlowManager インターフェースへのアクセス権は、呼び出し元がアクティビティーに対するすべてのアクションを実行できることは保証しません。呼び出し元は、アクションを実行する許可が与えられているロールを使用して、クライアント・アプリケーションにログオンする必要があります。

次の表に、それぞれのロールで実行できるアクティビティー・インスタンス上のアクションを示します。

アクション	呼び出し元のプリンシパルのロール				
	リーダー	エディター	潜在的な所有者	所有者	管理者
cancelClaim				x	x
claim			x		x
complete				x	x
createMessage	x	x	x	x	x
createWorkItem					x
deleteWorkItem					x
forceComplete					x
forceRetry					x
getActivityInstance	x	x	x	x	x
getAllWorkItems	x	x	x	x	x
getClientUISettings	x	x	x	x	x
getCustomProperties	x	x	x	x	x
getCustomProperty	x	x	x	x	x
getCustomPropertyNames	x	x	x	x	x
getFaultMessage	x	x	x	x	x
getFaultNames	x	x	x	x	x

アクション	呼び出し元のプリンシパルのロール				
	リーダー	エディター	潜在的な所有者	所有者	管理者
getInputMessage	x	x	x	x	x
getOutputMessage	x	x	x	x	x
getVariable	x	x	x	x	x
getWorkItems	x	x	x	x	x
setCustomProperty		x		x	x
setFaultMessage		x		x	x
setOutputMessage		x		x	x
setVariable					x
transferWorkItem				x 潜在的な所有者または管理者に対してのみ	x

ビジネス・プロセスのライフ・サイクルの管理:

プロセスを開始できる Business Process Choreographer API メソッドが呼び出されると、プロセス・インスタンスが生成されます。プロセス・インスタンスのすべてのアクティビティーが終了状態になるまで、プロセス・インスタンスのナビゲーションは続きます。プロセス・インスタンスに対してさまざまなアクションを実行し、そのライフ・サイクルを管理することができます。

プロセスに対する以下の標準のライフ・サイクル・アクションに対応したアプリケーションを開発する方法を示した例が提供されています。

ビジネス・プロセスの開始:

ビジネス・プロセスを開始する方法は、プロセスが microflow であるか長期実行プロセスであるかによって異なります。プロセスを開始するサービスも、プロセスの開始方法にとって重要です。プロセスに固有の開始サービスを 1 つ設定するか、複数の開始サービスを設定することができます。

microflow や長期実行プロセスを開始する標準のシナリオに対応したアプリケーションを開発する方法を示した例が提供されています。

固有の開始サービスを含む microflow の実行:

microflow は、receive アクティビティーまたは pick アクティビティーから開始できます。開始サービスが固有であるのは、microflow が receive アクティビティーを使って開始された場合、または pick アクティビティー内に 1 つの onMessage 定義のみがある場合です。

microflow によって要求/応答操作がインプリメントされている場合、つまり、プロセスに応答が入っている場合、call メソッドを使用してそのプロセスを実行し、その呼び出しでパラメーターとしてプロセス・テンプレート名を渡すことができます。

microflow が片方向操作である場合は、sendMessage メソッドを使用してプロセスを実行します。このメソッドは、次の例には含まれていません。

1. **オプション:** プロセス・テンプレートをリストして、実行するプロセスの名前を探します。

   プロセスの名前が既に分かっている場合、このステップはオプションです。

   ```
   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
   ("PROCESS_TEMPLATE.EXECUTION_MODE =
          PROCESS_TEMPLATE.EXECUTION_MODE.EXCECUTION_MODE_MICROFLOW",
    "PROCESS_TEMPLATE.NAME",
     new Integer(50),
     null);
   ```

   結果は名前でソートされます。call メソッドによって開始できるソート済みテンプレートのうちの最初の 50 個を収容した配列が照会から戻されます。

2. 該当するタイプの入力メッセージを使ってプロセスを開始します。

   メッセージを作成する場合、メッセージ・タイプ名を指定して、メッセージ定義が含まれるようにする必要があります。

   ```
   ProcessTemplateData template = processTemplates[0];
   //create a message for the single starting receive activity
   ClientObjectWrapper input = process.createMessage
                              (template.getID(),
                               template.getInputMessageTypeName());
   DataObject myMessage = null;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     //set the strings in the message, for example, a customer name
     myMessage.setString("CustomerName", "Smith");
   }

   //run the process
   ClientObjectWrapper output = process.call(template.getName(), input);
   DataObject myOutput = null;
   if ( output.getObject() != null && output.getObject() instanceof DataObject )
   {
     myOutput  = (DataObject)output.getObject();
     int order = myOutput.getInt("OrderNo");
   }
   ```

   このアクションによって、プロセス・テンプレート CustomerTemplate のインスタンスが作成され、一部の顧客データが受け渡されます。この操作は、プロセスが完了してからでないと戻りません。プロセスの結果 OrderNo が、呼び出し元に戻されます。

非固有の開始サービスを含む microflow の実行:

microflow は、receive アクティビティーまたは pick アクティビティーから開始できます。 microflow が複数の onMessage 定義を含む pick アクティビティーを使用して開始された場合、開始サービスは固有ではありません。

microflow によって要求/応答操作がインプリメントされている場合、つまり、プロセスに応答が入っている場合、call メソッドを使用してそのプロセスを実行し、その呼び出しで開始サービスの ID を渡すことができます。

microflow が片方向操作である場合は、sendMessage メソッドを使用してプロセスを実行します。このメソッドは、次の例には含まれていません。

1. **オプション:** プロセス・テンプレートをリストして、実行するプロセスの名前を探します。

   プロセスの名前が既に分かっている場合、このステップはオプションです。

   ```
   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
   ("PROCESS_TEMPLATE.EXECUTION_MODE =
          PROCESS_TEMPLATE.EXECUTION_MODE.EXCECUTION_MODE_MICROFLOW",
    "PROCESS_TEMPLATE.NAME",
     new Integer(50),
     null);
   ```

   結果は名前でソートされます。microflow として開始できるソート済みテンプレートのうちの最初の 50 個を収容した配列が、照会から戻されます。

2. 呼び出すべき開始サービスを判別します。

   この例では、最初に検出されたテンプレートを使用します。

   ```
   ProcessTemplateData template = processTemplates[0];
   ActivityServiceTemplateData[] startActivities =
           process.getStartActivities(template.getID());
   ```

3. 該当するタイプの入力メッセージを使ってプロセスを開始します。

   メッセージを作成する場合、メッセージ・タイプ名を指定して、メッセージ定義が含まれるようにする必要があります。

   ```
   ActivityServiceTemplateData activity = startActivities[0];
   //create a message for the service to be called
   ClientObjectWrapper input =
         process.createMessage(activity.getServiceTemplateID(),
                               activity.getActivityTemplateID(),
                               activity.getInputMessageTypeName());
   DataObject myMessage = null;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     //set the strings in the message, for example, a customer name
     myMessage.setString("CustomerName", "Smith");
   }
   //run the process
   ClientObjectWrapper output = process.call(activity.getServiceTemplateID(),
                                             activity.getActivityTemplateID(),
                                             input);
   //check the output of the process, for example, an order number
   DataObject myOutput = null;
   if ( output.getObject() != null && output.getObject() instanceof DataObject )
   {
     myOutput  = (DataObject)output.getObject();
     int order = myOutput.getInt("OrderNo");
   }
   ```

   このアクションによって、プロセス・テンプレート CustomerTemplate のインスタンスが作成され、一部の顧客データが受け渡されます。この操作は、プロセスが完了してからでないと戻りません。プロセスの結果 OrderNo が、呼び出し元に戻されます。

固有の開始サービスを含む長期実行プロセスの開始:

開始サービスが固有の場合、initiate メソッドを使用して、プロセス・テンプレート名をパラメーターとして渡すことができます。これは、長期実行プロセスが、単一の receive アクティビティーまたは pick アクティビティーのいずれかを使用して開始する、および単一の pick アクティビティーが 1 つのみの onMessage 定義を持つ場合にあてはまります。

1. **オプション:** プロセス・テンプレートをリストして、開始するプロセスの名前を探します。

   プロセスの名前が既に分かっている場合、このステップはオプションです。

   ```
   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
     ("PROCESS_TEMPLATE.EXECUTION_MODE =
          PROCESS_TEMPLATE.EXECUTION_MODE.EXCECUTION_MODE_LONG_RUNNING",
      "PROCESS_TEMPLATE.NAME",
       new Integer(50),
       null);
   ```

   結果は名前でソートされます。initiate メソッドによって開始できるソート済みテンプレートのうちの最初の 50 個を収容した配列が照会から戻されます。

2. 該当するタイプの入力メッセージを使ってプロセスを開始します。

   メッセージを作成する場合、メッセージ・タイプ名を指定して、メッセージ定義が含まれるようにする必要があります。プロセス・インスタンス名を指定する場合、アンダースコアーで開始しないようにする必要があります。プロセス・インスタンス名が指定されていない場合、ストリング・フォーマットのプロセス・インスタンス ID (PIID) が名前として使用されます。

   ```
   ProcessTemplateData template = processTemplates[0];
   //create a message for the single starting receive activity
   ClientObjectWrapper input = process.createMessage
                               (template.getID(),
                               template.getInputMessageTypeName());
   DataObject myMessage = null;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     //set the strings in the message, for example, a customer name
     myMessage.setString("CustomerName", "Smith");
   }
   //start the process
   PIID piid = process.initiate(template.getName(), "CustomerOrder", input);
   ```

   このアクションによって、インスタンス CustomerOrder が作成され、一部の顧客データが受け渡されます。プロセスが開始されると、新規プロセス・インスタンスのオブジェクト ID を呼び出し元に戻します。

   プロセス・インスタンスのスターターは、要求の呼び出し元に設定されます。このユーザーは、このプロセス・インスタンスの作業項目を受信します。プロセス・インスタンスのプロセス管理者、リーダー、およびエディターが決定され、プロセス・インスタンスの作業項目を受信します。追加のアクティビティー・インスタンスが決定されます。これらは自動的に開始されるか、または staff、receive、pick アクティビティーの場合、作業項目が潜在的な所有者に対して作成されます。

非固有の開始サービスを含む長期実行プロセスの開始:

長期実行プロセスは、複数の開始 receive アクティビティーまたは pick アクティビティーを介して開始することができます。initiate メソッドを使用して、プロセスを開始することができます。例えば、プロセスが複数の receive または pick アクティビティー、または複数の onMessage 定義を持つ pick アクティビティーから開始される場合など、開始サービスが固有のものではない場合、呼び出されるサービスを識別する必要があります。

1. **オプション:** プロセス・テンプレートをリストして、開始するプロセスの名前を探します。

   プロセスの名前が既に分かっている場合、このステップはオプションです。

   ```
   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
     ("PROCESS_TEMPLATE.EXECUTION_MODE =
         PROCESS_TEMPLATE.EXECUTION_MODE.EXCECUTION_MODE_LONG_RUNNING",
      "PROCESS_TEMPLATE.NAME",
       new Integer(50),
       null);
   ```

   結果は名前でソートされます。長期実行プロセスとして開始できるソート済みテンプレートのうちの最初の 50 個を収容した配列が照会から戻されます。

2. 呼び出すべき開始サービスを判別します。

   ```
   ProcessTemplateData template = processTemplates[0];
   ActivityServiceTemplateData[] startActivities =
           process.getStartActivities(template.getID());
   ```

3. 該当するタイプの入力メッセージを使ってプロセスを開始します。

   メッセージを作成する場合、メッセージ・タイプ名を指定して、メッセージ定義が含まれるようにする必要があります。プロセス・インスタンス名を指定する場合、アンダースコアーで開始しないようにする必要があります。プロセス・インスタンス名が指定されていない場合、ストリング・フォーマットのプロセス・インスタンス ID (PIID) が名前として使用されます。

   ```
   ActivityServiceTemplateData activity = startActivities[0];
   //create a message for the service to be called
   ClientObjectWrapper input = process.createMessage
                               (activity.getServiceTemplateID(),
                                activity.getActivityTemplateID(),
                                activity.getInputMessageTypeName());
   DataObject myMessage = null;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     //set the strings in the message, for example, a customer name
     myMessage.setString("CustomerName", "Smith");
   }
   //start the process
   PIID piid = process.sendMessage(activity.getServiceTemplateID(),
                                   activity.getActivityTemplateID(),
                                   input);
   ```

   このアクションによって、インスタンスが作成され、一部の顧客データが受け渡されます。プロセスが開始されると、新規プロセス・インスタンスのオブジェクト ID を呼び出し元に戻します。

   プロセス・インスタンスの開始は、要求の呼び出し元に設定され、プロセス・インスタンスの作業項目を受信します。プロセス・インスタンスのプロセス管理者、リーダー、およびエディターが決定され、プロセス・インスタンスの作業項

目を受信します。追加のアクティビティー・インスタンスが決定されます。これらは自動的に開始されるか、または staff、receive、pick アクティビティーの場合、作業項目が潜在的な所有者に対して作成されます。

ビジネス・プロセスの中断と再開:

長期にわたって実行するトップレベルのプロセス・インスタンスを実行中に中断し、再開して完了することができます。

呼び出し元は、プロセス・インスタンスの管理者、またはビジネス・プロセス管理者でなければなりません。プロセス・インスタンスを中断するには、プロセス・インスタンスが実行状態または失敗状態でなければなりません。

例えば、プロセスで後で使用されるバックエンド・システムへのアクセスを構成するために、プロセス・インスタンスを中断することがあります。プロセスの前提条件を満たしていれば、そのプロセス・インスタンスを再開することができます。プロセスを中断してからプロセス・インスタンスの失敗の原因となっている問題の修正を行い、問題が修正されたら再びプロセスを再開することもできます。

1. 中断する実行中のプロセス CustomerOrder を取得します。

   ```
   ProcessInstanceData processInstance =
                       process.getProcessInstance("CustomerOrder");
   ```

2. プロセス・インスタンスを中断します。

   ```
   PIID piid = processInstance.getID();
   process.suspend( piid );
   ```

   このアクションにより、指定したトップレベルのプロセス・インスタンスが中断します。プロセス・インスタンスは、中断状態になります。 autonomy 属性が child に設定されたサブプロセスも、状態が実行中、失敗、終了、または補正になっていれば中断されます。このプロセス・インスタンスに関連するインライン・タスクも中断されますが、このプロセス・インスタンスに関連するスタンドアロン・タスクは中断されません。

   この状態では、開始されたアクティビティーはまだ完了することはできますが、新規のアクティビティーは活動化されません。例えば、要求済み状態のスタッフ・アクティビティーは完了することができます。

3. プロセス・インスタンスを再開します。

   ```
   process.resume( piid );
   ```

   このアクションにより、プロセス・インスタンスとそのサブプロセスが中断前の状態に戻ります。

ビジネス・プロセスの再開:

完了、終了、失敗、補正のいずれかの状態にあるプロセス・インスタンスを再開させることができます。

呼び出し元は、プロセス・インスタンスの管理者、またはビジネス・プロセス管理者でなければなりません。

プロセス・インスタンスの再開は、プロセス・インスタンスを初めて開始する手順と同様です。ただし、プロセス・インスタンスの再開時には、プロセス・インスタンス ID が認識されているため、インスタンスの入力メッセージが使用可能です。

プロセスに、プロセス・インスタンスを作成可能な複数の receive アクティビティーまたは pick アクティビティー (receive choice アクティビティーとも呼ばれる) が含まれる場合、これらのアクティビティーに属するすべてのメッセージを使用して、プロセス・インスタンスを再始動します。これらのアクティビティーのいずれかが、要求/応答操作をインプリメントする場合、関連する reply アクティビティーがナビゲートされると、応答が再度送信されます。

1. 再開させるプロセスを取得します。

```
ProcessInstanceData processInstance =
                    process.getProcessInstance("CustomerOrder");
```

2. プロセス・インスタンスを再開します。

```
PIID piid = processInstance.getID();
process.restart( piid );
```

このアクションにより、指定されたプロセス・インスタンスが再開されます。

プロセス・インスタンスの終了:

プロセス管理者権限を持つユーザーが、リカバリー不能状態として認識されているトップレベルのプロセス・インスタンスを終了する必要がある場合があります。プロセス・インスタンスは、未解決のサブプロセスやアクティビティーがあってもこれらを待たずに即時に終了するため、プロセス・インスタンスの終了は例外的な場合にのみ行ってください。

1. 終了するプロセス・インスタンスを検索します。

```
ProcessInstanceData processInstance =
       process.getProcessInstance("CustomerOrder");
```

2. プロセス・インスタンスを終了します。

プロセス・インスタンスを終了する場合、補正を使用してプロセス・インスタンスを終了することも、補正を使用せずに終了することもできます。

補正を使用してプロセス・インスタンスを終了するには、以下のようにします。

```
PIID piid = processInstance.getID();
process.forceTerminate(piid, CompensationBehaviour.INVOKE_COMPENSATION);
```

補正を使用しないでプロセス・インスタンスを終了するには、以下のようにします。

```
PIID piid = processInstance.getID();
process.forceTerminate(piid);
```

プロセス・インスタンスを補正を使用して終了する場合、プロセス・テンプレートに定義された補正ハンドラーが呼び出されます。プロセス・テンプレートに補正ハンドラーが定義されていない場合は、デフォルトの補正ハンドラーが呼び出されます。補正を使用せずにプロセス・インスタンスを終了する場合、プロセス・インスタンスはアクティビティー、参加タスク、インライン親タスクが正常に終了するのを待たずに、即時に終了されます。

プロセスおよびプロセスに関連するスタンドアロン・タスクによって開始されるアプリケーションは、強制終了要求によって終了されません。 そのようなアプリケーションを終了させる場合は、プロセスによって開始されるアプリケーションを明示的に終了するステートメントをプロセス・アプリケーションに追加する必要があります。

プロセス・インスタンスの削除:

完了済みのプロセス・インスタンスは、プロセス・モデル内のプロセス・テンプレートに対応するプロパティーが設定されていれば、Business Process Choreographer データベースから自動的に削除されます。監査ログに書き込まれていないプロセス・インスタンスのデータを照会する場合など、プロセス・インスタンスをデータベースに保存しておきたい場合があります。しかし、プロセス・インスタンス・データを格納しておくと、ディスク・スペースやパフォーマンスに影響が出るだけでなく、同じ相関セット値を使用するプロセス・インスタンスが作成されなくなります。そのため、データベースからプロセス・インスタンス・データを定期的に削除する必要があります。

プロセス・インスタンスを削除するには、プロセス管理者権限が必要であり、そのプロセス・インスタンスは、トップレベルのプロセス・インスタンスでなければなりません。

以下の例では、完了したプロセス・インスタンスをすべて削除する方法が示されています。

1. 完了したプロセス・インスタンスをリストします。

```
QueryResultSet result =
     process.query("DISTINCT PROCESS_INSTANCE.PIID",
                   "PROCESS_INSTANCE.STATE =
                           PROCESS_INSTANCE.STATE.STATE_FINISHED",
                    null, null, null);
```

   このアクションは、完了したプロセス・インスタンスをリストした照会結果セットを戻します。

2. 完了したプロセス・インスタンスを削除します。

```
while (result.next() )
{
   PIID piid = (PIID) result.getOID(1);
   process.delete(piid);
}
```

   このアクションは、選択されたプロセス・インスタンスおよび、そのインライン・タスクをデータベースから削除します。

スタッフ・アクティビティーの処理:

ビジネス・プロセス内のスタッフ・アクティビティーは、作業項目を通じて、組織内のさまざまな人に割り当てられます。プロセスが開始されると、潜在的な所有者に対して作業項目が作成されます。

潜在的な所有者がアクティビティーを要求します。このユーザーは、関係のある情報の提供とアクティビティーの完了に対して責任があります。

1. 作業の準備ができている、ログオン・ユーザーに属するアクティビティーをリストします。

```
QueryResultSet result =
     process.query("ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_READY AND
                    ACTIVITY.KIND = ACTIVITY.KIND.KIND_STAFF AND
                    WORK_ITEM.REASON =
                         WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                    null, null, null);
```

   このアクションは、ログオン・ユーザーが作業することができるアクティビティーが含まれる照会結果セットを戻します。

2. 作業対象のアクティビティーを要求します。

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ClientObjectWrapper input = process.claim(aiid);
 DataObject activityInput = null ;
  if ( input.getObject()!= null && input.getObject() instanceof DataObject )
  {
    activityInput = (DataObject)input.getObject();
    // read the values
    ...
  }
}
```

   アクティビティーが要求されると、アクティビティーの入力メッセージが戻されます。

3. アクティビティーの作業が終了したら、アクティビティーを完了します。 アクティビティーは、正常に完了することも、障害メッセージが表示されて完了することもあります。アクティビティーが正常に完了した場合、出力メッセージが渡されます。アクティビティーが失敗した場合、アクティビティーは失敗状態または停止状態に置かれ、障害メッセージが渡されます。これらのアクションに対して、適切なメッセージを作成する必要があります。メッセージを作成する場合、メッセージ・タイプ名を指定して、メッセージ定義が含まれるようにする必要があります。

   a. アクティビティーを正常に完了するには、出力メッセージを作成します。

```
ActivityInstanceData activity = process.getActivityInstance(aiid);
ClientObjectWrapper output =
      process.createMessage(aiid, activity.getOutputMessageTypeName());
DataObject myMessage = null ;
if ( output.getObject()!= null && output.getObject() instanceof DataObject )
{
  myMessage = (DataObject)output.getObject();
  //set the parts in your message, for example, an order number
  myMessage.setInt("OrderNo", 4711);
}

//complete the activity
process.complete(aiid, output);
```

      このアクションは、オーダー番号が含まれる出力メッセージを設定します。

   b. 障害が発生した場合にアクティビティーを完了するには、障害メッセージを作成します。

```
//retrieve the faults modeled for the staff activity
List faultNames = process.getFaultNames(aiid);

//create a message of the appropriate type
ClientObjectWrapper myFault =
      process.createMessage(aiid, faultNames.get(0));

// set the parts in your fault message, for example, an error number
DataObject myMessage = null ;
if ( myFault.getObject()!= null && input.getObject() instanceof DataObject )
{
  myMessage = (DataObject)myFault.getObject();
  //set the parts in the message, for example, a customer name
  myMessage.setInt("error",1304);
}

process.complete(aiid, (String)faultNames.get(0), myFault);
```

このアクションは、アクティビティーを失敗状態または停止状態のいずれかに設定します。プロセス・モデル内のアクティビティーの **continueOnError** パラメーターが真に設定されている場合、アクティビティーは失敗状態に置かれ、ナビゲーションが続行されます。 **continueOnError** パラメーターが false に設定され、障害が周囲の有効範囲で catch されない場合、そのアクティビティーは停止状態になります。この状態では、強制終了または強制再試行を使用してアクティビティーを修復できます。

1 ユーザーのワークフローの処理:

ワークフローの中には、1 人のユーザーだけで実行されるものがあります。例えば、オンライン・ブックストアでの本の注文などです。このタイプのワークフローには、並列パスは存在しません。 completeAndClaimSuccessor API は、このタイプのワークフローの処理をサポートします。

オンライン・ブックストアでは、購入者は一連の操作を完了することで本を注文します。この一連の操作は、スタッフ・アクティビティー (参加タスク) としてインプリメントできます。購入者が複数の書籍を注文する場合は、これが次のスタッフ・アクティビティーの要求に相当します。このタイプのワークフローは、ページ・フローとも呼ばれます。ユーザー・インターフェース定義が、ユーザー・インターフェースのダイアログのフローを制御するアクティビティーと関連付けられているためです。

completeAndClaimSuccessor API はスタッフ・アクティビティーを完了し、ログオン・ユーザーの同じプロセス・インスタンスで次のアクティビティーを要求します。そして、次に要求したアクティビティーの情報 (処理される入力メッセージなど) を戻します。次のアクティビティーは、完了したアクティビティーと同じトランザクション内で使用可能になるため、トランザクション境界がプロセス・モデルで participates に設定される必要があります。

1. アクティビティー・シーケンスで最初のアクティビティーを要求します。

```
//
//Query the list of activities that can be claimed by the logged-on user
//
QueryResultSet result =
     process.query("ACTIVITY.AIID",
                   "PROCESS_INSTANCE.NAME = 'CustomerOrder' AND
                    ACTIVITY.STATE = ACTIVITY.STATE.STATE_READY AND
                    ACTIVITY.KIND = ACTIVITY.KIND.KIND_STAFF AND
```

```
                    WORK_ITEM.REASON =
                         WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                    null, null, null);
...
//
//Claim the first activity
//
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ClientObjectWrapper input = process.claim(aiid);
 DataObject activityInput = null ;
  if ( input.getObject()!= null && input.getObject() instanceof DataObject )
  {
    activityInput = (DataObject)input.getObject();
    // read the values
    ...
  }
}
```

アクティビティーが要求されると、アクティビティーの入力メッセージが戻されます。

2. アクティビティーの作業が終了したら、そのアクティビティーを完了して次のアクティビティーを要求します。

   アクティビティーを完了するには、出力メッセージを渡します。出力メッセージを作成する場合、メッセージ・タイプ名を指定して、メッセージ定義が含まれるようにする必要があります。

```
ActivityInstanceData activity = process.getActivityInstance(aiid);
ClientObjectWrapper output =
      process.createMessage(aiid, activity.getOutputMessageTypeName());
DataObject myMessage = null ;
if ( output.getObject()!= null && output.getObject() instanceof DataObject )
{
  myMessage = (DataObject)output.getObject();
  //set the parts in your message, for example, an order number
  myMessage.setInt("OrderNo", 4711);
}

//complete the activity and claim the next one
CompleteAndClaimSuccessorResult successor =
   process.completeAndClaimSuccessor(aiid, output);
```

   このアクションにより、アイテム番号を含む出力メッセージが設定され、シーケンス内の次のアクティビティーが要求されます。後続アクティビティーに AutoClaim が設定されている場合は、ランダムなアクティビティーが次のアクティビティーとして戻されます。このユーザーに割り当てられる後続アクティビティーがもうない場合は、Null が戻されます。後続アクティビティーに AutoClaim が設定されていて、その後に続くパスが複数ある場合は、後続アクティビティーのすべてが要求され、ランダムなアクティビティーが次のアクティビティーとして戻されます。

   後に続くことができる並列パスがプロセスに含まれ、これらのパスに、ログオン・ユーザーが潜在的な所有者であるスタッフ・アクティビティーが複数含まれる場合、ランダムなアクティビティーが自動的に要求され、次のアクティビティーとして戻されます。

3. 次のアクティビティーを処理します。

```
String name = successor.getActivityName();

ClientObjectWrapper nextInput = successor.getInputMessage();
if ( nextInput.getObject()!=
               null && nextInput.getObject() instanceof DataObject )
{
  activityInput = (DataObject)input.getObject();
  // read the values
  ...
}
```

4. アクティビティーを完了する場合は、ステップ 2 に進みます。

待機中のアクティビティーへのメッセージの送信:

インバウンド・メッセージ・アクティビティー (receive アクティビティー、pick アクティビティーの onMessage、イベント・ハンドラーの onEvent) を使用して、実行中のプロセスを「外の世界」からのイベントと同期化することができます。例えば、情報に対する要求に応えたお客様からの E メールの受信は、このようなイベントとみなされます。

親タスクを使用して、メッセージをアクティビティーに送信することができます。

1. 特定のプロセス・インスタンス ID を持つプロセス・インスタンスのログオン・ユーザーからのメッセージを待機するアクティビティー・サービス・テンプレートをリストします。

   ```
   ActivityServiceTemplateData[] services = process.getWaitingActivities(piid);
   ```

2. 最初の待機サービスへメッセージを送信します。

   ここでは、最初のサービスが使用したいサービスであると想定しています。呼び出し元は、メッセージを受信するアクティビティーの潜在的なスターター、またはプロセス・インスタンスの管理者である必要があります。

   ```
   VTID vtid = services[0].getServiceTemplateID();
   ATID atid = services[0].getActivityTemplateID();
   String inputType = services[0].getInputMessageTypeName();

   // create a message for the service to be called
     ClientObjectWrapper message =
           process.createMessage(vtid,atid,inputMessageTypeName);
     DataObject myMessage = null;
     if ( message.getObject()!= null && message.getObject() instanceof DataObject )
     {
       myMessage = (DataObject)message.getObject();
       //set the strings in the message, for example, chocolate is to be ordered
       myMessage.setString("Order", "chocolate");
     }

     // send the message to the waiting activity
     process.sendMessage(vtid, atid, message);
   }
   ```

   このアクションによって、指定されたメッセージを待機アクティビティー・サービスに送信し、一部のオーダー・データを渡します。

   また、プロセス・インスタンス ID を指定して、メッセージが指定されたプロセス・インスタンスに送信されたことを確認することもできます。プロセス・インスタンス ID が指定されていない場合、メッセージは、アクティビティー・サービス、およびメッセージの相関値によって識別されたプロセス・インスタンスに

送信されます。プロセス・インスタンス ID が指定された場合、相関値を使用して検出されたプロセス・インスタンスがチェックされ、指定されたプロセス・インスタンス ID であることが確認されます。

イベントの処理:

ビジネス・プロセス全体とビジネス・プロセスの各スコープを、関連するイベントの発生時に呼び出されるイベント・ハンドラーと関連付けることができます。 プロセスによりイベント・ハンドラーを使用して、Web サービス操作を提供できるという点で、イベント・ハンドラーは、receive アクティビティーや pick アクティビティーと似ています。

イベント・ハンドラーは、対応するスコープが実行中である限り、何度でも呼び出すことができます。また、イベント・ハンドラーの複数インスタンスを並行して活動化することができます。

以下のコードの断片では、あるプロセス・インスタンス用のアクティブなイベント・ハンドラーを取得する方法、および入力メッセージを送信する方法を示しています。

1. プロセス・インスタンス ID のデータを判別し、そのプロセスのアクティブなイベント・ハンドラーをリストします。

```
ProcessInstanceData processInstance =
       process.getProcessInstance( "CustomerOrder2711");
EventHandlerTemplateData[] events = process.getActiveEventHandlers(
                                      processInstance.getID() );
```

2. 入力メッセージを送信します。

   この例では、最初に検出されたイベント・ハンドラーを使用します。

```
EventHandlerTemplateData event = null;
if ( events.length > 0 )
{
    event = events[0];

    // create a message for the service to be called
    ClientObjectWrapper input = process.createMessage(
    event.getID(), event.getInputMessageTypeName());

    if (input.getObject() != null && input.getObject() instanceof DataObject )
    {
       DataObject inputMessage = (DataObject)input.getObject();
       // set content of the message, for example, a customer name, order number
       inputMessage.setString("CustomerName", "Smith");
       inputMessage.setString("OrderNo", "2711");

        // send the message
       process.sendMessage( event.getProcessTemplateName(),
                            event.getPortTypeNamespace(),
                            event.getPortTypeName(),
                            event.getOperationName(),
     input );
     }
 }
```

   このアクションにより、指定されたメッセージがプロセスのアクティブなイベント・ハンドラーに送信されます。

プロセスの結果の分析:

プロセスは、Web サービス記述言語 (WSDL) の片方向操作、または要求/応答操作としてモデル化される Web サービス操作を公開できます。長時間実行プロセスが片方向操作を公開する場合、そのプロセスの結果 (プロセス変数の値など) はデータベースから取得する必要があります。

プロセスの結果は、プロセス・インスタンスが派生したプロセス・テンプレートが、派生したプロセス・インスタンスの自動削除を指定しない場合にのみ、データベースに保管されます。

プロセスの結果を分析し、例えば、オーダー番号などを確認します。

```
QueryResultSet result = process.query
                       ("PROCESS_INSTANCE.PIID",
                        "PROCESS_INSTANCE.NAME = 'CustomerOrder' AND
                         PROCESS_INSTANCE.STATE =
                                 PROCESS_INSTANCE.STATE.STATE_FINISHED",
                         null, null, null);
if (result.size() > 0)
{
  result.first();
  PIID piid = (PIID) result.getOID(1);
  ClientObjectWrapper output = process.getOutputMessage(piid);
  DataObject myOutput = null;
  if ( output.getObject() != null && output.getObject() instanceof DataObject )
  {
     myOutput  = (DataObject)output.getObject();
     int order = myOutput.getInt("OrderNo");
  }
}
```

アクティビティーの修復:

長期実行プロセスには、やはり長期間実行されるアクティビティーが含まれる場合があります。これらのアクティビティーでは、catch されていないエラーが発生して、停止状態になる可能性があります。実行状態のアクティビティーが、反応していないように見える可能性もあります。どちらの場合でも、プロセス管理者は、プロセスのナビゲーションを継続できるように、いくつかの方法でアクティビティーを処理することができます。

Business Process Choreographer API は、アクティビティーの修復のために、forceRetry メソッドおよび forceComplete メソッドを提供しています。アクティビティーの修復アクションをアプリケーションに追加する方法を示した例が提供されています。

アクティビティーの強制完了:

長期実行プロセスのアクティビティーで、障害が発生することがあります。これらの障害が、囲んでいるスコープ内で障害ハンドラーによって catch されておらず、関連したアクティビティー・テンプレートが、エラー発生時にアクティビティーが停止するように指定している場合、アクティビティーは修復することができるように停止状態になります。この状態で、アクティビティーの完了を強制することができます。

例えば、アクティビティーが応答しない場合、実行状態のアクティビティーを強制的に完了することもできます。

特定のタイプのアクティビティーでは、追加要件が存在します。

staff アクティビティー
送信されるはずだったメッセージ、または引き起こされるはすだった障害など、強制完了呼び出しでパラメーターを渡すことができます。

script アクティビティー
強制完了呼び出しで、パラメーターを渡すことはできません。ただし、修復する必要がある変数を設定する必要があります。

invoke アクティビティー
invoke アクティビティーが実行状態の場合、サブプロセスでない非同期サービスを呼び出す invoke アクティビティーを強制的に完了することもできます。例えば、非同期サービスが呼び出されて応答がない場合、こうすることがあります。

1. 停止状態の停止アクティビティーをリストします。

```
QueryResultSet result =
     process.query("DISTINCT ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
                    PROCESS_INSTANCE.NAME='CustomerOrder'",
                    null, null, null);
```

   このアクションは、 CustomerOrder プロセス・インスタンスに対して停止アクティビティーを戻します。

2. 例えば、停止したスタッフ・アクティビティーなどのアクティビティーを完了します。

   この例では、出力メッセージが渡されます。

```
if (result.size() > 0)
{
   result.first();
   AIID aiid = (AIID) result.getOID(1);
   ActivityInstanceData activity = process.getActivityInstance(aiid);
  ClientObjectWrapper output =
        process.createMessage(aiid, activity.getOutputMessageTypeName());
  DataObject myMessage = null;
  if ( output.getObject()!= null && output.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)output.getObject();
     //set the parts in your message, for example, an order number
     myMessage.setInt("OrderNo", 4711);
   }

   boolean continueOnError = true;
   process.forceComplete(aiid, output, continueOnError);
}
```

   このアクションによって、アクティビティーが完了します。エラーが発生した場合、**continueOnError** パラメーターが、forceComplete 要求によって障害が発生する場合に取るべきアクションを決定します。

   例では、**continueOnError** が true です。この値は、障害が発生した場合、アクティビティーは失敗状態になることを意味します。障害は、処理されるかプロセス・スコープに到達するまで、アクティビティーの囲んでいるスコープに伝搬されます。次にプロセスは障害状態になり、最終的に失敗状態になります。

停止されたアクティビティーの再試行:

長期実行プロセスのアクティビティーに、囲んでいるスコープ内で catch されていない障害が発生した場合、関連したアクティビティー・テンプレートが、エラー発生時にアクティビティーの停止を指定しているときは、アクティビティーは停止状態になり、修復することができます。アクティビティーの実行を再試行することができます。

アクティビティーが使用する変数を設定することができます。また、script アクティビティーの例外と共に、アクティビティーが予想したメッセージなど、強制再試行呼び出しのパラメーターを渡すこともできます。

1. 停止アクティビティーをリストします。

```
QueryResultSet result =
      process.query("DISTINCT ACTIVITY.AIID",
                    "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
                     PROCESS_INSTANCE.NAME='CustomerOrder'",
                     null, null, null);
```

   このアクションは、 CustomerOrder プロセス・インスタンスに対して停止アクティビティーを戻します。

2. 例えば、停止したスタッフ・アクティビティーなどのアクティビティーの実行を再試行します。

```
if (result.size() > 0)
{
  result.first();
  AIID aiid = (AIID) result.getOID(1);
  ActivityInstanceData activity = process.getActivityInstance(aiid);
  ClientObjectWrapper input =
        process.createMessage(aiid, activity.getOutputMessageTypeName());
  DataObject myMessage = null;
  if ( input.getObject()!= null && input.getObject() instanceof DataObject )
    {
      myMessage = (DataObject)input.getObject();
      //set the strings in your message, for example, chocolate is to be ordered
      myMessage.setString("OrderNo", "chocolate");
    }

   boolean continueOnError = true;
   process.forceRetry(aiid, input, continueOnError);
}
```

   このアクションによって、アクティビティーが再試行されます。エラーが発生した場合、**continueOnError** パラメーターによって、forceRetry 要求の処理中にエラーが発生した場合に実行するアクションが決まります。

   例では、**continueOnError** が true です。この場合、forceRetry 要求の処理中にエラーが発生すると、アクティビティーは失敗状態になります。障害は、処理されるかプロセス・スコープに到達するまで、アクティビティーの囲んでいるスコープに伝搬されます。次にプロセスは障害状態になり、最終的に失敗状態になります。

BusinessFlowManagerService インターフェース:

BusinessFlowManagerService インターフェースは、クライアント・アプリケーションから呼び出すことができるビジネス・プロセス機能を公開します。

BusinessFlowManagerService インターフェースから呼び出すことができるメソッドは、プロセスまたはアクティビティーの状態、およびそのメソッドが含まれているアプリケーションを使用するユーザーの権限によって異なります。ビジネス・プロセス・オブジェクトを操作するための main メソッドを、以下にリストします。これらのメソッドおよび BusinessFlowManagerService インターフェースで使用可能なその他のメソッドについての詳細は、com.ibm.bpe.api パッケージ内の Javadoc を参照してください。

プロセス・テンプレート

プロセス・テンプレートは、バージョン付けされ、デプロイされ、インストールされるプロセス・モデルで、ビジネス・プロセスの仕様を含んでいます。これは、例えば sendMessage() などの適切な要求を発行することによって、インスタンス化および開始することができます。プロセス・インスタンスの実行は、サーバーによって自動的に駆動されます。

表 21. プロセス・テンプレート用の API メソッド

メソッド	説明
getProcessTemplate	指定されたプロセス・テンプレートを取得します。
queryProcessTemplate	データベースに保管されているプロセス・テンプレートを取得します。

プロセス・インスタンス

以下の API メソッドは、プロセス・インスタンスを開始します。

表 22. プロセス・インスタンスを開始するための API メソッド

メソッド	説明
call	マイクロフローを作成および実行します。
callWithReplyContext	指定されたプロセス・テンプレートから、固有の開始サービスでのマイクロフローまたは固有の開始サービスでの長期実行プロセスを作成および実行します。呼び出しは、結果を非同期で待ちます。
callWithUISettings	マイクロフローを作成および実行し、出力メッセージとクライアント・ユーザー・インターフェース (UI) の設定を戻します。
initiate	プロセス・インスタンスを作成し、そのプロセス・インスタンスの処理を開始します。このメソッドは、長時間実行プロセスに使用します。このメソッドは、応答不要送信を適用するマイクロフローに対しても使用できます。

表 22. プロセス・インスタンスを開始するための API メソッド (続き)

メソッド	説明
sendMessage	指定されたメッセージを、指定されたアクティビティー・サービスおよびプロセス・インスタンスに送信します。同じ相関セット値を持つプロセス・インスタンスが存在しない場合は、作成されます。プロセスは、固有または非固有の開始サービスのどちらかを持ちます。
getStartActivities	指定されたプロセス・テンプレートからプロセス・インスタンスを開始できるアクティビティーに関する情報を戻します。
getActivityServiceTemplate	指定されたアクティビティー・サービス・テンプレートを取得します。

表 23. プロセス・インスタンスのライフ・サイクルを制御するための API メソッド

メソッド	説明
suspend	実行状態または失敗状態にある、長期実行中のトップレベルのプロセス・インスタンスの実行を中断します。
resume	中断状態にある、長期実行中のトップレベルのプロセス・インスタンスの実行を再開します。
restart	完了、失敗、または終了状態にある、長期実行中のトップレベルのプロセス・インスタンスを再始動します。
forceTerminate	指定されたトップレベルのプロセス・インスタンスと、子 autonomy を含むそのサブプロセス、およびその実行中のアクティビティー、要求済みのアクティビティー、または待機中のアクティビティーを終了します。
delete	指定されたトップレベルのプロセス・インスタンスと、子 autonomy を含むそのサブプロセスを削除します。
query	データベースから、検索基準に一致するプロパティーを取得します。

アクティビティー

invoke アクティビティーの場合、プロセス・モデルで、それらのアクティビティーがエラー状態でも続行されるように指定できます。continueOnError フラグを false に設定し、未処理エラーが発生すると、そのアクティビティーは停止状態になります。その場合は、プロセス管理者が、そのアクティビティーを修復することができます。continueOnError フラグおよびそれに関連する修復機能は、例えば、invoke アクティビティーが失敗することがある長期実行プロセスなどで使用することができますが、補正および障害処理のモデル化には、かなりの労力が必要です。

アクティビティーの操作および修復には、以下のメソッドが使用可能です。

表 24. アクティビティー・インスタンスのライフ・サイクルを制御するための API メソッド

メソッド	説明
claim	準備ができたアクティビティー・インスタンスを要求し、ユーザーがそのアクティビティーを使用できるようにします。
cancelClaim	アクティビティー・インスタンスの要求を取り消します。
complete	アクティビティー・インスタンスを完了します。
completeAndClaimSuccessor	スタッフ・アクティビティーを完了し、ログオン・ユーザーの同じプロセス・インスタンスで次のアクティビティーを要求します。
forceComplete	実行状態または停止状態にあるアクティビティー・インスタンスを強制的に完了します。
forceRetry	実行状態または停止状態にあるアクティビティー・インスタンスを強制的に反復します。
query	データベースから、検索基準に一致するプロパティーを取得します。

変数およびカスタム・プロパティー

このインターフェースは、変数の値を取得および設定するための get および set メソッドを提供します。指定されたプロパティーをプロセス・インスタンスおよびアクティビティー・インスタンスに関連付けたり、指定されたプロパティーをプロセス・インスタンスおよびアクティビティー・インスタンスから取得することもできます。カスタム・プロパティーの名前および値は、java.lang.String 型である必要があります。

表 25. 変数およびカスタム・プロパティーの API メソッド

メソッド	説明
getVariable	指定された変数を取得します。
setVariable	指定された変数を設定します。
getCustomProperty	指定されたアクティビティーまたはプロセス・インスタンスの指定されたカスタム・プロパティーを取得します。
getCustomProperties	指定されたアクティビティーまたはプロセス・インスタンスの指定されたカスタム・プロパティーを取得します。
getCustomPropertyNames	指定されたアクティビティーまたはプロセス・インスタンスのカスタム・プロパティーの名前を取得します。
setCustomProperty	指定されたアクティビティーまたはプロセス・インスタンスのカスタム固有値を保管します。

ヒューマン・タスク用のアプリケーションの開発

タスクは、コンポーネントが人をサービスとして呼び出したり、人がサービスを呼び出すための手段となります。ヒューマン・タスクに関する標準的なアプリケーションの例が提供されています。

Business Process Choreographer API について詳しくは、com.ibm.task.api パッケージにある Javadoc を参照してください。

ヒューマン・タスクのための許可のロール:

ヒューマン・タスクで実行できるアクションは、ご使用の許可のロールに依存します。このロールは、J2EE のロールまたはインスタンス・ベースのロールとすることができます。

ロールとは、同じ権限レベルを共用する従業員の集合です。Java 2 Platform, Enterprise Edition (J2EE) のロールは、ヒューマン・タスク・コンテナーが構成されたときにセットアップされます。インスタンス・ベースのロールは、タスクがモデル化されたときにヒューマン・タスクおよびエスカレーションに割り当てられます。ロール・ベースの許可については、グローバル・セキュリティーが WebSphere Application Server で使用可能になっていることが必要です。

J2EE のロール

以下の J2EE のロールがサポートされています。

- J2EE TaskSystemAdministrator。このロールを割り当てられたユーザーは、すべての特権を持ちます。また、このロールは、ヒューマン・タスクのシステム管理者とも言われます。
- J2EE TaskSystemMonitor。このロールを割り当てられたユーザーは、すべてのタスク・オブジェクトのプロパティーを表示できます。また、このロールは、ヒューマン・タスクのシステム・モニターとも言われます。

これらのロールへのユーザーおよびグループの割り当てを変更するには、管理コンソールを使用できます。

RACF セキュリティーによるロールの設定: この RACF 許可は、以下のセキュリティー・フィールドを指定した場合に適用されます。

- **com.ibm.security.SAF.authorization= true**

```
RDEFINE EJBROLE TaskSystemAdministrator  UACC(NONE)
PERMIT TaskSystemAdministrator CLASS(EJBROLE)  ID(userid) ACCESS(READ)

RDEFINE EJBROLE TaskSystemMonitor  UACC(NONE)
PERMIT TaskSystemMonitor  CLASS(EJBROLE)  ID(userid) ACCESS(READ)
```

- **com.ibm.security.SAF.delegation= true**

```
RDEFINE EJBROLE JMSAPIUser  UACC(NONE) APPLDATA(' userid')
```

Security Authorization Facility (SAF) ベースの許可 (RACF EJBROLE プロファイルの使用など) を使用して、WebSphere Application Server 管理コンソール・アプリケーションなど、EJB および Web アプリケーションにおける Java 2 Platform, Enterprise Edition (J2EE) ロールに対するクライアントのアクセスを制御できます。詳細については、 WebSphere Application Server for z/OS インフォメーション・セ

ンターの『役割ベースの許可の System Authorization Facility』を参照してください。

インスタンス・ベースのロール

タスク・インスタンスまたはエスカレーション・インスタンスはタスク・モデルではスタッフ・メンバーには直接割り当てられておらず、その代わり、使用可能なロールの 1 つに割り当てられます。インスタンス・ベースのロールに割り当てられたスタッフ・メンバーは、そのロールに応じたアクションを実行できます。インスタンス・ベースのロールとユーザーの関連付けは、スタッフの解決を使用して実行時に決定されます。

以下のインスタンス・ベースのロールがサポートされています。
- タスクに対して: 可能なインスタンス作成者、オリジネーター、可能なスターター、スターター、可能な所有者、所有者、読者、編集者、管理者
- エスカレーションに対して: エスカレーション受信者

これらのロールは、以下のアクションの実行が許可されています。

ロール	許可されたアクション
潜在的インスタンス作成者	このロールのメンバーは、タスクのインスタンスを作成できます。タスク・テンプレートまたはアプリケーション・コンポーネントに対して可能なインスタンス作成者が定義されていない場合、すべてのユーザーがこのロールのメンバーとして考慮されます。
オリジネーター	このロールのメンバーは、タスクが開始されるまで管理権限を持ちます。タスクが開始されると、オリジネーターは読者の権限を持ち、タスクの中断や再開、および作業項目の転送などの一部の管理アクションを実行できます。
可能なスターター	このロールのメンバーは、既存のタスク・インスタンスを開始できます。可能なスターターが指定されない場合、オリジネーターが可能なスターターになります。可能なスターターがない場合のインライン・タスクについては、デフォルトは全員となります。
スターター	このロールのメンバーは読者の権限を持ち、作業項目の転送などの一部の管理アクションを実行できます。
可能な所有者	このロールのメンバーはタスクを要求できます。タスク・テンプレートまたはアプリケーション・コンポーネントに対して可能な所有者が定義されていない場合、すべてのユーザーがこのロールのメンバーとして考慮されます。このロールのスタッフの解決が失敗する場合、管理者は潜在的な所有者として割り当てられます。
所有者	タスクを処理し、完了します。
読者	すべてのタスク・オブジェクトのプロパティーを表示できますが、操作はできません。
編集者	このロールのメンバーはタスクの内容を扱うことができますが、要求または完了することはできません。
管理者	このロールのメンバーは、タスク、タスク・テンプレート、およびエスカレーションを管理できます。

ロール	許可されたアクション
エスカレーション受信者	このロールのメンバーは、エスカレーションおよびエスカレートされたタスクの読者権限を持ちます。

タスクに対するアクションに必要なロール:

LocalHumanTaskManager または HumanTaskManager インターフェースへのアクセス権は、呼び出し元がタスクに対するすべてのアクションを実行できることは保証しません。呼び出し元は、そのアクションの実行も許可されている必要があります。次の表に、それぞれのロールで実行できるアクションを示します。

アクション	呼び出し元のプリンシパルのロール								
	所有者	(潜) 所有者	スターター	(潜) スターター	Origin	Admin	エディター	リーダー	Esc 受信側
callTask				X[1]	X[1]	X[1]			
cancelClaim	X					X			
claim		X				X			
complete	X					X			
completeWithFollowOnTask[3]	X					X			
createFaultMessage	X	X	X	X	X[1]	X	X	X	X
createInputMessage	X	X	X	X	X[1]	X	X	X	X
createOutputMessage	X	X	X	X	X[1]	X	X	X	X
createWorkItem					X[1、2]	X			
delete					X	X			
deleteWorkItem					X[1、2]	X			
getCustomProperty	X	X	X	X	X[1]	X	X	X	X
getDocumentation	X	X	X	X	X[1]	X	X	X	X
getFaultMessage	X	X	X	X	X[1]	X	X	X	X
getFaultNames	X	X	X	X	X[1]	X	X	X	X
getInputMessage	X	X	X	X	X[1]	X	X	X	X
getOutputMessage	X	X	X	X	X[1]	X	X	X	X
getRoleInfo	X	X	X	X	X[1]	X	X	X	X
getTask	X	X	X	X	X[1]	X	X	X	X
getUISettings	X	X	X	X	X[1]	X	X	X	X
resume	X				X[1]	X			
setCustomProperty	X		X		X[1]	X	X		
setFaultMessage	X					X	X		
setOutputMessage	X					X	X		
startTask				X	X[1]	X			
startTaskAsSubtask[4]	X					X			
suspend	X				X[1]	X			
suspendWithCancelClaim	X					X			

アクション	呼び出し元のプリンシパルのロール								
	所有者	(潜) 所有者	スターター	(潜) スターター	Origin	Admin	エディター	リーダー	Esc 受信側
terminate	X		X[1]		X[1]	X			
transferWorkItem	X		X		X[6]	X			
update	X		X		X[1]	X	X		
updateInactiveTask					X[5]				

注:

1. スタンドアロンのタスク、臨時のタスク、およびタスク・テンプレートから派生したタスクの場合のみ。
2. 潜在的な所有者、潜在的なスターター、編集者、読者、およびエスカレーション受信側作業項目の場合のみ。
3. また、呼び出し元は、後続のタスクに対して少なくともタスク・リーダー権限を持っている必要があります。
4. また、呼び出し元は、サブタスクに対して少なくともタスク・リーダー権限を持っている必要があります。
5. スタンドアロンのタスクおよび臨時のタスクの場合のみ。
6. 潜在的な所有者、潜在的なスターター、オリジネーター、編集者、読者、およびエスカレーション受信側作業項目の場合のみ。

略語:

Admin 管理者

Esc 受信側
エスカレーションの受信側

Origin オリジネーター

(潜) 所有者
潜在的な所有者

(潜) スターター
潜在的なスターター

同期インターフェースを起動する親タスクの開始:

同期インターフェースを起動する親タスクには、microflow 内のインライン親タスク、microflow 内のスタンドアロン親タスク、および 単純 Java クラスなどによってインプリメントされる SCA (サービス・コンポーネント・アーキテクチャー) コンポーネントを開始する親タスクが含まれます。

このシナリオでは、タスク・テンプレートのインスタンスが作成され、一部の顧客データが渡されます。両方向操作が戻るまで、タスクは実行状態のままです。タスクの結果である OrderNo が呼び出し元に戻されます。

1. **オプション:** タスク・テンプレートをリストして、実行する親タスクの名前を探します。

   タスクの名前が既に分かっている場合は、このステップはオプションです。

   ```
   TaskTemplate[] taskTemplates = task.queryTaskTemplates
   ("TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_ORIGINATING",
    "TASK_TEMPL.NAME",
     new Integer(50),
     null);
   ```

結果は名前でソートされます。ソート済みの派生元テンプレートのうちの最初の 50 個を収容した配列が照会から戻されます。

2. 該当する型の入力メッセージを作成します。

```
TaskTemplate template = taskTemplates[0];

// create a message for the selected task
ClientObjectWrapper input = task.createInputMessage( template.getID());
DataObject myMessage = null ;
if ( input.getObject()!= null && input.getObject() instanceof DataObject )
{
  myMessage = (DataObject)input.getObject();
  //set the parts in the message, for example, a customer name
  myMessage.setString("CustomerName", "Smith");
}
```

3. タスクを作成して、タスクを同期実行します。

   タスクを同期実行するには、両方向の操作であることが必要です。例では、createAndCallTask メソッドを使用してタスクを作成および実行します。

```
ClientObjectWrapper output = task.createAndCallTask( template.getName(),
                                                     template.getNamespace(),
                                                     input);
```

4. タスクの結果を分析します。

```
DataObject myOutput = null;
if ( output.getObject() != null && output.getObject() instanceof DataObject )
{
  myOutput  = (DataObject)output.getObject();
  int order = myOutput.getInt("OrderNo");
}
```

非同期インターフェースを起動する親タスクの開始:

非同期インターフェースを起動する親タスクには、microflow 内のインライン親タスク、microflow 内のスタンドアロン親タスク、および 単純 Java クラスなどによってインプリメントされる SCA (サービス・コンポーネント・アーキテクチャー) コンポーネントを開始する親タスクが含まれます。

このシナリオでは、タスク・テンプレートのインスタンスが作成され、一部の顧客データが渡されます。

1. **オプション:** タスク・テンプレートをリストして、実行する親タスクの名前を探します。

   タスクの名前が既に分かっている場合は、このステップはオプションです。

```
TaskTemplate[] taskTemplates = task.queryTaskTemplates
  ("TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_ORIGINATING",
   "TASK_TEMPL.NAME",
    new Integer(50),
    null);
```

   結果は名前でソートされます。ソート済みの派生元テンプレートのうちの最初の 50 個を収容した配列が照会から戻されます。

2. 該当する型の入力メッセージを作成します。

```
TaskTemplate template = taskTemplates[0];

// create a message for the selected task
ClientObjectWrapper input = task.createInputMessage( template.getID());
```

```
DataObject myMessage = null ;
if ( input.getObject()!= null && input.getObject() instanceof DataObject )
{
  myMessage = (DataObject)input.getObject();
  //set the parts in the message, for example, a customer name
  myMessage.setString("CustomerName", "Smith");
}
```

3. タスクを作成して、非同期に実行します。

   例では、createAndStartTask メソッドを使用してタスクを作成および実行します。

```
task.createAndStartTask( template.getName(),
                         template.getNamespace(),
                         input,
                         null);
```

タスク・インスタンスの作成と開始:

このシナリオでは、ヒューマン・タスクを定義するタスク・テンプレートのインスタンスを作成し、タスク・インスタンスを開始する方法を示します。

1. **オプション:** タスク・テンプレートをリストして、実行する親タスクの名前を探します。

   タスクの名前が既に分かっている場合は、このステップはオプションです。

```
TaskTemplate[] taskTemplates = task.queryTaskTemplates
("TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_HUMAN",
 "TASK_TEMPL.NAME",
  new Integer(50),
  null);
```

   結果は名前でソートされます。ソート済みのヒューマン・タスク・テンプレートのうちの最初の 50 個を収容した配列が照会から戻されます。

2. 該当する型の入力メッセージを作成します。

```
TaskTemplate template = taskTemplates[0];

// create a message for the selected task
ClientObjectWrapper input = task.createInputMessage( template.getID());
DataObject myMessage = null ;
if ( input.getObject()!= null && input.getObject() instanceof DataObject )
{
  myMessage = (DataObject)input.getObject();
  //set the parts in the message, for example, a customer name
  myMessage.setString("CustomerName", "Smith");
}
```

3. ヒューマン・タスクを作成し、開始します。この例では、応答ハンドラーは指定されません。

   この例では、createAndStartTask メソッドを使用してタスクを作成し、開始します。

```
TKIID tkiid = task.createAndStartTask( template.getName(),
                                       template.getNamespace(),
                                       input,
                                       null);
```

   タスク・インスタンスに関連する人に対して作業項目が作成されます。例えば、潜在的な所有者は、新規タスク・インスタンスを要求できます。

4. タスク・インスタンスを要求します。

```
ClientObjectWrapper input2 = task.claim(tkiid);
DataObject taskInput = null ;
if ( input2.getObject()!= null && input2.getObject() instanceof DataObject )
{
  taskInput = (DataObject)input2.getObject();
  // read the values
  ...
}
```

タスク・インスタンスが要求されると、タスクの入力メッセージが戻されます。

参加タスクまたは純粋なヒューマン・タスクの処理:

参加タスクや純粋なヒューマン・タスクは、作業項目を通じて、組織内のさまざまな人に割り当てられます。プロセスが staff アクティビティーにナビゲートしたときなどに、参加タスクとそれに関連した作業項目が作成されます。潜在的な所有者の中の 1 人が、作業項目に関連したタスクを要求します。このユーザーは、関係のある情報の提供とタスクの完了に対して責任があります。

1. 作業の準備ができている、ログオン・ユーザーに属するタスクをリストします。

```
QueryResultSet result =
     task.query("TASK.TKIID",
                "TASK.STATE = TASK.STATE.STATE_READY AND
                (TASK.KIND = TASK.KIND.KIND_PARTICIPATING OR
                 TASK.KIND = TASK.KIND.KIND_HUMAN)AND
                 WORK_ITEM.REASON =
                   WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                 null, null, null);
```

このアクションは、ログオン・ユーザーが作業することができるタスクが含まれる照会結果セットを戻します。

2. 作業対象のタスクを要求します。

```
if (result.size() > 0)
{
  result.first();
  TKIID tkiid = (TKIID) result.getOID(1);
  ClientObjectWrapper input = task.claim(tkiid);
  DataObject taskInput = null ;
  if ( input.getObject()!= null && input.getObject() instanceof DataObject )
  {
    taskInput = (DataObject)input.getObject();
    // read the values
    ...
  }
}
```

タスクが要求されると、タスクの入力メッセージが戻されます。

3. タスクの作業が完了した場合、タスクを完了します。

タスクは、正常に完了することも、障害メッセージが表示されて完了することもあります。タスクが正常に完了した場合、出力メッセージが渡されます。タスクが正常に完了しなかった場合、障害メッセージが渡されます。これらのアクションに対して、適切なメッセージを作成する必要があります。

a. タスクを正常に完了するには、出力メッセージを作成します。

```
ClientObjectWrapper output =
      task.createOutputMessage(tkiid);
DataObject myMessage = null ;
if ( output.getObject()!= null && output.getObject() instanceof DataObject )
{
  myMessage = (DataObject)output.getObject();
  //set the parts in your message, for example, an order number
  myMessage.setInt("OrderNo", 4711);
}

//complete the task
task.complete(tkiid, output);
```

このアクションは、オーダー番号が含まれる出力メッセージを設定します。タスクは、完了状態になります。

b. 障害が発生した場合にタスクを完了するには、障害メッセージを作成します。

```
//retrieve the faults modeled for the task
List faultNames = task.getFaultNames(tkiid);

//create a message of the appropriate type
ClientObjectWrapper myFault =
      task.createFaultMessage(tkiid, (String)faultNames.get(0));

// set the parts in your fault message, for example, an error number
DataObject myMessage = null ;
if ( myFault.getObject()!= null && input.getObject() instanceof DataObject )
{
  myMessage = (DataObject)myFault.getObject();
  //set the parts in the message, for example, a customer name
  myMessage.setInt("error",1304);
}

task.complete(tkiid, (String)faultNames.get(0), myFault);
```

このアクションにより、エラー・コードを含む障害メッセージが設定されます。タスクは、失敗状態になります。

タスク・インスタンスの中断と再開:

ヒューマン・タスク・インスタンスまたは参加しているタスク・インスタンスを中断し、それらを再び再開して完了することができます。

タスク・インスタンスは、作動可能状態または要求済み状態にすることができます。これをエスカレートすることが可能です。呼び出し元は、タスク・インスタンスの所有者、オリジネーター、または管理者である必要があります。

タスク・インスタンスは、実行中に中断することができます。そうすることによって、例えば、タスクの完了に必要な情報を収集することができます。情報が使用可能になったら、タスク・インスタンスを再開できます。

1. ログオン・ユーザーによって要求されたタスクのリストを取得します。

```
QueryResultSet result = task.query("DISTINCT TASK.TKIID",
                                   "TASK.STATE = TASK.STATE.STATE_CLAIMED",
                                   null, null, null);
```

このアクションにより、ログオン・ユーザーによって要求されたタスクのリストを含む照会結果セットが戻されます。

2. タスク・インスタンスを中断します。

```
if (result.size() > 0)
{
  result.first();
  TKIID tkiid = (TKIID) result.getOID(1);
  task.suspend(tkiid);
}
```

このアクションにより、指定されたタスク・インスタンスが中断されます。タスク・インスタンスは中断状態になります。

3. プロセス・インスタンスを再開します。

```
task.resume( tkiid );
```

このアクションにより、タスク・インスタンスが中断前の状態になります。

タスクの結果の分析:

参加タスクまたは純粋なヒューマン・タスクは非同期に実行されます。タスク開始時に応答ハンドラーが指定された場合、タスク完了時に自動的に出力メッセージが戻されます。応答ハンドラーが指定されていない場合、メッセージを明示的に検索する必要があります。

タスクの結果は、そのタスク・インスタンスの派生元となったタスク・テンプレートに、派生したタスク・インスタンスの自動削除が指定されていない場合にのみ、データベースに保管されます。

タスクの結果を分析します。

例では、正常に完了したタスクのオーダー番号を確認する方法を示します。

```
QueryResultSet result = task.query("DISTINCT TASK.TKIID",
                                   "TASK.NAME = 'CustomerOrder' AND
                                    TASK.STATE = TASK.STATE.STATE_FINISHED",
                                    null, null, null);
if (result.size() > 0)
{
  result.first();
  TKIID tkiid = (TKIID) result.getOID(1);
  ClientObjectWrapper output = task.getOutputMessage(tkiid);
  DataObject myOutput = null;
  if ( output.getObject() != null && output.getObject() instanceof DataObject)
  {
     myOutput  = (DataObject)output.getObject();
     int order = myOutput.getInt("OrderNo");
  }
}
```

タスク・インスタンスの終了:

管理者権限を有する担当者が、リカバリー不能状態になったと分かったタスク・インスタンスを終了する必要が生じる場合があります。タスク・インスタンスは即座に終了されるため、例外的な状況の場合のみ、タスク・インスタンスを終了することをお勧めします。

1. 終了するタスク・インスタンスを検索します。

```
Task taskInstance = task.getTask(tkiid);
```

2. タスク・インスタンスを終了します。

```
TKIID tkiid = taskInstance.getID();
task.terminate(tkiid);
```

タスク・インスタンスは、未解決のタスクを待機しないで即時に終了します。

タスク・インスタンスの削除:

タスク・インスタンスは、インスタンスの派生元となった関連タスク・テンプレートに自動削除が指定されている場合にのみ、完了時に自動的に削除されます。以下の例では、完了して自動的に削除されなかったタスク・インスタンスをすべて削除する方法を示します。

1. 完了したタスク・インスタンスをリストします。

```
QueryResultSet result =
     task.query("DISTINCT TASK.TKIID",
                "TASK.STATE = TASK.STATE.STATE_FINISHED",
                 null, null, null);
```

   このアクションにより、完了したタスク・インスタンスをリストした照会の結果セットが戻されます。

2. 完了したタスク・インスタンスを削除します。

```
while (result.next() )
{
 TKIID tkiid = (TKIID) result.getOID(1);
 task.delete(tkiid);
}
```

要求されたタスクの解放:

潜在的な所有者がタスクを要求した場合、この所有者にはタスクの完了に対する責任があります。ただし、要求されたタスクを、別の潜在的所有者が要求できるように解放しなければならない場合があります。

管理者権限を持つユーザーが、要求されたタスクを解放する必要がある場合があります。この状態は、例えば、タスクを完了する必要があるが、タスクの所有者が不在の場合に発生する可能性があります。タスクの所有者も、要求されたタスクを解放できます。

1. 特定のユーザー (例では、Smith) 所有の、要求されたタスクをリストします。

```
QueryResultSet result =
     task.query("DISTINCT TASK.TKIID",
                "TASK.STATE = TASK.STATE.STATE_CLAIMED AND
                 TASK.OWNER = 'Smith'",
                 null, null, null);
```

   このアクションは、指定されたユーザーの Smith が要求したタスクをリストする照会結果セットを戻します。

2. 要求されたタスクを解放します。

```
if (result.size() > 0)
{
  result.first();
  TKIID tkiid = (TKIID) result.getOID(1);
  task.cancelClaim(tkiid, true);
}
```

   このアクションは、タスクを作動可能状態に戻すので、他の潜在的な所有者の 1 人が要求することができます。元の所有者が設定した出力データまたは障害データはすべて保持されます。

作業項目の管理:

アクティビティー・インスタンスまたはタスク・インスタンスの存続時間中に、オブジェクトに関連したユーザーのセットが変わる場合があります。例えば、社員が休暇に入ったり、新しい社員を雇用したり、またはワークロードを異なった方法で分散させる必要がある場合です。このような変更に対応するために、作業項目を作成、削除、または転送するようにアプリケーションを開発することができます。

作業項目は、特定の理由でのユーザーまたはユーザーのグループへのオブジェクトの割り当てを表します。通常、このオブジェクトは、staff アクティビティー・インスタンス、プロセス・インスタンス、ヒューマン・タスクのいずれかです。理由は、ユーザーがアクティビティーまたはタスクに対して持っているロールから派生します。ユーザーは、アクティビティーまたはタスクとの関連において異なるロールを持つことができ、作業項目はこのようなロールそれぞれに対して作成されるため、1 つのアクティビティーやタスクが複数の作業項目を持つことが可能です。

作業項目を管理するために実行可能なアクションは、ユーザーのロールによって異なります。例えば、管理者は作業項目を作成、削除、および転送できますが、タスク所有者は作業項目の転送のみが可能です。

- 作業項目を作成します。

```
// query the task instance for which an additional
// administrator is to be specified
QueryResultSet result = task.query("TASK.TKIID",
                                   "TASK.NAME='CustomerOrder'",
                                    null, null, null);
if (result.size() > 0)
{
  result.first();
  // create the work item
  task.createWorkItem((TKIID)(result.getOID(1)),
                          WorkItem.REASON_ADMINISTRATOR,"Smith");
}
```

  このアクションによって、管理者のロールがあるユーザー Smith の作業項目が作成されます。

- 作業項目を削除します。

```
// query the task instance for which a work item is to be deleted
QueryResultSet result = task.query("TASK.TKIID",
                                   "TASK.NAME='CustomerOrder'",
                                    null, null, null);
if (result.size() > 0)
{
  result.first();
  // delete the work item
  task.deleteWorkItem((TKIID)(result.getOID(1)),
                          WorkItem.REASON_READER,"Smith");
}
```

  このアクションによって、リーダーのロールがあるユーザー Smith の作業項目が削除されます。

- 作業項目を転送します。

```
// query the task that is to be rescheduled
QueryResultSet result =
     task.query("DISTINCT TASK.TKIID",
                "TASK.NAME='CustomerOrder' AND
                 TASK.STATE=TASK.STATE.STATE_READY AND
```

```
                WORK_ITEM.REASON=WORK_ITEM.REASON.REASON_POTENTIAL_OWNER AND
                WORK_ITEM.OWNER_ID='Miller'",
                null, null, null);
if (result.size() > 0)
{
  result.first();
  // transfer the work item from user Miller to user Smith
  // so that Smith can work on the task
  task.transferWorkItem((TKIID)(result.getOID(1)),
                        WorkItem.REASON_POTENTIAL_OWNER,"Miller","Smith");
}
```

このアクションによって、作業項目をユーザー Smith に転送するので、 Smith は作業項目で作業することができます。

実行時のタスク・テンプレートおよびタスク・インスタンスの作成:

通常、WebSphere Integration Developer などのモデル化ツールを使用して、タスク・テンプレートを作成します。次に、タスク・テンプレートを WebSphere Process Server にインストールし、例えば Business Process Choreographer Explorer を使用して、これらのテンプレートからインスタンスを作成します。ただし、実行時にヒューマン・タスクまたは参加タスクのインスタンスまたはテンプレートを作成することもできます。例えば、アプリケーションのデプロイ時にタスク定義が使用不可である場合、ワークフローに含まれるタスクがまだ認識されていない場合、またはタスクがユーザーのグループ間の随時のコラボレーションを対象とする必要がある場合などに、このようにすることがあります。

1. **オプション:** ご使用のインターフェースに、単純 Java 型ではない型が含まれる場合、ランタイム・タスクまたはテンプレートが使用するデータ型を含むアプリケーションを作成または特定します。

   ランタイム・タスクまたはタスク・テンプレートは、アプリケーションのコンテキストで実行され、データ型へアクセスできるようになります。アプリケーションが Business Process Choreographer によってロードされるように、アプリケーションにタスク定義またはプロセス定義も含めるようにしてください。これらのタスクまたはプロセスは、ダミー・タスクまたはダミー・プロセスでもかまいません。
2. タスク・モデルを作成します。

   このモデルは、ステップ 1 で特定されたアプリケーション内のデータ型を参照します。
3. タスク・モデルを検証します。
4. タスク・テンプレートまたはタスク・インスタンスを作成します。

   HumanTaskManagerService インターフェースを使用して、このアクションを完了します。インターフェースに単純 Java 型以外の型が含まれる場合は、タスク・インスタンスまたはタスク・テンプレートを作成するときに、そのデータ型定義を含むアプリケーションの名前を指定してください。

単純 Java 型を使用するランタイム・タスクの作成:

この例では、インターフェースで単純 Java 型 (例えば String オブジェクト) のみを使用するランタイム・タスクを作成します。

この例は、リソースがロードされる呼び出し側エンタープライズ・アプリケーションのコンテキスト内のみで実行されます。

1. ClientTaskFactory にアクセスし、新規タスク・モデルの定義を格納するリソース・セットを作成します。

```
ClientTaskFactory factory = ClientTaskFactory.newInstance();
ResourceSet resourceSet = factory.createResourceSet();
```

2. 操作の WSDL 定義を作成し、記述を追加します。

```
// create the WSDL interface
Definition definition = factory.createWSDLDefinition
         ( resourceSet, new QName( "http://www.ibm.com/task/test/", "test" ) );

// create a port type
PortType portType = factory.createPortType( definition, "doItPT" );

// create an operation; the input and output messages are of type String:
// a fault message is not specified
Operation operation = factory.createOperation
        ( definition, portType, "doIt",
          new QName( "http://www.w3.org/2001/XMLSchema", "string" ),
          new QName( "http://www.w3.org/2001/XMLSchema", "string" ),
          null );
```

3. 新規ヒューマン・タスクの EMF モデルを作成します。

   タスク・インスタンスを作成する場合は、有効開始日 (UTCDate) は必要ありません。

```
TTask humanTask = factory.createTTask( resourceSet,
                                       TTaskKinds.HTASK_LITERAL,
                                       "TestTask",
                                       new UTCDate( "2005-01-01T00:00:00" ),
                                       "http://www.ibm.com/task/test/",
                                       portType,
                                       operation );
```

   このステップは、タスク・モデルのプロパティーをデフォルト値で初期化します。

4. ヒューマン・タスク・モデルのプロパティーを変更します。

```
// use the methods from the com.ibm.wbit.tel package, for example,
humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );

// retrieve the task factory to create or modify composite task elements
TaskFactory taskFactory = factory.getTaskFactory();

// specify escalation settings
TVerb verb = taskFactory.createTVerb();
verb.setName("John");

// create escalationReceiver and add verb
TEscalationReceiver escalationReceiver =
                    taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);

// create escalation and add escalation receiver
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);
```

5. すべてのリソース定義を含むタスク・モデルを作成します。

```
TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );
```

6. タスク・モデルを検証し、検証で問題が検出された場合それを訂正します。

```
ValidationProblem[] validationProblems = taskModel.validate();
```

7. ランタイム・タスク・インスタンスまたはランタイム・タスク・テンプレートを作成します。

   HumanTaskManagerService インターフェースを使用して、タスク・インスタンスまたはタスク・テンプレートを作成します。アプリケーションが単純 Java 型のみを使用するため、アプリケーション名を指定する必要はありません。

   - 以下の断片はタスク・インスタンスを作成します。

     ```
     task.createTask( taskModel, null, "HTM" );
     ```

   - 以下の断片はタスク・テンプレートを作成します。

     ```
     task.createTaskTemplate( taskModel, null );
     ```

ランタイム・タスク・インスタンスが作成された場合、それを始動することができます。ランタイム・タスク・テンプレートが作成された場合、そのテンプレートからタスク・インスタンスを作成できます。

複合タイプを使用するランタイム・タスクの作成:

この例では、インターフェースで複合タイプを使用するランタイム・タスクを作成します。複合タイプは既に定義されています。すなわち、クライアントのローカル・ファイル・システムには、複合タイプの記述を含む XSD ファイルがあります。

この例は、リソースがロードされる呼び出し側エンタープライズ・アプリケーションのコンテキスト内のみで実行されます。

1. ClientTaskFactory にアクセスし、新規タスク・モデルの定義を格納するリソース・セットを作成します。

   ```
   ClientTaskFactory factory = ClientTaskFactory.newInstance();
   ResourceSet resourceSet = factory.createResourceSet();
   ```

2. 複合タイプの XSD 定義をリソース・セットに追加して、操作の定義時に使用できるようにします。

   ファイルは、コードが実行されたロケーションの相対位置に配置されます。

   ```
   factory.loadXSDSchema( resourceSet, "InputBO.xsd" );
   factory.loadXSDSchema( resourceSet, "OutputBO.xsd" );
   ```

3. 操作の WSDL 定義を作成し、記述を追加します。

   ```
   // create the WSDL interface
   Definition definition = factory.createWSDLDefinition
            ( resourceSet, new QName( "http://www.ibm.com/task/test/", "test" ) );

   // create a port type
   PortType portType = factory.createPortType( definition, "doItPT" );

   // create an operation; the input message is an InputBO and
   // the output message an OutputBO;
   // a fault message is not specified
   Operation operation = factory.createOperation
           ( definition, portType, "doIt",
             new QName( "http://Input", "InputBO" ),
             new QName( "http://Output", "OutputBO" ),
             null );
   ```

4. 新規ヒューマン・タスクの EMF モデルを作成します。

タスク・インスタンスを作成する場合は、有効開始日 (UTCDate) は必要ありません。

```
TTask humanTask = factory.createTTask( resourceSet,
                                       TTaskKinds.HTASK_LITERAL,
                                       "TestTask",
                                       new UTCDate( "2005-01-01T00:00:00" ),
                                       "http://www.ibm.com/task/test/",
                                       portType,
                                       operation );
```

このステップは、タスク・モデルのプロパティーをデフォルト値で初期化します。

5. ヒューマン・タスク・モデルのプロパティーを変更します。

```
// use the methods from the com.ibm.wbit.tel package, for example,
humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );

// retrieve the task factory to create or modify composite task elements
TaskFactory taskFactory = factory.getTaskFactory();

// specify escalation settings
TVerb verb = taskFactory.createTVerb();
verb.setName("John");

// create escalationReceiver and add verb
TEscalationReceiver escalationReceiver =
                    taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);

// create escalation and add escalation receiver
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);
```

6. すべてのリソース定義を含むタスク・モデルを作成します。

```
TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );
```

7. タスク・モデルを検証し、検証で問題が検出された場合それを訂正します。

```
ValidationProblem[] validationProblems = taskModel.validate();
```

8. ランタイム・タスク・インスタンスまたはランタイム・タスク・テンプレートを作成します。

   HumanTaskManagerService インターフェースを使用して、タスク・インスタンスまたはタスク・テンプレートを作成します。データ型定義を利用できるように、データ型定義を含むアプリケーション名を指定する必要があります。アプリケーションが Business Process Choreographer によってロードされるように、アプリケーションにダミー・タスクまたはダミー・プロセスも含まれるようにしてください。

   - 以下の断片はタスク・インスタンスを作成します。

     ```
     task.createTask( taskModel, "BOapplication", "HTM" );
     ```

   - 以下の断片はタスク・テンプレートを作成します。

     ```
     task.createTaskTemplate( taskModel, "BOapplication" );
     ```

ランタイム・タスク・インスタンスが作成された場合、それを始動することができます。ランタイム・タスク・テンプレートが作成された場合、そのテンプレートからタスク・インスタンスを作成できます。

既存のインターフェースを使用するランタイム・タスクの作成:

この例は、既に定義されているインターフェースを使用するランタイム・タスクを作成します。すなわち、クライアントのローカル・ファイル・システムには、インターフェースの記述を含むファイルがあります。

この例は、リソースがロードされる呼び出し側エンタープライズ・アプリケーションのコンテキスト内のみで実行されます。

1. ClientTaskFactory にアクセスし、新規タスク・モデルの定義を格納するリソース・セットを作成します。

   ```
   ClientTaskFactory factory = ClientTaskFactory.newInstance();
   ResourceSet resourceSet = factory.createResourceSet();
   ```

2. 操作の WSDL 定義および記述にアクセスします。

   インターフェース記述は、コードが実行されたロケーションの相対位置に配置されます。

   ```
   Definition definition = factory.loadWSDLDefinition(
                             resourceSet, "interface.wsdl"  );
   PortType portType = definition.getPortType(
                        new QName( definition.getTargetNamespace(), "doItPT" ) );
   Operation operation = portType.getOperation("doIt", null, null );
   ```

3. 新規ヒューマン・タスクの EMF モデルを作成します。

   タスク・インスタンスを作成する場合は、有効開始日 (UTCDate) は必要ありません。

   ```
   TTask humanTask = factory.createTTask( resourceSet,
                                          TTaskKinds.HTASK_LITERAL,
                                          "TestTask",
                                          new UTCDate( "2005-01-01T00:00:00" ),
                                          "http://www.ibm.com/task/test/",
                                          portType,
                                          operation );
   ```

   このステップは、タスク・モデルのプロパティーをデフォルト値で初期化します。

4. ヒューマン・タスク・モデルのプロパティーを変更します。

   ```
   // use the methods from the com.ibm.wbit.tel package, for example,
   humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );

   // retrieve the task factory to create or modify composite task elements
   TaskFactory taskFactory = factory.getTaskFactory();

   // specify escalation settings
   TVerb verb = taskFactory.createTVerb();
   verb.setName("John");

   // create escalationReceiver and add verb
   TEscalationReceiver escalationReceiver =
                       taskFactory.createTEscalationReceiver();
   escalationReceiver.setVerb(verb);

   // create escalation and add escalation receiver
   TEscalation escalation = taskFactory.createTEscalation();
   escalation.setEscalationReceiver(escalationReceiver);
   ```

5. すべてのリソース定義を含むタスク・モデルを作成します。

   ```
   TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );
   ```

6. タスク・モデルを検証し、検証で問題が検出された場合それを訂正します。

```
ValidationProblem[] validationProblems = taskModel.validate();
```

7. ランタイム・タスク・インスタンスまたはランタイム・タスク・テンプレートを作成します。

   HumanTaskManagerService インターフェースを使用して、タスク・インスタンスまたはタスク・テンプレートを作成します。データ型定義を利用できるように、データ型定義を含むアプリケーション名を指定する必要があります。アプリケーションが Business Process Choreographer によってロードされるように、アプリケーションにダミー・タスクまたはダミー・プロセスも含まれるようにしてください。

   - 以下の断片はタスク・インスタンスを作成します。

     ```
     task.createTask( taskModel, "BOapplication", "HTM" );
     ```

   - 以下の断片はタスク・テンプレートを作成します。

     ```
     task.createTaskTemplate( taskModel, "BOapplication" );
     ```

ランタイム・タスク・インスタンスが作成された場合、それを始動することができます。ランタイム・タスク・テンプレートが作成された場合、そのテンプレートからタスク・インスタンスを作成できます。

呼び出し側アプリケーションのインターフェースを使用するランタイム・タスクの作成:

この例では、呼び出し側アプリケーションに属するインターフェースを使用するランタイム・タスクを作成します。例えば、ランタイム・タスクがビジネス・プロセスの Java 断片で作成され、プロセス・アプリケーションからのインターフェースを使用します。

この例は、リソースがロードされる呼び出し側エンタープライズ・アプリケーションのコンテキスト内のみで実行されます。

1. ClientTaskFactory にアクセスし、新規タスク・モデルの定義を格納するリソース・セットを作成します。

   ```
   ClientTaskFactory factory = ClientTaskFactory.newInstance();

   // specify the context class loader so that following resources are found
   ResourceSet resourceSet = factory.createResourceSet
                ( Thread.currentThread().getContextClassLoader() );
   ```

2. 操作の WSDL 定義および記述にアクセスします。

   収容パッケージ JAR ファイル内でパスを指定します。

   ```
   Definition definition = factory.loadWSDLDefinition( resourceSet,
                           "com/ibm/workflow/metaflow/interface.wsdl" );
                        PortType portType = definition.getPortType(
                        new QName( definition.getTargetNamespace(), "doItPT" ) );
   Operation operation = portType.getOperation("doIt", null, null );
   ```

3. 新規ヒューマン・タスクの EMF モデルを作成します。

   タスク・インスタンスを作成する場合は、有効開始日 (UTCDate) は必要ありません。

   ```
   TTask humanTask = factory.createTTask( resourceSet,
                                          TTaskKinds.HTASK_LITERAL,
                                          "TestTask",
   ```

```
new UTCDate( "2005-01-01T00:00:00" ),
"http://www.ibm.com/task/test/",
portType,
operation );
```

このステップは、タスク・モデルのプロパティーをデフォルト値で初期化します。

4. ヒューマン・タスク・モデルのプロパティーを変更します。

```
// use the methods from the com.ibm.wbit.tel package, for example,
humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );

// retrieve the task factory to create or modify composite task elements
TaskFactory taskFactory = factory.getTaskFactory();

// specify escalation settings
TVerb verb = taskFactory.createTVerb();
verb.setName("John");

// create escalationReceiver and add verb
TEscalationReceiver escalationReceiver =
                    taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);

// create escalation and add escalation receiver
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);
```

5. すべてのリソース定義を含むタスク・モデルを作成します。

```
TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );
```

6. タスク・モデルを検証し、検証で問題が検出された場合それを訂正します。

```
ValidationProblem[] validationProblems = taskModel.validate();
```

7. ランタイム・タスク・インスタンスまたはランタイム・タスク・テンプレートを作成します。

   HumanTaskManagerService インターフェースを使用して、タスク・インスタンスまたはタスク・テンプレートを作成します。データ型定義を利用できるように、データ型定義を含むアプリケーション名を指定する必要があります。

   - 以下の断片はタスク・インスタンスを作成します。

     ```
     task.createTask( taskModel, "WorkflowApplication", "HTM" );
     ```

   - 以下の断片はタスク・テンプレートを作成します。

     ```
     task.createTaskTemplate( taskModel, "WorkflowApplication" );
     ```

ランタイム・タスク・インスタンスが作成された場合、それを始動することができます。ランタイム・タスク・テンプレートが作成された場合、そのテンプレートからタスク・インスタンスを作成できます。

ヒューマン・タスク機能をカスタマイズするプラグインの作成:

Business Process Choreographer では、ヒューマン・タスクの処理中に発生するイベントのイベント処理インフラストラクチャーを提供します。プラグイン・ポイントも提供されるので、必要に応じて機能を適合させることができます。サービス・プロバイダー・インターフェース (SPI) を使用すると、イベントの処理およびスタッフ照会の処理用にプラグインをカスタマイズすることができます。

ヒューマン・タスク API イベントとエスカレーション通知イベント用のプラグインを作成することができます。また、スタッフ解決から戻される結果を処理するプラグインを作成することもできます。例えば、ピーク時に結果リストにユーザーを追加して、ワークロードのバランスを取ることもできます。

プラグインは、グローバル・レベルのすべてのタスク、1 つのアプリケーション・コンポーネントのタスク、1 つのタスク・テンプレートに関連付けられたすべてのタスク、または単一のタスク・インスタンスなど、さまざまなレベルで登録することができます。

API イベント・ハンドラーの作成:

API メソッドがヒューマン・タスクを操作するときに API イベントが発生します。API イベント・ハンドラー・プラグインのサービス・プロバイダー・インターフェース (SPI) を使用して、API イベントまたは同等の API イベントを持つ内部イベントが送信するタスク・イベントを処理するプラグインを作成します。

API イベント・ハンドラーを作成するには、次のステップを実行します。

1. APIEventHandlerPlugin2 インターフェースをインプリメントするクラス、または APIEventHandler インプリメンテーション・クラスを拡張するクラスを作成します。 このクラスは他のクラスのメソッドを呼び出すことができます。
   - APIEventHandlerPlugin2 インターフェースを使用する場合は、APIEventHandlerPlugin2 インターフェースおよび APIEventHandlerPlugin インターフェースのすべてのメソッドをインプリメントする必要があります。
   - SPI インプリメンテーション・クラスを拡張する場合は、必要なメソッドを上書きしてください。

   このクラスは、Java 2 Enterprise Edition (J2EE) Enterprise JavaBeans (EJB) アプリケーションのコンテキストで実行します。このクラスおよびそのヘルパー・クラスが EJB 仕様に準拠していることを確認します。

   ヒント: このクラスから HumanTaskManagerService インターフェースを呼び出す場合は、イベントを作成したタスクを更新するメソッドは呼び出さないでください。これを呼び出すと、データベース・デッドロックが発生します。
2. プラグイン・クラスとそのヘルパー・クラスを JAR ファイルにアセンブルします。

   ヘルパー・クラスが複数の J2EE アプリケーションで使用される場合、それらのクラスを別の JAR ファイルにパッケージして、共用ライブラリーとして登録することができます。
3. プラグインのサービス・プロバイダー構成ファイルを、JAR ファイルの META-INF/services/ ディレクトリーに作成します。

   この構成ファイルによって、プラグインを識別し、ロードするメカニズムが提供されます。このファイルは、Java 2 サービス・プロバイダー・インターフェース仕様に準拠します。

   a. `com.ibm.task.spi.`*`plug-in_name`*`APIEventHandlerPlugin` という名前のファイルを作成します。 *plug-in_name* はプラグインの名前です。

例えば、Customer という名前のプラグインが com.ibm.task.spi.APIEventHandlerPlugin インターフェースをインプリメントする場合、構成ファイルの名前は com.ibm.task.spi.**Customer**APIEventHandlerPlugin になります。

b. このファイルのコメント行、ブランク行以外の最初の行で、ステップ 1 で作成したプラグイン・クラスの完全修飾名を指定します。

例えば、MyAPIEventHandler というプラグイン・クラスが com.customer.plugins パッケージにある場合は、構成ファイルの最初の行に com.customer.plugins.MyAPIEventHandler という項目が含まれていなければなりません。

API イベントを処理するプラグインを含むインストール可能 JAR ファイルと、プラグインのロードに使用できるサービス・プロバイダー構成ファイルができます。

ヒント: イベント・ハンドラーを登録するために使用可能な eventHandlerName プロパティーは 1 つだけです。API イベント・ハンドラーおよび通知イベント・ハンドラーを使用する場合、両方のプラグイン・インプリメンテーションは同じ名前でなければなりません (例えば、SPI インプリメンテーションのイベント・ハンドラー名として Customer など)。

両方のプラグイン・インプリメンテーションを同じ JAR ファイルにパッケージする場合は、META-INF/services/ ディレクトリーに 2 つのファイルを作成します (例: com.ibm.task.spi.CustomerNotificationEventHandlerPlugin と com.ibm.task.spi.CustomerAPIEventHandlerPlugin2 など)。両方の SPI インプリメンテーションを別々の JAR ファイルにパッケージすることもできます。

また、1 つのクラスで両方のインターフェースをインプリメントすることもできます。この方式を選択すると、META-INF/services/ ディレクトリーに 2 つのエントリーが必要になります。単一クラスで両方のインターフェースをインプリメントすると、クラス・ロード問題が発生する可能性があるため、単一クラスを 2 つの異なる JAR ファイルにパッケージしないでください。

次に、実行時にヒューマン・タスク・コンテナーで使用できるように、このプラグインをインストールおよび登録する必要があります。API イベント・ハンドラーをタスク・インスタンス、タスク・テンプレート、またはアプリケーション・コンポーネントで登録することができます。

API イベント・ハンドラー:

ヒューマン・タスクが変更されたり、その状態が変わったりすると、API イベントが発生します。これらの API イベントを処理するために、タスクが変更される前 (pre-event メソッド)、および API 呼び出しが戻る直前 (post-event メソッド) に、イベント・ハンドラーが直接呼び出されます。

pre-event メソッドが ApplicationVetoException 例外をスローする場合、 API アクションは実行されません。例外は API 呼び出し元に戻され、イベントに関連したトランザクションはロールバックされます。pre-event メソッドが内部イベントによって

起動され、ApplicationVetoException 例外がスローされる場合、自動要求などの内部イベントは実行されませんが、例外はクライアント・アプリケーションに戻されません。この場合、情報メッセージが SystemOut.log ファイルに書き込まれます。API メソッドが処理中に例外をスローする場合、例外がキャッチされ、post-event メソッドに渡されます。 post-event メソッドが戻ると、例外は再び呼び出し元に渡されます。

以下の規則が、pre-event メソッドに適用されます。

- pre-event メソッドは、関連した API メソッドまたは内部イベントのパラメーターを受け取ります。
- pre-event メソッドは、処理が継続されないようにするため、ApplicationVetoException 例外をスローできます。

以下の規則が、post-event メソッドに適用されます。

- post-event メソッドは、API 呼び出しに提供されたパラメーター、および戻り値を受け取ります。例外が API メソッド・インプリメンテーションによってスローされる場合、post-event メソッドも例外を受け取ります。
- post-event メソッドは、戻り値を変更できません。
- post-event メソッドは例外をスローできません。実行時例外がログに記録されますが、無視されます。

API イベント・ハンドラーをインプリメントするには、APIEventHandlerPlugin インターフェースを拡張する APIEventHandlerPlugin2 インターフェースを使用するか、またはデフォルトの com.ibm.task.spi.APIEventHandler SPI インプリメンテーション・クラスを拡張します。イベント・ハンドラーは、デフォルトのインプリメンテーション・クラスから継承される場合、常に最新バージョンの SPI をインプリメントします。より新しいバージョンの Business Process Choreographer にアップグレードすれば、新規 SPI メソッドを活用するために必要な変更が少なくてすみます。

通知イベント・ハンドラーと API イベント・ハンドラーの両方がある場合、イベント・ハンドラー名は 1 つしか登録できないので、両方のハンドラーの名前は同じでなければなりません。

通知イベント・ハンドラーの作成:

ヒューマン・タスクがエスカレートされると、通知イベントが作成されます。Business Process Choreographer は、エスカレーション処理の機能 (エスカレーション作業項目の作成または E メールの送信など) を提供します。通知イベント・ハンドラーを作成し、エスカレーションが処理される方法をカスタマイズすることができます。

通知イベント・ハンドラーをインプリメントするには、NotificationEventHandlerPlugin インターフェースを使用する方法と、デフォルトの com.ibm.task.spi.NotificationEventHandler サービス・プロバイダー・インターフェース (SPI) インプリメンテーション・クラスを拡張する方法があります。

通知イベント・ハンドラーを作成するには、次のステップを実行します。

1. NotificationEventHandlerPlugin インターフェースをインプリメントするクラス、または NotificationEventHandler インプリメンテーション・クラスを拡張するクラスを作成します。 このクラスは他のクラスのメソッドを呼び出すことができます。

   NotificationEventHandlerPlugin インターフェースを使用する場合は、すべてのインターフェース・メソッドをインプリメントする必要があります。SPI インプリメンテーション・クラスを拡張する場合は、必要なメソッドを上書きしてください。

   このクラスは、Java 2 Enterprise Edition (J2EE) Enterprise JavaBeans (EJB) アプリケーションのコンテキストで実行します。このクラスおよびそのヘルパー・クラスが EJB 仕様に準拠していることを確認します。

   プラグインは、EscalationUser ロールの権限で呼び出されます。このロールは、ヒューマン・タスク・コンテナーが構成されると、定義されます。

   ヒント: このクラスから HumanTaskManagerService インターフェースを呼び出す場合は、イベントを作成したタスクまたはエスカレーションを更新するメソッドは呼び出さないでください。これを呼び出すと、データベース・デッドロックが発生します。
2. プラグイン・クラスとそのヘルパー・クラスを JAR ファイルにアセンブルします。

   ヘルパー・クラスが複数の J2EE アプリケーションで使用される場合、それらのクラスを別の JAR ファイルにパッケージして、共用ライブラリーとして登録することができます。
3. プラグインのサービス・プロバイダー構成ファイルを、JAR ファイルの META-INF/services/ ディレクトリーに作成します。

   この構成ファイルによって、プラグインを識別し、ロードするメカニズムが提供されます。このファイルは、Java 2 サービス・プロバイダー・インターフェース仕様に準拠します。

   a. `com.ibm.task.spi.`*`plug-in_name`*`NotificationEventHandlerPlugin` という名前のファイルを作成します。 *plug-in_name* はプラグインの名前です。

      例えば、`HelpDeskRequest` (イベント・ハンドラー名) という名前のプラグインが com.ibm.task.spi.NotificationEventHandlerPlugin インターフェースをインプリメントする場合、構成ファイルの名前は `com.ibm.task.spi.`**`HelpDeskRequest`**`NotificationEventHandlerPlugin` になります。

   b. このファイルのコメント行、ブランク行以外の最初の行で、ステップ 1 で作成したプラグイン・クラスの完全修飾名を指定します。

      例えば、`MyEventHandler` というプラグイン・クラスが `com.customer.plugins` パッケージにある場合は、構成ファイルの最初の行に `com.customer.plugins.MyEventHandler` という項目が含まれていなければなりません。

通知イベントを処理するプラグインを含むインストール可能 JAR ファイルと、プラグインのロードに使用できるサービス・プロバイダー構成ファイルができます。API イベント・ハンドラーをタスク・インスタンス、タスク・テンプレート、またはアプリケーション・コンポーネントで登録することができます。

ヒント: イベント・ハンドラーを登録するために使用可能な eventHandlerName プロパティーは 1 つだけです。API イベント・ハンドラーおよび通知イベント・ハンドラーを使用する場合、両方のプラグイン・インプリメンテーションは同じ名前でなければなりません (例えば、SPI インプリメンテーションのイベント・ハンドラー名として `Customer` など)。

両方のプラグイン・インプリメンテーションを同じ JAR ファイルにパッケージする場合は、META-INF/services/ ディレクトリーに 2 つのファイルを作成します (例: com.ibm.task.spi.CustomerNotificationEventHandlerPlugin と com.ibm.task.spi.CustomerAPIEventHandlerPlugin2 など)。両方の SPI インプリメンテーションを別々の JAR ファイルにパッケージすることもできます。

また、1 つのクラスで両方のインターフェースをインプリメントすることもできます。この方式を選択すると、META-INF/services/ ディレクトリーに 2 つのエントリーが必要になります。単一クラスで両方のインターフェースをインプリメントすると、クラス・ロード問題が発生する可能性があるため、単一クラスを 2 つの異なる JAR ファイルにパッケージしないでください。

次に、実行時にヒューマン・タスク・コンテナーで使用できるように、このプラグインをインストールおよび登録する必要があります。通知イベント・ハンドラーをタスク・インスタンス、タスク・テンプレート、またはアプリケーション・コンポーネントで登録することができます。

スタッフ照会結果の後処理を行うプラグインの作成:

スタッフ解決は、特定のロール (例えばタスクの潜在的な所有者) に割り当てられているユーザーのリストを戻します。スタッフ解決で戻されるスタッフ照会の結果を変更するプラグインを作成することができます。例えば、ワークロード・バランシングを改善するために、既にワークロードが高いユーザーを照会結果から除去するプラグインを使用することがあります。

後処理プラグインは 1 つしか所有できません。つまり、そのプラグインですべてのタスクのスタッフ結果を処理する必要があります。このプラグインでは、ユーザーの追加または除去、あるいはユーザーまたはグループ情報の変更ができます。また、結果タイプを、例えばユーザー・リストからグループに、あるいは全員に、変更することもできます。

プラグインはスタッフ解決の完了後に実行されるので、機密性またはセキュリティーを保持するための規則が既に適用されています。プラグインは、スタッフの解決中に除去されたユーザーについての情報を受け取ります (HTM_REMOVED_USERS マップ・キーで)。プラグインがこのコンテキスト情報を使用して、機密性またはセキュリティー規則が保持されるようにしてください。

スタッフ照会結果の後処理をインプリメントするには、StaffQueryResultPostProcessingPlugin インターフェースを使用します。このインターフェースには、タスク、エスカレーション、タスク・テンプレート、およびアプリケーション・コンポーネントの照会結果を変更するメソッドが含まれています。

スタッフ照会結果の後処理を行うプラグインを作成するには、次のステップを実行します。

1. StaffQueryResultPostProcessingPlugin インターフェースをインプリメントするクラスを作成します。

   すべてのインターフェース・メソッドをインプリメントする必要があります。このクラスは他のクラスのメソッドを呼び出すことができます。

   このクラスは、Java 2 Enterprise Edition (J2EE) Enterprise JavaBeans (EJB) アプリケーションのコンテキストで実行します。このクラスおよびそのヘルパー・クラスが EJB 仕様に準拠していることを確認します。

   ヒント: このクラスから HumanTaskManagerService インターフェースを呼び出す場合は、イベントを作成したタスクを更新するメソッドは呼び出さないでください。これを呼び出すと、データベース・デッドロックが発生します。

   次の例では、SpecialTask というタスクのエディター・ロールを変更する方法を示しています。

```
public StaffQueryResult processStaffQueryResult
                        (StaffQueryResult originalStaffQueryResult,
                                          Task task,
                                          int  role,
                                          Map  context)
{
  StaffQueryResult newStaffQueryResult = originalStaffQueryResult;
  StaffQueryResultFactory staffResultFactory =
                          StaffQueryResultFactory.newInstance();
  if (role == com.ibm.task.api.WorkItem.REASON_EDITOR &&
      task.getName() != null &&
      task.getName().equals("SpecialTask"))
  {
      UserData user = staffResultFactory.newUserData
                           ("SuperEditor",
                             new Locale("en-US"),
                            "SuperEditor@company.com");
      ArrayList userList = new ArrayList();
      userList.add(user);

      newStaffQueryResult = staffResultFactory.newStaffQueryResult(userList);
   }
   return(newStaffQueryResult);
}
```

2. プラグイン・クラスとそのヘルパー・クラスを JAR ファイルにアセンブルします。

   ヘルパー・クラスが複数の J2EE アプリケーションで使用される場合、それらのクラスを別の JAR ファイルにパッケージして、共用ライブラリーとして登録することができます。

3. プラグインのサービス・プロバイダー構成ファイルを、JAR ファイルの META-INF/services/ ディレクトリーに作成します。

この構成ファイルによって、プラグインを識別し、ロードするメカニズムが提供されます。このファイルは、Java 2 サービス・プロバイダー・インターフェース仕様に準拠します。

a. com.ibm.task.spi.*plug-in_name*StaffQueryResultPostProcessingPlugin という名前のファイルを作成します。 *plug-in_name* はプラグインの名前です。

   例えば、MyHandler という名前のプラグインが com.ibm.task.spi.StaffQueryResultPostProcessingPlugin インターフェースをインプリメントする場合、構成ファイルの名前は com.ibm.task.spi.**MyHandler**StaffQueryResultPostProcessingPlugin になります。

b. このファイルのコメント行、ブランク行以外の最初の行で、ステップ 1 で作成したプラグイン・クラスの完全修飾名を指定します。

   例えば、StaffPostProcessor というプラグイン・クラスが com.customer.plugins パッケージにある場合は、構成ファイルの最初の行に com.customer.plugins.StaffPostProcessor という項目が含まれていなければなりません。 通知イベントを処理するプラグインを含むインストール可能 JAR ファイルと、プラグインのロードに使用できるサービス・プロバイダー構成ファイルができます。

4. プラグインをインストールします。

   スタッフ照会結果の後処理プラグインは 1 つしか所有できません。プラグインを共用ライブラリーとしてインストールする必要があります。

5. プラグインを登録します。

   a. 管理コンソールで、ヒューマン・タスク・コンテナーの「カスタム・プロパティー」ページに移動します。(**「アプリケーション・サーバー」** → ***server_name*** → **「ヒューマン・タスク・コンテナー」** → **「カスタム・プロパティー」**)。

   b. **Staff.PostProcessingPlugin** という名前と、プラグインにつけた名前の値 (この例では、MyHandler) を持つカスタム・プロパティーを追加します。

プラグインのインストール:

プラグインを使用するには、タスク・コンテナーからアクセスできるように、プラグインをインストールする必要があります。

プラグインのインストール方法は、そのプラグインが 1 つの Java 2 Enterprise Edition (J2EE) アプリケーションでのみ使用されるか、複数のアプリケーションで使用されるかによって異なります。

プラグインをインストールするには、以下のいずれかのステップを実行します。

- 単一の J2EE アプリケーションで使用するプラグインをインストールする場合。

  アプリケーション EAR ファイルにプラグイン JAR ファイルを追加します。WebSphere Integration Developer のデプロイメント記述子エディターで、プラグインの JAR ファイルを、主要な Enterprise JavaBeans (EJB) モジュールの J2EE アプリケーション用のプロジェクト・ユーティリティー JAR ファイルとしてインストールします。

- 複数の J2EE アプリケーションで使用するプラグインをインストールする場合。

  JAR ファイルを WebSphere Application Server 共用ライブラリーに入れ、そのライブラリーを、プラグインへのアクセスが必要なアプリケーションと関連付けます。 JAR ファイルをネットワーク・デプロイメント環境で使用できるようにするには、各サーバーに手動で JAR ファイルを配布し、その後、各セルに一度ずつ共用ライブラリーをインストールします。

これで、プラグインを登録することができます。

プラグインの登録:

プラグインをタスク・コンテナーの成果物階層に異なるレベルで登録することができます。例えば、グローバル・レベルのすべてのタスク、1 つのアプリケーション・コンポーネントのタスク、1 つのタスク・テンプレートに関連付けられたすべてのタスク、または単一のタスク・インスタンスなどに分けます。

複数のプラグインを登録すると、スコーピングがサポートされます。つまり、タスク・コンテナーの成果物階層の低いレベルで登録されているプラグイン (タスク・インスタンスなど) が、高いレベルで登録されているプラグイン (タスク・テンプレート、アプリケーション・コンポーネントなど) の代わりに使用されていることを意味しています。スコーピングは、すべての階層レベルでサポートされます。タスク・コンテナーは、階層の最下位レベルで登録されているプラグインを使用します。

以下のいずれかの方法で、プラグインを登録できます。

- タスク・モデルにプラグインを登録します。

  WebSphere Integration Developer のタスク・エディターで、タスクのプロパティー領域の「詳細」ページにある「**イベント・ハンドラー名 (Event handler name)**」フィールドにイベント・ハンドラーの名前を指定します。
- 臨時タスク用または実行時に作成するタスク・テンプレート用にプラグインを登録します。

  TTask クラスの setEventHandlerName メソッドを使用して、イベント・ハンドラーの名前を登録します。
- 実行時のタスク・インスタンスの登録済みイベント・ハンドラーを変更します。

  update(Task task) メソッドを使用して、実行時のタスク・インスタンスに異なるイベント・ハンドラーを使用します。呼び出し元は、このプロパティーを更新するために、タスク管理者権限を持っていなければなりません。
- グローバル・レベルでプラグインを登録します。

  ヒューマン・タスク・コンテナーの「カスタム・プロパティー」ページの管理コンソールで、プラグインのカスタム・プロパティーを定義します。カスタム・プロパティーの値はプラグイン名です。

HumanTaskManagerService インターフェース:

HumanTaskManagerService インターフェースは、ローカルまたはリモート・クライアントから呼び出すことができるタスク関連機能を公開します。

呼び出すことができるメソッドは、タスクの状態、およびそのメソッドが含まれているアプリケーションを使用する人物の権限によって異なります。タスク・オブジェクトを操作するための main メソッドを、以下にリストします。これらのメソッドおよび HumanTaskManagerService インターフェースで使用可能なその他のメソッドについての詳細は、com.ibm.task.api パッケージ内の Javadoc を参照してください。

タスク・テンプレート

タスク・テンプレートを扱う作業には、以下のメソッドが使用可能です。

表 26. タスク・テンプレート用の API メソッド

メソッド	説明
getTaskTemplate	指定されたタスク・テンプレートを取得します。
createAndCallTask	指定されたタスク・テンプレートからタスク・インスタンスを作成および実行し、同期式に結果を待機します。
createAndStartTask	指定されたタスク・テンプレートからタスク・インスタンスを作成および開始します。
createTask	指定されたタスク・テンプレートからタスク・インスタンスを作成します。
createInputMessage	指定されたタスク・テンプレート用の入力メッセージを作成します。例えば、タスクの開始に使用できるメッセージを作成します。
queryTaskTemplates	データベースに保管されているタスク・テンプレートを取得します。

タスク・インスタンス

タスク・インスタンスを扱う作業には、以下のメソッドが使用可能です。

表 27. タスク・インスタンス用の API メソッド

メソッド	説明
getTask	タスク・インスタンスを取得します。任意の状態のタスク・インスタンスを取得できます。
callTask	親タスクを同期で開始します。
startTask	既に作成済みのタスクを開始します。
suspend	ヒューマン・タスクまたは参加タスクを中断します。
resume	ヒューマン・タスクまたは参加タスクを再開します。
terminate	指定されたタスク・インスタンスを終了します。親タスクを終了する場合、このアクションは、呼び出されたサービスには影響しません。

表 27. タスク・インスタンス用の API メソッド (続き)

メソッド	説明
delete	指定されたタスク・インスタンスを削除します。
claim	処理のためタスクを要求します。
update	タスク・インスタンスを更新します。
complete	タスク・インスタンスを完了します。
cancelClaim	別の潜在的な所有者が処理できるように、要求されたタスク・インスタンスをリリースします。
createWorkItem	タスク・インスタンスの作業項目を作成します。
transferWorkItem	指定された所有者に作業項目を転送します。
deleteWorkItem	作業項目を削除します。

エスカレーション

エスカレーションを扱う作業には、以下のメソッドが使用可能です。

表 28. エスカレーションで使用できる API メソッド

メソッド	説明
getEscalation	指定されたエスカレーション・インスタンスを取得します。

カスタム・プロパティー

タスク、タスク・テンプレート、およびエスカレーションはすべてカスタム・プロパティーを持つことができます。このインターフェースは、カスタム・プロパティーの値を取得および設定するための get および set メソッドを提供します。指定されたプロパティーをプロセス・インスタンスおよびアクティビティー・インスタンスに関連付けたり、指定されたプロパティーをタスク・インスタンスから取得することもできます。カスタム・プロパティーの名前および値は、java.lang.String 型である必要があります。次のメソッドは、タスク、タスク・テンプレート、およびエスカレーションで有効です。

表 29. 変数およびカスタム・プロパティーの API メソッド

メソッド	説明
getCustomProperty	指定されたタスク・インスタンスの指定された 1 つのカスタム・プロパティーを取得します。
getCustomProperties	指定されたタスク・インスタンスのカスタム・プロパティー (複数) を取得します。
getCustomPropertyNames	タスク・インスタンスのすべてのカスタム・プロパティーの名前を取得します。
setCustomProperty	指定されたタスク・インスタンスのカスタム固有値を保管します。

各タスクで許可されるアクション:

タスクに対して実行可能なアクションは、そのタスクが参加タスク、純粋なヒューマン・タスク、親タスク、管理用タスクのどのタイプであるかによって異なります。

LocalHumanTaskManager または HumanTaskManager インターフェースによって提供されるすべてのアクションが、すべての種類のタスクに対して使用できるわけではありません。次の表に、タスクの種類ごとに実行可能なアクションを示します。

	タスクの種類			
アクション	**参加タスク**	**ヒューマン・タスク**	**親タスク**	**管理用タスク**
callTask			X[1]	
cancelClaim	X	X[1]		
claim	X	X[1]		
complete	X	X[1]		X
completeWithFollowOnTask[4]	X	X[1]		
completeWithFollowOnTask[5]		X[3]	X[3]	
createFaultMessage	X	X	X	X
createInputMessage	X	X	X	X
createOutputMessage	X	X	X	X
createWorkItem	X	X[1]	X	X
delete	X[1]	X[1]	X	X[1]
deleteWorkItem	X	X[1]	X	X
getCustomProperty	X	X[1]	X	X
getDocumentation	X	X[1]	X	X
getFaultNames	X	X[1]		
getFaultMessage	X	X[1]	X	
getInputMessage	X	X[1]	X	
getOutputMessage	X	X[1]	X	
getRoleInfo	X	X[1]	X	X
getTask	X	X[1]	X	X
getUISettings	X	X[1]	X	X
resume	X	X[1]		
setCustomProperty	X	X[1]	X	X
setFaultMessage	X	X[1]		
setOutputMessage	X	X[1]		
startTask	X[1]	X[1]	X	X
startTaskAsSubtask[6]	X	X[1]		
startTaskAsSubtask[7]		X[3]	X[3]	
suspend	X	X[1]		
suspendWithCancelClaim	X	X[1]		
terminate	X[1]	X[1]	X[1]	

アクション	タスクの種類			
	参加タスク	ヒューマン・タスク	親タスク	管理用タスク
transferWorkItem	X	X[1]	X	X
updateInactiveTask	X[2]	X[3]	X[2]	X[2]
updateTask	X	X[1]	X	X

注:

1. スタンドアロンのタスク、臨時のタスク、およびタスク・テンプレートの場合のみ。
2. スタンドアロンのタスク、ビジネス・プロセス内のインライン・タスク、および臨時のタスクの場合のみ
3. スタンドアロンのタスクおよび臨時のタスクの場合のみ。
4. 後続タスクを持つことのできるタスクの種類。
5. 後続タスクとして使用できるタスクの種類。
6. サブタスクを持つことのできるタスクの種類。
7. サブタスクとして使用できるタスクの種類。

例外および障害の処理

BPEL プロセスは、プロセスのさまざまな時点で障害に遭遇する可能性があります。

Business Process Execution Language (BPEL) 障害は、以下から発生します。

- Web サービス呼び出し (Web サービス記述言語 (WSDL) 障害)
- スロー (throw) アクティビティー
- Business Process Choreographer によって認識される BPEL 標準障害

これらの障害を処理する機構が存在します。プロセス・インスタンスによって生成される障害を処理するには、以下の手段のいずれかを使用します。

- 対応する障害ハンドラーへの制御の引き渡し
- プロセス内の前作業の補正
- プロセスの停止と状態の修復の依頼 (強制再試行、強制完了)

BPEL プロセスは、プロセスが指定する操作の呼び出し元へ障害を戻すこともできます。プロセスの障害を、障害名および障害データを持つ応答アクティビティーとしてモデル化することができます。これらの障害は、チェック例外として API 呼び出し元に戻されます。

BPEL プロセスが BPEL 障害を処理しない場合または API 例外が発生する場合は、実行時の例外が API 呼び出し元に戻されます。API 例外の例として、インスタンスの作成元のプロセス・モデルが存在しない場合があげられます。

障害および例外の処理は、以下のタスクで説明します。

API 例外の処理:

BusinessFlowManagerService インターフェースまたは HumanTaskManagerService インターフェースのメソッドが正常に完了しない場合、エラーの原因を示す例外がスローされます。この例外を特別に処理して、呼び出し元にガイダンスを提供することができます。

ただし、例外のサブセットのみを特別に処理し、その他の潜在的な例外に対しては汎用のガイダンスを提供するのが一般的です。すべての固有の例外は、汎用の ProcessException または TaskException から継承しています。最終の catch(ProcessException) または catch(TaskException) ステートメントを使用して汎用の例外を catch することが*最良実例* です。このステートメントでは、発生する可能性があるその他すべての例外を考慮に入れるため、このステートメントによって、ご使用のアプリケーション・プログラムの上位互換性を確保することができます。

アクティビティーに設定された障害の検査:

1. 失敗状態または停止状態のタスク・アクティビティーをリストします。

```
QueryResultSet result =
     process.query("ACTIVITY.AIID",
                   "(ACTIVITY.STATE = ACTIVITY.STATE.STATE_FAILED OR
                     ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED) AND
                     ACTIVITY.KIND=ACTIVITY.KIND.KIND_STAFF",
                     null, null, null);
```

   このアクションは、失敗または停止のアクティビティーが含まれる照会結果セットを戻します。

2. 障害の名前を読み取ります。

```
if (result.size() > 0)
{
  result.first();
  AIID aiid = (AIID) result.getOID(1);
  ClientObjectWrapper faultMessage = process.getFaultMessage(aiid);
  DataObject fault = null ;
  if ( faultMessage.getObject() != null && faultMessage.getObject()
       instanceof DataObject )
  {
    fault = (DataObject)faultMessage.getObject();
    Type type = fault.getType();
    String name = type.getName();
    String uri = type.getURI();
  }
}
```

   これは、障害名を戻します。また、障害名を取得する代わりに、停止アクティビティーの未処理の例外を分析することもできます。

停止した invoke アクティビティーで発生した障害の検査:
アクティビティーで障害が発生した場合、障害タイプによってそのアクティビティーの修復のために実行できるアクションが決まります。

1. 停止状態の staff アクティビティーをリストします。

```
QueryResultSet result =
     process.query("ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
                    ACTIVITY.KIND=ACTIVITY.KIND.KIND_INVOKE",
                    null, null, null);
```

このアクションは、停止された invoke アクティビティーが含まれる照会結果セットを戻します。

2. 障害の名前を読み取ります。

```
if (result.size() > 0)
{
  result.first();
  AIID aiid = (AIID) result.getOID(1);
  ActivityInstanceData activity = process.getActivityInstance(aiid);

  ProcessException excp = activity.getUnhandledException();
  if ( excp instanceof ApplicationFaultException )
  {
   ApplicationFaultException fault = (ApplicationFaultException)excp;
   String faultName = fault.getFaultName();
  }
}
```

Web サービス API クライアント・アプリケーションの開発

Web サービス API を介してビジネス・プロセス・アプリケーションとヒューマン・タスク・アプリケーションにアクセスするクライアント・アプリケーションを開発できます。

クライアント・アプリケーションは、任意の Web サービス・クライアント環境 (Java Web サービス、Microsoft .NET など) で開発できます。

概要: Web サービス

Web サービスは、クライアント・アプリケーションとデータを交換するために、オープンな XML ベース標準およびトランスポート・プロトコルを使用する Web ベースのエンタープライズ・アプリケーションです。 Web サービスにより、言語および環境に中立なプログラミング・モデルを使用することができます。

Web サービスは次のコア・テクノロジーを使用します。

- XML (Extensible Markup Language)。 XML は、データ独立性の問題を解決します。これを使用してデータを記述し、さらにアプリケーションまたはプログラミング言語との間でデータをマップします。
- WSDL (Web サービス記述言語)。この XML ベースの言語を使用して、基礎となるアプリケーションの記述を作成します。この記述は、基礎となるアプリケーションとその他の Web 対応アプリケーションとの間のインターフェースとして機能することにより、アプリケーションを Web サービスに変換します。
- SOAP (Simple Object Access Protocol)。SOAP は、Web 用のコアとなる通信プロトコルで、ほとんどの Web サービスはこのプロトコルを使用して相互に対話します。

Web サービス・コンポーネントおよび一連の制御

多くのクライアント・サイドおよびサーバー・サイドのコンポーネントは、 Web サービスの要求と応答を表す一連の制御に関与します。

標準的な一連の制御は以下のとおりです。

1. クライアント・サイド:
   a. クライアント・アプリケーション (ユーザーによって提供される) は、Web サービスの要求を発行します。

b. プロキシー・クライアント (ユーザーによっても提供されるが、クライアント・サイド・ユーティリティーを使用して自動的に生成することが可能) は、サービス要求を SOAP 要求エンベロープでラップします。

c. クライアント・サイド開発インフラストラクチャーは、Web サービスのエンドポイントとして定義された URL に要求を転送します。

2. ネットワークは、HTTP または HTTPS を使用して Web サービス・エンドポイントに要求を送信します。

3. サーバー・サイド:

a. 汎用 Web サービス API は、要求を受信し、デコードします。

b. 要求は、Business Flow Manager または Human Task Manager の汎用コンポーネントによって直接処理されるか、指定されたビジネス・プロセスまたはヒューマン・タスクに転送されます。

c. 戻されたデータは SOAP 応答エンベロープでラップされます。

4. ネットワークは、HTTP または HTTPS を使用してクライアント・サイド環境に応答を送信します。

5. クライアント・サイドに戻る:

a. クライアント・サイド開発インフラストラクチャーは、SOAP 応答エンベロープをアンラップします。

b. プロキシー・クライアントはデータを SOAP 応答から抽出して、それをクライアント・アプリケーションに受け渡します。

c. クライアント・アプリケーションは、必要に応じて、戻されたデータを処理します。

Web サービス API の概要

Web サービス API により、Business Process Choreographer 環境で実行しているビジネス・プロセスおよびヒューマン・タスクに Web サービスを使用してアクセスするクライアント・アプリケーションを開発できます。

Business Process Choreographer Web サービス API には、次の 2 つの別個の Web サービス・インターフェース (WSDL ポート・タイプ) があります。

- Business Flow Manager API。クライアント・アプリケーションが microflow および長期間のプロセスと対話できるようにします。例えば、以下の操作を実行できます。
  - プロセス・テンプレートとプロセス・インスタンスの作成
  - 既存のプロセスの要求
  - ID によるプロセスの照会

  実行可能なアクションの詳細なリストについては、234 ページの『ビジネス・プロセス用のアプリケーションの開発』を参照してください。
- Human Task Manager API。クライアント・アプリケーションは以下の操作を実行できます。
  - タスクの作成と開始
  - 既存のタスクの要求
  - タスクの完了

– ID によるタスクの照会
– タスクの集合の照会

実行可能なアクションの詳細なリストについては、257 ページの『ヒューマン・タスク用のアプリケーションの開発』を参照してください。

クライアント・アプリケーションは、Web サービス・インターフェースのいずれかまたは両方を使用できます。

例

Human Task Manager Web サービス API にアクセスして、関係するヒューマン・タスクを処理するクライアント・アプリケーションの概要としては以下のようなものが考えられます。

1. クライアント・アプリケーションは、`query(...)` Web サービス呼び出しを WebSphere Process Server に対して発行します。これにより、ユーザーによって処理される参加タスクのリストを要求します。
2. 参加タスクのリストが、SOAP/HTTP 応答エンベロープで戻されます。
3. クライアント・アプリケーションは、`claim(...)` Web サービス呼び出しを発行して、いずれかの参加タスクを要求します。
4. WebSphere Process Server は、タスクの入力メッセージを戻します。
5. クライアント・アプリケーションは、`complete(...)` Web サービス呼び出しを発行して、出力メッセージまたは障害メッセージのあるタスクを完了します。

ビジネス・プロセスおよびヒューマン・タスクの要件

Business Process Choreographer 上で実行するように WebSphere Integration Developer で開発されたビジネス・プロセスとヒューマン・タスクが、Web サービス API によってアクセス可能となるためには、特定の規則に準拠する必要があります。

要件は以下のとおりです。

1. ビジネス・プロセスとヒューマン・タスクのインターフェースは、 Java API for XML-based RPC (JAX-RPC 1.1) 仕様で定義された「document/literal wrapped」スタイルを使用して定義する必要があります。これは、WID で開発するすべてのビジネス・プロセスとヒューマン・タスクのデフォルト・スタイルです。
2. Web サービス・オペレーションのビジネス・プロセスとヒューマン・タスクによって出される障害メッセージは、 XML スキーマ・エレメントで定義された単一の WSDL メッセージ部分で構成されている必要があります。以下に例を示します。

   ```
   <wsdl:part name="myFault" element="myNamespace:myFaultElement"/>
   ```

   関連情報

   Java API for XML based RPC (JAX-RPC) のダウンロード・ページ

   使用すべき WSDL のスタイル

クライアント・アプリケーションの開発

クライアント・アプリケーションの開発プロセスは多数のステップで構成されています。

1. クライアント・アプリケーションが使用する必要のある Web サービス API を、 Business Flow Manager API、Human Task Manager API、またはその両方のいずれかに決定します。
2. WebSphere Process Server 環境から必要なファイルをエクスポートします。 あるいは、WebSphere Process Server クライアント CD からファイルをコピーすることもできます。
3. 選択したクライアント・アプリケーション開発環境で、エクスポートした成果物を使用してプロキシー・クライアントを生成します。
4. **オプション:** ヘルパー・クラスを生成します。 ヘルパー・クラスが必要になるのは、クライアント・アプリケーションが、WebSphere サーバー上の具象プロセスまたはタスクと直接対話する場合です。ただし、クライアント・アプリケーションが汎用タスク (照会の発行など) を実行するだけの場合は、ヘルパー・クラスは不要です。
5. クライアント・アプリケーション用のコードを開発します。
6. 必要なセキュリティー・メカニズムをクライアント・アプリケーションに追加します。

成果物のコピー

クライアント・アプリケーションの作成を補助するために、いくつかの成果物を WebSphere 環境からコピーする必要があります。

成果物を取得するには、以下の 2 つの方法があります。

- WebSphere Process Server 環境から成果物を公開およびエクスポートします。
- WebSphere Process Server クライアント CD からファイルをコピーします。

サーバー環境からの成果物の公開およびエクスポート:

クライアント・アプリケーションを開発して Web サービス API にアクセスする前に、WebSphere サーバー環境から、いくつかの成果物を公開およびエクスポートする必要があります。

エクスポート対象の成果物は、以下のとおりです。

- Web サービス API を構成するポート・タイプおよび操作を記述する Web サービス記述言語 (WSDL) ファイル。
- WSDL ファイルのサービスおよびメソッドによって参照されるデータ型の定義を含む XML スキーマ定義 (XSD)。
- ビジネス・オブジェクトを記述するその他の WSDL および XSD ファイル。ビジネス・オブジェクトは、WebSphere サーバーで実行する具象ビジネス・プロセスまたはヒューマン・タスクを記述します。このその他のファイルは、クライアント・アプリケーションが Web サービス API を介して具象ビジネス・プロセスまたはヒューマン・タスクに直接対話する必要がある場合のみ必要です。クライアント・アプリケーションが汎用タスク (照会の発行など) を実行するだけの場合は、これらは不要です。

これらの成果物は、公開された後、ご使用のクライアント・プログラミング環境にコピーする必要があります。それらは、プロキシー・クライアントおよびヘルパー・クラスを生成するために使用されます。

Web サービス・エンドポイント・アドレスの指定:

Web サービス・エンドポイント・アドレスは、クライアント・アプリケーションが Web サービス API にアクセスするために指定する必要のある URL です。エンドポイント・アドレスは、クライアント・アプリケーション用のプロキシー・クライアントを生成するためにエクスポートする WSDL ファイルに書き込まれます。

使用する Web サービス・エンドポイント・アドレスは、WebSphere サーバーの構成によって異なります。

- シナリオ 1. WebSphere サーバーが 1 台だけの場合。指定する WebSphere エンドポイント・アドレスは、サーバーのホスト名とポート番号です (例: **host1:9080**)。
- シナリオ 2. 複数のサーバーで構成される WebSphere クラスターの場合。指定する WebSphere エンドポイント・アドレスは、Web サービス API をホスティングするサーバーのホスト名とポートです (例: **host2:9081**)。
- シナリオ 3. Web サーバーをフロントエンドとして使用する場合。指定する WebSphere エンドポイント・アドレスは、Web サーバーのホスト名とポートです (例: **host:80**)。

デフォルトでは、Web サービス・エンドポイント・アドレスの形式は *protocol*:**//***host*:*port*/*context_root*/*fixed_path* です。各部の意味は、次のとおりです。

- *protocol*。クライアント・アプリケーションと WebSphere サーバー間で使用される通信プロトコル。デフォルト・プロトコルは HTTP です。代わりに、もっと安全な HTTPS (HTTP over SSL) プロトコルを使用することもできます。HTTPS を使用することをお勧めします。
- *host:port*。Web サービス API をホスティングするマシンへのアクセスに使用するホスト名とポート番号。この値は、クライアント・アプリケーションがアプリケーションに直接アクセスするか、Web サーバー・フロントエンド経由でアクセスするかなど、WebSphere サーバー構成によって異なります。
- *context_root*。コンテキスト・ルートには、任意の値を選択できます。ただし、選択する値は個々の WebSphere セル内で固有でなければなりません。デフォルト値は、「node_server/cluster」サフィックスを使用して、ネーミング競合のリスクを回避しています。
- *fixed_path* は、`/sca/com/ibm/bpe/api/BFMWS` (Business Flow Manager API の場合) または `/sca/com/ibm/task/api/HTMWS` (Human Task Manager API の場合) のどちらかで、変更することはできません。

Web サービス・エンドポイント・アドレスは、最初は、ビジネス・プロセス・コンテナーまたはヒューマン・タスク・コンテナーの構成時に指定されます。

1. 管理者権限のあるユーザー ID で、管理コンソールにログオンします。
2. 「**アプリケーション**」 → 「**SCA モジュール**」を選択します。

注: 「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」を選択して、使用可能なすべてのエンタープライズ・アプリケーションのリストを表示することもできます。

3. SCA モジュールまたはアプリケーションのリストから、「**BPEContainer**」(ビジネス・プロセス・コンテナーの場合) または「**TaskContainer**」(ヒューマン・タスク・コンテナーの場合) を選択します。
4. 「**追加プロパティー**」リストから、「**HTTP エンドポイント URL 情報を提供 (Provide HTTP endpoint URL information)**」を選択します。
5. リストからデフォルトのプレフィックスのいずれか 1 つを選択するか、カスタム・プレフィックスを入力します。 クライアント・アプリケーションを、Web サービス API をホスティングするアプリケーション・サーバーに直接接続する場合は、デフォルト・プレフィックス・リストのプレフィックスを使用します。そうでない場合は、カスタム・プレフィックスを指定します。
6. 「**適用**」をクリックして、選択したプレフィックスを SCA モジュールにコピーします。
7. 「**OK**」をクリックします。 URL 情報がワークスペースに保管されます。

管理コンソールで現在の値を表示できます (例えばビジネス・プロセス・コンテナーの場合は、「**エンタープライズ・アプリケーション**」 → 「**BPEContainer**」 → 「**デプロイメント記述子の表示**」)。

エクスポートされた WSDL ファイルでは、`soap:address` エレメントの `location` 属性に、指定した Web サービス・エンドポイント・アドレスが含まれています。以下に例を示します。

```
<wsdl:service name="BFMWSService">
  <wsdl:port name="BFMWSPort" binding="this:BFMWSBinding">
   <soap:address location=
         "https://myserver:9080/WebServicesAPIs/sca/com/ibm/bpe/api/BFMWS"/>
```

関連概念

304 ページの『セキュリティーの追加 (Java Web サービス)』
クライアント・アプリケーションにセキュリティー・メカニズムを実装することにより、 Web サービス通信を保護する必要があります。

関連タスク

311 ページの『セキュリティーの追加 (.NET)』
Web サービス通信は、クライアント・アプリケーションにセキュリティー・メカニズムを 組み込むことにより保護できます。

WSDL ファイルの公開:

Web サービス記述言語 (WSDL) ファイルには、Web サービス API で使用できるすべての操作の詳細な説明が含まれています。Business Flow Manager Web サービス API と Human Task Manager Web サービス API では、使用できる WSDL ファイルが異なります。これらの WSDL ファイルは、まず公開して、その後 WebSphere 環境からご使用の開発環境にコピーし、プロキシー・クライアントの生成に使用することになります。

WSDL ファイルを公開する前に、指定した Web サービス・エンドポイント・アドレスが正しいことを確認してください。このアドレスは、クライアント・アプリケーションが Web サービス API へのアクセスに使用する URL です。

WSDL ファイルの公開が必要なのは一度だけです。

注: WebSphere Process Server クライアント CD を所有している場合は、代わりに、その CD からクライアント・プログラミング環境に直接ファイルをコピーすることもできます。

ビジネス・プロセス WSDL の公開:

管理コンソールを使用して、WSDL ファイルを公開します。

1. 管理者権限のあるユーザー ID で、管理コンソールにログオンします。
2. 「**アプリケーション**」 → 「**SCA モジュール**」を選択します。

   注: 「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」を選択して、使用可能なすべてのエンタープライズ・アプリケーションのリストを表示することもできます。
3. SCA モジュールまたはアプリケーションのリストから **BPEContainer** アプリケーションを選択します。
4. 「**追加プロパティー**」のリストから「**WSDL ファイルの公開 (Publish WSDL files)**」を選択します。
5. リスト内の zip ファイルをクリックします。
6. 表示された「ファイルのダウンロード (File Download)」ウィンドウで、「**保管**」をクリックします。
7. ローカル・フォルダーを参照し、「**保管**」をクリックします。

エクスポートされた zip ファイルの名前は BPEContainer_WSDLFiles.zip です。 zip ファイルには、Web サービスを記述する WSDL ファイルと、その WSDL ファイル内から参照される XSD ファイルが含まれています。

ヒューマン・タスク WSDL の公開:

管理コンソールを使用して、WSDL ファイルを公開します。

1. 管理者権限のあるユーザー ID で、管理コンソールにログオンします。
2. 「**アプリケーション**」 → 「**SCA モジュール**」を選択します。

   注: 「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」を選択して、使用可能なすべてのエンタープライズ・アプリケーションのリストを表示することもできます。
3. SCA モジュールまたはアプリケーションのリストから **TaskContainer** アプリケーションを選択します。
4. 「**追加プロパティー**」のリストから「**WSDL ファイルの公開 (Publish WSDL files)**」を選択します。
5. リスト内の zip ファイルをクリックします。
6. 表示された「ファイルのダウンロード (File Download)」ウィンドウで、「**保管**」をクリックします。

7. ローカル・フォルダーを参照し、「**保管**」をクリックします。

エクスポートされた zip ファイルの名前は TaskContainer_WSDLFiles.zip です。 zip ファイルには、Web サービスを記述する WSDL ファイルと、その WSDL ファイル内から参照される XSD ファイルが含まれています。

ビジネス・オブジェクトのエクスポート:

ビジネス・プロセスおよびヒューマン・タスクには、Web サービスとして外部にアクセスできるよう明確に定義されたインターフェースが用意されています。これらのインターフェースがビジネス・オブジェクトを参照する場合は、インターフェース定義とビジネス・オブジェクトをクライアント・プログラミング環境にエクスポートする必要があります。

この手順は、クライアント・アプリケーションが対話する必要があるビジネス・オブジェクトごとに、繰り返してください。

WebSphere Process Server では、ビジネス・オブジェクトが、ビジネス・プロセスまたはヒューマン・タスクと対話する要求、応答、および障害メッセージの形式を定義します。これらのメッセージに、複合データ・タイプの定義を含めることもできます。

例えば、ヒューマン・タスクを作成して開始するには、以下の情報項目をタスク・インターフェースに渡す必要があります。

- タスク・テンプレート名
- タスク・テンプレート・ネームスペース
- 入力メッセージ (フォーマット済みのビジネス・データを含む)
- 応答メッセージを戻すための応答ラッパー
- 障害および例外を戻すための障害メッセージ

以上の項目は、単一のビジネス・オブジェクト内でカプセル化されます。Web サービス・インターフェースのすべての操作は、「document/literal wrapped」操作としてモデル化されます。これらの操作の入出力パラメーターは、ラッパー文書でカプセル化されます。他のビジネス・オブジェクトは、それに対応する応答および障害のメッセージ形式を定義します。

Web サービスを介してビジネス・プロセスまたはヒューマン・タスクを作成および開始するためには、これらのラッパー・オブジェクトがクライアント・サイドのクライアント・アプリケーションで使用可能になる必要があります。

そのためには、WebSphere 環境からビジネス・オブジェクトを Web サービス記述言語 (WSDL) ファイルおよび XML スキーマ定義 (XSD) ファイルとしてエクスポートし、データ・タイプ定義をクライアント・プログラミング環境にインポートしてから、それをクライアント・アプリケーションで使用できるヘルパー・クラスに変換します。

1. WebSphere Integration Developer (WID) ワークスペースが稼働していない場合は、起動します。

2. エクスポートするビジネス・オブジェクトを含むライブラリー・モジュールを選択します。 ライブラリー・モジュールは、必要なビジネス・オブジェクトが入っている圧縮ファイルです。
3. ライブラリー・モジュールをエクスポートします。
4. エクスポートしたファイルをクライアント・アプリケーション開発環境にコピーします。

例

ビジネス・プロセスが以下の Web サービス操作を公開するとします。

```
<wsdl:operation name="updateCustomer">
    <wsdl:input message="tns:updateCustomerRequestMsg"
                name="updateCustomerRequest"/>
    <wsdl:output message="tns:updateCustomerResponseMsg"
                 name="updateCustomerResponse"/>
    <wsdl:fault message="tns:updateCustomerFaultMsg"
                name="updateCustomerFault"/>
  </wsdl:operation>
```

公開は、以下のように定義された WSDL メッセージを介して行われるとします。

```
<wsdl:message name="updateCustomerRequestMsg">
    <wsdl:part element="types:updateCustomer"
               name="updateCustomerParameters"/>
  </wsdl:message>
  <wsdl:message name="updateCustomerResponseMsg">
    <wsdl:part element="types:updateCustomerResponse"
               name="updateCustomerResult"/>
  </wsdl:message>
  <wsdl:message name="updateCustomerFaultMsg">
    <wsdl:part element="types:updateCustomerFault"
               name="updateCustomerFault"/>
  </wsdl:message>
```

具象 ユーザー定義エレメント「types:updateCustomer」、「types:updateCustomerResponse」、および「types:updateCustomerFault」は、クライアント・アプリケーションが実行するすべての*汎用* 命令 (**call**、**sendMessage** など) で、<xsd:any> パラメーターを使用して Web サービス API に渡したり Web サービス API から受け取ったりする必要があります。これらのユーザー定義エレメントは、エクスポートされた XSD ファイルから生成されるヘルパー・クラスを使用して、クライアント・アプリケーション上で作成、直列化、および非直列化されます。

関連タスク

308 ページの『BPEL プロセス用ヘルパー・クラスの作成 (.NET)』
特定の Web サービス API 操作では、クライアント・アプリケーションが「document/literal」スタイルのラップ・エレメントを使用する必要があります。クライアント・アプリケーションは、 ヘルパー・クラスに、必要なラッパー・エレメントの生成を担当させます。

クライアント CD のファイルの使用:

WebSphere サーバー環境からの成果物をエクスポートする代わりに、WebSphere Process Server クライアント CD からクライアント・アプリケーションの生成に必要なファイルをコピーできます。

この場合は、Business Flow Manager API または Human Task Manager API のデフォルト Web サービス・エンドポイント・アドレスを手動で変更する必要があります。

クライアント・アプリケーションが両方の API にアクセスする場合は、両方の API のデフォルト・エンドポイント・アドレスを編集する必要があります。

クライアント CD からファイルのコピー:

Web サービス API にアクセスするのに必要なファイルは、WebSphere Process Server クライアント CD で入手できます。

1. クライアント CD にアクセスして、`ProcessChoreographer¥client` ディレクトリーを参照します。
2. 必要なファイルをクライアント・アプリケーション開発環境にコピーします。

   Business Flow Manager API の場合、以下をコピーします。

   BFMWS.wsdl
   Business Flow Manager Web サービス API で使用できる Web サービスを記述します。このファイルには、エンドポイント・アドレスが含まれています。

   BFMIF.wsdl
   Business Flow Manager Web サービス API の各 Web サービスのパラメーターおよびデータ型を記述します。

   BFMIF.xsd
   Business Flow Manager Web サービス API で使用されるデータ型を記述します。

   BPCGEN.xsd
   Business Flow Manager と Human Task Manager の Web サービス API 間で共通するデータ型を含みます。

   Human Task Manager API の場合は、以下をコピーします。

   HTMWS.wsdl
   Human Task Manager Web サービス API で使用できる Web サービスを記述します。このファイルには、エンドポイント・アドレスが含まれています。

   HTMIF.wsdl
   Human Task Manager Web サービス API の各 Web サービスのパラメーターおよびデータ型を記述します。

   HTMIF.xsd
   Human Task Manager Web サービス API で使用されるデータ型を記述します。

   BPCGEN.xsd
   Business Flow Manager と Human Task Manager の Web サービス API 間で共通するデータ型を含みます。

   注: BPCGen.xsd ファイルは、両方の API に共通です。

ファイルをコピーした後、BFMWS.wsdl または HTMWS.wsdl ファイルの Web サービス API エンドポイント・アドレスを、Web サービス API をホスティングする WebSphere アプリケーション・サーバーのアドレスに手動で変更する必要があります。

Web サービス・エンドポイント・アドレスの手動変更:

クライアント CD からファイルをコピーする場合は、WSDL ファイルで指定したデフォルトの Web サービス・エンドポイント・アドレスを、Web サービス API をホスティングするサーバーのアドレスに変更する必要があります。

管理コンソールを使用すると、WSDL ファイルをエクスポートする前に Web サービス・エンドポイント・アドレスを設定することができます。ただし、WSDL ファイルを WebSphere Process Server クライアント CD からコピーする場合は、デフォルトの Web サービス・エンドポイント・アドレスを手動で変更する必要があります。

使用する Web サービス・エンドポイント・アドレスは、WebSphere サーバーの構成によって異なります。

- シナリオ 1. WebSphere サーバーが 1 台だけの場合。指定する WebSphere エンドポイント・アドレスは、サーバーのホスト名とポート番号です (例: **host1:9080**)。
- シナリオ 2. 複数のサーバーで構成される WebSphere クラスターの場合。指定する WebSphere エンドポイント・アドレスは、Web サービス API をホスティングするサーバーのホスト名とポートです (例: **host2:9081**)。
- シナリオ 3. Web サーバーをフロントエンドとして使用する場合。指定する WebSphere エンドポイント・アドレスは、Web サーバーのホスト名とポートです (例: **host:80**)。

Business Flow Manager API エンドポイントの変更:

WebSphere Process Server クライアント CD から Business Flow Manager API ファイルをコピーする場合は、デフォルトのエンドポイント・アドレスを手動で編集する必要があります。

1. クライアント CD からコピーされたファイルを含むディレクトリーへナビゲートします。
2. テキスト・エディターまたは XML エディターで BFMWS.wsdl ファイルを開きます。
3. `soap:address` エレメントを検索します (ファイルの最後の方にあります)。
4. `location` 属性の値を、Web サービス API が実行しているサーバーの HTTP URL で変更します。 これを行うには、次のようにします。
   a. オプションで、より機密保護機能のある HTTPS プロトコルを使用するために `http` を `https` で置換します。
   b. `localhost` を、Web サービス API サーバー・エンドポイント・アドレスのホスト名または IP アドレスで置換します。
   c. `9080` をアプリケーション・サーバーのポート番号で置換します。

d. BPEContainer_N1_server1 を、Web サービス API を実行しているアプリケーションの「コンテキスト・ルート」で置換します。 デフォルト・コンテキスト・ルートは以下で構成されます。
   - BPEContainer。アプリケーション名。
   - N1。ノード名。
   - server1。サーバー名。

e. URL の固定部分 (/sca/com/ibm/bpe/api/BFMWS) は変更しないでください。

例えば、アプリケーションがサーバー **s1.n1.ibm.com** で実行していて、サーバーがポート **9080** で SOAP/HTTP 要求を受け入れている場合、<soap:address> エレメントを以下のように変更します。

```
<soap:address location="http://si.n1.ibm.com:9080/
                  BPEContainer_N1_server1/sca/com/ibm/bpe/api/BFMWS"/>
```

関連概念

304 ページの『セキュリティーの追加 (Java Web サービス)』
クライアント・アプリケーションにセキュリティー・メカニズムを実装することにより、 Web サービス通信を保護する必要があります。

関連タスク

311 ページの『セキュリティーの追加 (.NET)』
Web サービス通信は、クライアント・アプリケーションにセキュリティー・メカニズムを 組み込むことにより保護できます。

Human Task Manager API エンドポイントの変更:

WebSphere Process Server クライアント CD から Human Task Manager API ファイルをコピーする場合は、デフォルトのエンドポイント・アドレスを手動で編集する必要があります。

1. クライアント CD からコピーされたファイルを含むディレクトリーへナビゲートします。
2. テキスト・エディターまたは XML エディターで HTMWS.wsdl ファイルを開きます。
3. soap:address エレメントを検索します (ファイルの最後の方にあります)。
4. location 属性の値を、正しいエンドポイント・アドレスで置換します。 これを行うには、次のようにします。
   a. オプションで、より機密保護機能のある HTTPS プロトコルを使用するために http を https で置換します。
   b. localhost を、Web サービス API サーバーのエンドポイント・アドレスのホスト名または IP アドレスで置換します。
   c. 9080 をアプリケーション・サーバーのポート番号で置換します。
   d. HTMContainer_N1_server1 を、Web サービス API を実行しているアプリケーションの「コンテキスト・ルート」で置換します。 デフォルト・コンテキスト・ルートは以下で構成されます。
      - HTMContainer。アプリケーション名。
      - N1。ノード名。
      - server1。サーバー名。

e. URL の固定部分 (/sca/com/ibm/task/api/HTMWS) は変更しないでください。

例えば、アプリケーションがサーバー **s1.n1.ibm.com** で実行していて、サーバーがポート **9081** で SOAP/HTTP 要求を受け入れている場合、<soap:address> を以下のように変更します。

```
<soap:address location="https://si.n1.ibm.com:9081/
                  HTMContainer_N1_server1/sca/com/ibm/task/api/HTMWS"/>
```

関連概念

304 ページの『セキュリティーの追加 (Java Web サービス)』
クライアント・アプリケーションにセキュリティー・メカニズムを実装することにより、 Web サービス通信を保護する必要があります。

関連タスク

311 ページの『セキュリティーの追加 (.NET)』
Web サービス通信は、クライアント・アプリケーションにセキュリティー・メカニズムを 組み込むことにより保護できます。

Java Web サービス環境でのクライアント・アプリケーションの開発

Java Web サービスと互換性のある Java ベースの開発環境を使用して、Web サービス API 用のクライアント・アプリケーションを開発できます。

プロキシー・クライアントの生成 (Java Web サービス):

Java Web サービス・クライアント・アプリケーションは、*プロキシー・クライアント* を使用して Web サービス API と対話します。

Java Web サービスのプロキシー・クライアントには、Web サービス要求を実行するためにクライアント・アプリケーションが呼び出す、いくつかの Java Bean クラスが含まれています。プロキシー・クライアントは、サービス・パラメーターの SOAP メッセージへのアセンブリーを処理し、その SOAP メッセージを HTTP 経由で Web サービスに送信し、Web サービスから応答を受信して、戻されたデータをクライアント・アプリケーションに渡します。

したがって、基本的にプロキシー・クライアントは、クライアント・アプリケーションが Web サービスをローカル機能のように呼び出せるようにするためのものです。

注: プロキシー・クライアントの生成が必要なのは一度だけです。同じ Web サービス API にアクセスするクライアント・アプリケーションはすべて、以後は同じプロキシー・クライアントを使用できます。

IBM Web サービス環境でプロキシー・クライアントを生成するには、以下の 2 つの方法があります。

- Rational® Application Developer または WebSphere Integration Developer の統合開発環境を使用する。
- WSDL2Java コマンド行・ツールを使用する。

他の Java Web サービス開発環境には通常、WSDL2Java ツールまたは独自のクライアント・アプリケーション生成機能が含まれます。

Rational Application Developer によるプロキシー・クライアントの生成:

Rational Application Developer の統合開発環境では、クライアント・アプリケーション用のプロキシー・クライアントを生成できます。

プロキシー・クライアントを生成するには、その前に、ビジネス・プロセスまたはヒューマン・タスクの Web サービス・インターフェースを記述した WSDL ファイルを WebSphere 環境 (または WebSphere Process Server クライアント CD) からエクスポートし、それをクライアントのプログラミング環境にコピーしておく必要があります。

1. 該当する WSDL ファイルをプロジェクトに追加します。
   - ビジネス・プロセスのための BFMWS.WSDL
   - ヒューマン・タスクのための HTMWS.WSDL
2. Web サービス・ウィザード・プロパティーを変更します。
   a. Rational Application Developer で、「**設定**」 → 「**Web サービス**」 → 「**コード生成**」 → 「**IBM Websphere ランタイム (IBM WebSphere runtime)**」を選択します。
   b. 「**ラップ・スタイルを使用せずに WSDL から Java を生成 (Generate Java from WSDL using the no wrapped style)**」オプションを選択します。
3. BFMWS.WSDL ファイルまたは HTMWS.WSDL ファイルを選択します。
4. クライアント・サイドのコンテナー・タイプを選択します。**client** (クライアント・コンテナー)、**ejb** (EJB コンテナー)、**java** (Java コンテナー) 、**none** のいずれかです。
5. 右クリックして、「**Web サービス**」 → 「**クライアントの生成 (Generate client)**」を選択します。
6. 表示される指示に従います。

複数のプロキシー、ロケーター、ヘルパー Java クラスで構成されるプロキシー・クライアントが生成され、プロジェクトに追加されます。デプロイメント記述子も更新されます。

WSDL2Java によるプロキシー・クライアントの生成:

WSDL2Java は、プロキシー・クライアントを生成するコマンド行ツールです。プロキシー・クライアントによって、クライアント・アプリケーションのプログラミングが容易になります。

プロキシー・クライアントを生成するには、その前に、ビジネス・プロセスまたはヒューマン・タスクの Web サービス API を記述した WSDL ファイルを WebSphere 環境 (または WebSphere Process Server クライアント CD) からエクスポートし、それをクライアントのプログラミング環境にコピーしておく必要があります。

1. WSDL2Java ツールを使用してプロキシー・クライアントを生成します。 タイプ :

   wsdl2java *options WSDLfilepath*

   各部の意味は、次のとおりです。
   - オプション は以下のとおりです。

-noWrappedOperations (-w)
ラップ操作の検出を使用不可にします。要求および応答メッセージ用の Java Bean が生成されます。

注: これはデフォルト値ではありません。

-role (-r)
値 **client** を指定すると、クライアント・サイド開発用のファイルおよびバインディング・ファイルが生成されます。

-container (-c)
使用するクライアント・サイド・コンテナー。有効な引数は以下のとおりです。

クライアント
クライアント・コンテナー

ejb Enterprise JavaBeans (EJB) コンテナー。

none コンテナーなし

web Web コンテナー

-output (-o)
生成されたファイルを保管するフォルダー。

WSDL2Java パラメーターの完全なリストが必要な場合は、**/help** コマンド行スイッチを使用するか、WID/RAD で WSDL2Java ツールのオンライン・ヘルプを参照してください。

- *WSDLfilepath* は、WebSphere 環境からエクスポートしたか、クライアント CD からコピーした WSDL ファイルのパスおよびファイル名です。

次の例では、ヒューマン・タスク・アクティビティー Web サービス API 用のプロキシー・クライアントが生成されます。

```
call wsdl2java.bat -r client -c client -noWrappedOperations
                   -output c:¥ws¥proxyClient  c:¥ws¥bin¥HTMWS.wsdl
```

2. 生成されたクラス・ファイルをプロジェクトに組み込みます。

BPEL プロセス用ヘルパー・クラスの作成 (Java Web サービス):

具象 API 要求 (sendMessage 、call など) で参照されるビジネス・オブジェクトでは、クライアント・アプリケーションが「document/literal wrapped」スタイル・エレメントを使用する必要があります。クライアント・アプリケーションは、ヘルパー・クラスに、必要なラッパー・エレメントの生成を担当させます。

ヘルパー・クラスを作成するには、Web サービス API の WSDL ファイルを、WebSphere Process Server 環境からエクスポートしておく必要があります。

Web サービス API の call() 命令や sendMessage() 命令を実行すると、WebSphere Process Server 上で、BPEL プロセスとの対話ができるようになります。call() 命令の入力メッセージは、プロセスの入力メッセージの document/literal ラッパーが提供されると予想します。

BPEL プロセスまたはヒューマン・タスク用のヘルパー・クラスを生成するには、次のようないくつかの技法があります。

1. SoapElement オブジェクトを使用します。

   WebSphere Integration Developer で使用可能な Rational Application Developer 環境では、Web サービス・エンジンが JAX-RPC 1.1 をサポートします。JAX-RPC 1.1 では、SoapElement オブジェクトが Document Object Model (DOM) エレメントを拡張するため、DOM API を使用した SOAP メッセージの作成、読み取り、ロード、および保管が可能になりました。

   例えば、ワークフロー・プロセスまたはヒューマン・タスク用として以下の入力メッセージが WSDL ファイルに含まれることを想定してみます。

   ```
   <xsd:element name="operation1">
           <xsd:complexType>
             <xsd:sequence>
               <xsd:element name="input1" nillable="true" type="xsd:string"/>
             </xsd:sequence>
           </xsd:complexType>
         </xsd:element>
   ```

   DOM API を使用して対応する SOAP を作成するには、次のようにします。

   ```
   SOAPFactory soapfactoryinstance = SOAPFactory.newInstance();
   SOAPElement soapmessage = soapfactoryinstance.createElement
                             ("operation1", namespaceprefix, interfaceURI);
   SOAPElement inputelement = soapfactoryinstance.createElement("input1");
   inputelement.addTextNode( message value);
   soapmessage.addChildElement(outputelement);
   ```

   以下の例では、sendMessage 操作用の入力パラメーターを作成する方法を示します。

   ```
   SendMessage inWsend = new SendMessage();
   inWsend.setProcessTemplateName(processtemplatename);
   inWsend.setPortType(porttype);
   inWsend.setOperation(operationname);
   inWsend.set_any(soapmessage);
   ```

2. WebSphere Custom Data Binding 機能を使用します。

   この技法は、以下の developerWorks 記事に記載されています。

   - How to choose a custom mapping technology for Web services
   - Developing Web Services with EMF SDOs for complex XML schema

   Interoperability With Patterns and Strategies for Document-Based Web Services

   Web Services support for Schema/WSDL(s) containing optional JAX-RPC 1.0/1.1 XML Schema Types

クライアント・アプリケーションの作成 (Java Web サービス):

クライアント・アプリケーションは、Web サービス API に要求を送信し、Web サービス API からの応答を受信します。プロキシー・クライアントを使用して、複合データ・タイプのフォーマット設定を行う通信クラスおよびヘルパー・クラスを管理すると、クライアント・アプリケーションから、Web サービス・メソッドをローカル機能のように呼び出すことができます。

クライアント・アプリケーションの作成を開始する前に、プロキシー・クライアントと必要なヘルパー・クラスを生成します。

Web サービス対応の開発ツール (IBM Rational Application Developer (RAD) など) を使用すれば、クライアント・アプリケーションを開発できます。どのタイプの Web サービス・アプリケーションをビルドしても、Web サービス API を呼び出すことができます。

1. 新規クライアント・アプリケーション・プロジェクトを作成します。
2. プロキシー・クライアントを生成し 、Java ヘルパー・クラスをプロジェクトに追加します。
3. クライアント・アプリケーションをコーディングします。
4. プロジェクトをビルドします。
5. クライアント・アプリケーションを実行します。

次の例では、Business Flow Manager Web サービス API を使用する方法を示します。

```
// create the service locator and the proxy
       BFMWSServiceLocator  locator = new BFMWSServiceLocator();
       BFMIF proxy = locator.getBFMWSPort();

 // prepare the input data for the operation
      GetProcessTemplate iW = new GetProcessTemplate();
      iW.setIdentifier(your_process_template_name);

 // invoke the operation
      GetProcessTemplateResponse oW = proxy.getProcessTemplate(iW);

 // process output of the operation
      ProcessTemplateType ptd = oW.getProcessTemplate();
      System.out.println("getName= " + ptd.getName());
      System.out.println("getPtid= " + ptd.getPtid());
```

関連タスク

300 ページの『プロキシー・クライアントの生成 (Java Web サービス)』
Java Web サービス・クライアント・アプリケーションは、*プロキシー・クライアント* を使用して Web サービス API と対話します。

302 ページの『BPEL プロセス用ヘルパー・クラスの作成 (Java Web サービス)』
具象 API 要求 (sendMessage 、call など) で参照されるビジネス・オブジェクトでは、クライアント・アプリケーションが「document/literal wrapped」 スタイル・エレメントを使用する必要があります。クライアント・アプリケーションは、 ヘルパー・クラスに、必要なラッパー・エレメントの生成を担当させます。

セキュリティーの追加 (Java Web サービス):

クライアント・アプリケーションにセキュリティー・メカニズムを実装することにより、 Web サービス通信を保護する必要があります。

WebSphere Application Server は現在、Web サービス API の次のセキュリティー・メカニズムをサポートしています。

- ユーザー名トークン
- Lightweight Third Party Authentication (LTPA)

関連概念

188 ページの『ヒューマン・タスクのための許可のロール』
ヒューマン・タスクで実行できるアクションは、ご使用の許可のロールに依存します。このロールは、J2EE のロールまたはインスタンス・ベースのロールとすることができます。

178 ページの『ビジネス・プロセスのための許可のロール』
ビジネス・プロセスで実行できるアクションは、ご使用の許可のロールに依存します。このロールは、J2EE のロールまたはインスタンス・ベースのロールとすることができます。

関連情報


アセンブリーおよびデプロイメント時のアプリケーションの保護

ユーザー名トークンのインプリメント:

ユーザー名トークンのセキュリティー・メカニズムは、ユーザー名とパスワードの信任状を提供します。

ユーザー名トークンのセキュリティー・メカニズムで、さまざまなコールバック・ハンドラーをインプリメントすることが選択できます。次の選択方法があります。

- クライアント・アプリケーションを実行するたびに、ユーザー名とパスワードを入力するようプロンプトが出されます。
- ユーザー名とパスワードは、デプロイメント記述子に書き込まれます。

いずれの場合も、入力したユーザー名とパスワードは、対応するビジネス・プロセス・コンテナーまたはヒューマン・タスク・コンテナーの権限があるロールのユーザー名とパスワードと一致する必要があります。

ユーザー名とパスワードは、要求メッセージ・エンベロープにカプセル化されるので、SOAP メッセージ・ヘッダーには「明確に」表示されます。したがって、HTTPS (HTTP over SSL) 通信プロトコルを使用できるようクライアント・アプリケーションを構成することをお勧めします。そうすると、すべての通信が暗号化されます。Web サービス API のエンドポイント URL アドレスを指定するときに、HTTPS 通信プロトコルを選択することができます。

ユーザー名トークンを定義するには、次のようにします。

1. WebSphere Integration Developer 内で使用可能な Rational Application Developer 環境で、「**WS バインディング (WS Binding)**」 → 「**セキュリティー要求ジェネレーターのバインディング構成 (Security Request Generator Binding Configuration)**」 → 「**トークン・ジェネレーター (Token Generator)**」を選択します。
2. 「トークン・ジェネレーター (Token Generator)」ダイアログで、**Username** を**トークン・タイプ (Token type)**として選択します。
3. 「**コールバック・ハンドラー (Call back handler)**」フィールドで、**com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler** (クライアント・アプリケーションを実行すると、ユーザー名およびパスワードを入力するようプロンプトが表示される)、または **com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler** と入力します。

4. **NonPromptCallbackHandler** を選択すると、有効なユーザー名とパスワードをデプロイメント記述子に指定する必要があります。

関連タスク

292 ページの『Web サービス・エンドポイント・アドレスの指定』
Web サービス・エンドポイント・アドレスは、クライアント・アプリケーションが Web サービス API にアクセスするために 指定する必要のある URL です。エンドポイント・アドレスは、クライアント・アプリケーション用の プロキシー・クライアントを生成するためにエクスポートする WSDL ファイルに書き込まれます。

関連情報


IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6

***LTPA* セキュリティー・メカニズムのインプリメント:**

Lightweight Third Party Authentication (LTPA) セキュリティー・メカニズムは、クライアント・アプリケーションが前に設定済みのセキュリティー・コンテキスト内で実行しているとき使用することができます。

LTPA セキュリティー・メカニズムは、セキュリティー・コンテキストがすでに確立されている安全な環境でクライアント・アプリケーションが実行している場合にのみ使用可能です。例えば、クライアント・アプリケーションが Enterprise JavaBeans (EJB) コンテナーで実行されている場合、EJB クライアントはクライアント・アプリケーションを呼び出す前にログインする必要があります。次に、セキュリティー・コンテキストが確立されます。次に、EJB クライアントが Web サービスを呼び出すと、LTPA コールバック・ハンドラーが LTPA トークンをセキュリティー・コンテキストから取得し、それを SOAP 要求メッセージに追加します。サーバー・サイドでは、LTPA トークンが LTPA メカニズムによって処理されます。

LTPA セキュリティー・メカニズムをインプリメントするには、以下を実行します。

1. WebSphere Integration Developer 内で使用可能な Rational Application Developer 環境で、「**WS バインディング (WS Binding)**」 → 「**セキュリティー要求ジェネレーターのバインディング構成 (Security Request Generator Binding Configuration)**」 → 「**トークン・ジェネレーター (Token Generator)**」を選択します。
2. 「トークン・ジェネレーター (Token Generator)」ダイアログで、**LTPAToken** を **トークン・タイプ (Token type)** として選択します。
3. 「**コールバック・ハンドラー (Call back handler)**」フィールドでは、**com.ibm.wsspi.wssecurity.auth.callback.LTPATokenCallbackHandler** と入力します。

実行時に、**LTPATokenCallbackHandler** は、LTPA トークンを既存のセキュリティー・コンテキストから取得し、それを SOAP 要求メッセージに追加します。

トランザクション・サポートの追加 (Java Web サービス):

Java Web サービス・クライアント・アプリケーションは、クライアント・アプリケーション・コンテキストをサービス要求の一部として渡すことによって、サーバー・サイドの要求処理がクライアントのトランザクションに参加するように構成することができます。このアトミック・トランザクション・サポートは、Web Services-Atomic Transaction (WS-AT) 仕様で定義されています。

WebSphere Application Server は、個々の Web サービス API 要求を別々のアトミック・トランザクションとして実行します。クライアント・アプリケーションは、以下のいずれかの方法でトランザクション・サポートを使用するよう構成できます。

- トランザクションに参加する。サーバー・サイドの要求処理は、クライアント・アプリケーション・トランザクション・コンテキスト内で実行されます。この場合、Web サービス API 要求の実行中やロールバック中にサーバーに問題が発生すると、クライアント・アプリケーションの要求もロールバックされます。
- トランザクション・サポートを使用しない。WebSphere Application Server はやはり新しいトランザクションを作成して、その中で要求を実行しますが、サーバー・サイドの要求処理はクライアント・アプリケーション・トランザクション・コンテキストで実行されません。

関連情報

WebSphere Application Server の Web サービス・アトミック・トランザクション・サポート

.NET 環境でのクライアント・アプリケーションの開発

Microsoft .NET は、Web サービスを使用してアプリケーションを接続する強力な開発環境を提供します。

プロキシー・クライアントの生成 (.NET):

.NET クライアント・アプリケーションは、プロキシー・クライアント を使用して Web サービス API と対話します。プロキシー・クライアントのおかげで、クライアント・アプリケーションは、複雑な Web サービス・メッセージング・プロトコルを意識する必要がなくなります。

プロキシー・クライアントを作成するには、まず WebSphere 環境からいくつかの WSDL ファイルをエクスポートして、それをクライアントのプログラミング環境にコピーする必要があります。

注: WebSphere Process Server クライアント CD を所有している場合は、代わりにその CD からファイルをコピーすることもできます。

プロキシー・クライアントは、C# Bean クラスのセットで構成されています。各クラスには、単一の Web サービスで公開されるメソッドおよびオブジェクトがすべて含まれています。サービス・メソッドは、パラメーターを完全な SOAP メッセージにアセンブルし、その SOAP メッセージを HTTP 経由で Web サービスに送信し、Web サービスから応答を受信して、戻されたデータを処理します。

注: プロキシー・クライアントの生成が必要なのは一度だけです。 Web サービス API にアクセスするクライアント・アプリケーションはすべて、以後は同じプロキシー・クライアントを使用できます。

1. プロキシー・クライアントの生成には WSDL コマンドを使用します。 タイプ:

   wsdl *options WSDLfilepath*

   各部の意味は、次のとおりです。

   - オプション は以下のとおりです。

     /language
     プロキシー・クラスの作成に使用する言語を指定できます。デフォルトは C# です。言語引数として、**VB** (Visual Basic)、**JS** (JScript)、または **VJS** (Visual J#) を指定することもできます。

     /output
     該当するサフィックスの付いた出力ファイルの名前。例えば、proxy.cs です。

     /protocol
     プロキシー・クラスでインプリメントされるプロトコル。**SOAP** がデフォルトの設定です。

     WSDL.exe パラメーターの完全なリストが必要な場合は、**/?** コマンド行スイッチを使用するか、Visual Studio で WSDL ツールのオンライン・ヘルプを参照してください。

   - *WSDLfilepath* は、WebSphere 環境からエクスポートしたか、クライアント CD からコピーした WSDL ファイルのパスおよびファイル名です。

   次の例では、ヒューマン・タスク・マネージャー Web サービス API 用のプロキシー・クライアントが生成されます。

   ```
   wsdl /language:cs /output:proxyclient.cs c:¥ws¥bin¥HTMWS.wsdl
   ```

2. プロキシー・クライアントをダイナミック・リンク・ライブラリー (DLL) ファイルとしてコンパイルします。

BPEL プロセス用ヘルパー・クラスの作成 (.NET):

特定の Web サービス API 操作では、クライアント・アプリケーションが「document/literal」スタイルのラップ・エレメントを使用する必要があります。クライアント・アプリケーションは、ヘルパー・クラスに、必要なラッパー・エレメントの生成を担当させます。

ヘルパー・クラスを作成するには、Web サービス API の WSDL ファイルを、WebSphere Process Server 環境からエクスポートしておく必要があります。

Web サービス API の call() 命令と sendMessage() 命令により、BPEL プロセスが WebSphere Process Server 内で起動します。call() 命令の入力メッセージは、BPEL プロセスの入力メッセージの document/literal ラッパーが提供されると予想します。BPEL プロセスに必要な Bean およびクラスを生成するには、`<wsdl:types>` エレメントを新規 XSD ファイルにコピーしてから、**xsd.exe** ツールを使用してヘルパー・クラスを生成します。

1. BPEL プロセス・インターフェースの WSDL ファイルをまだ WebSphere Integration Developer からエクスポートしていない場合は、ここでエクスポートします。

2. テキスト・エディターまたは XML エディターで WSDL ファイルを開きます。
3. <wsdl:types> エレメントのすべての子エレメントの内容をコピーし、新しいスケルトン XSD ファイルに貼り付けます。
4. XSD ファイル上で xsd.exe ツールを実行します。

   call xsd.exe file.xsd /classes /o

   各部の意味は、次のとおりです。

   file.xsd 変換する XML スキーマ定義ファイル。

   /classes (/c)
   指定した XSD ファイル (複数も可) の内容に対応するヘルパー・クラスを生成します。

   /output (/o)
   生成したファイルの出力ディレクトリーを指定します。このディレクトリーを省略した場合、デフォルトは現行ディレクトリーです。

   以下に例を示します。

   call xsd.exe ProcessCustomer.xsd /classes /output:c:¥temp
5. 生成されるクラス・ファイルをクライアント・アプリケーションに追加します。例えば Visual Studio を使用する場合は、「**プロジェクト**」 → 「**既存項目の追加 (Add Existing Item)**」メニュー・オプションを実行します。

ProcessCustomer.wsdl ファイルの内容が以下のような場合:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:bons1="http://com/ibm/bpe/unittest/sca"
                  xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  name="ProcessCustomer"
                  targetNamespace="http://ProcessTypes/bpel/ProcessCustomer">
  <wsdl:types>
    <xsd:schema targetNamespace="http://ProcessTypes/bpel/ProcessCustomer"
                xmlns:bons1="http://com/ibm/bpe/unittest/sca"
                xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:import namespace="http://com/ibm/bpe/unittest/sca"
                schemaLocation="xsd-includes/http.com.ibm.bpe.unittest.sca.xsd"/>
      <xsd:element name="doit">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="input1" nillable="true" type="bons1:Customer"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="doitResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="output1" nillable="true" type="bons1:Customer"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
    <wsdl:message name="doitRequestMsg">
    <wsdl:part element="tns:doit" name="doitParameters"/>
  </wsdl:message>
```

```
    <wsdl:message name="doitResponseMsg">
    <wsdl:part element="tns:doitResponse" name="doitResult"/>
  </wsdl:message>
    <wsdl:portType name="ProcessCustomer">
    <wsdl:operation name="doit">
      <wsdl:input message="tns:doitRequestMsg" name="doitRequest"/>
      <wsdl:output message="tns:doitResponseMsg" name="doitResponse"/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

結果として出力される XSD ファイルは以下のようになります。

```
<xsd:schema xmlns:bons1="http://com/ibm/bpe/unittest/sca"
                      xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
                      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                      targetNamespace="http://ProcessTypes/bpel/ProcessCustomer">
<xsd:import namespace="http://com/ibm/bpe/unittest/sca"
              schemaLocation="Customer.xsd"/>
<xsd:element name="doit">
<xsd:complexType>
<xsd:sequence>
    <xsd:element name="input1" type="bons1:Customer" nillable="true"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="doitResponse">
<xsd:complexType>
<xsd:sequence>
    <xsd:element name="output1" type="bons1:Customer" nillable="true"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

関連情報

XML スキーマ定義ツール (XSD.EXE) の Microsoft 文書

クライアント・アプリケーションの作成 (.NET):

クライアント・アプリケーションは、Web サービス API に要求を送信し、Web サービス API からの応答を受信します。プロキシー・クライアントを使用して、複合データ・タイプのフォーマット設定を行う通信クラスおよびヘルパー・クラスを管理すると、クライアント・アプリケーションから、Web サービス・メソッドをローカル機能のように呼び出すことができます。

クライアント・アプリケーションの作成を開始する前に、プロキシー・クライアントと必要なヘルパー・クラスを生成します。

.NET 対応の開発ツール (Visual Studio .NET など) を使用すれば、.NET クライアント・アプリケーションを開発できます。どのタイプの .NET アプリケーションをビルドしても、汎用の Web サービス API を呼び出すことができます。

1. 新規クライアント・アプリケーション・プロジェクトを作成します。 例えば、Visual Studio で **WinFX Windows Application** を作成します。
2. プロジェクト・オプションで、プロキシー・クライアントのダイナミック・リンク・ライブラリー (DLL) ファイルに参照を追加します。 ビジネス・オブジェク

ト定義を含むすべてのヘルパー・クラスをプロジェクトに追加します。例えば Visual Studio では、**「プロジェクト」 → 「既存項目の追加 (Add existing item)」**オプションを実行します。

3. プロキシー・クライアント・オブジェクトを作成します。 以下に例を挙げます。

```
HTMClient.HTMReference.HumanTaskManagerComponent1Export_HumanTaskManagerHttpService service =
    new HTMClient.HTMReference.HumanTaskManagerComponent1Export_HumanTaskManagerHttpService();
```

4. メッセージで使用され、Web サービスとの間で送受信されるビジネス・オブジェクト・データ・タイプを宣言します。 以下に例を挙げます。

```
HTMClient.HTMReference.TKIID id = new HTMClient.HTMReference.TKIID();

ClipBG bg = new ClipBG();
Clip clip = new Clip();
```

5. 特定の Web サービス関数を呼び出し、必要なパラメーターを指定します。 例えば、ヒューマン・タスクを作成して開始するには、以下のようにします。

```
HTMClient.HTMReference.createAndStartTask task =
   new HTMClient.HTMReference.createAndStartTask();
HTMClient.HTMReference.StartTask sTask = new HTMClient.HTMReference.StartTask();

sTask.taskName = "SimpleTask";
sTask.taskNamespace = "http://myProcess/com/acme/task";
sTask.inputMessage = bg;
task.inputTask = sTask;

id = service.createAndStartTask(task).outputTask;
```

6. リモートのプロセスおよびタスクは、永続 ID (上記の例では `id`) で識別されます。 例えば、上記で作成したヒューマン・タスクを要求するには、以下のようにします。

```
HTMClient.HTMReference.claimTask claim = new HTMClient.HTMReference.claimTask();
claim.inputTask = id;
```

関連タスク

307 ページの『プロキシー・クライアントの生成 (.NET)』
.NET クライアント・アプリケーションは、*プロキシー・クライアント* を使用して Web サービス API と対話します。プロキシー・クライアントのおかげで、クライアント・アプリケーションは、 複雑な Web サービス・メッセージング・プロトコルを意識する必要がなくなります。

308 ページの『BPEL プロセス用ヘルパー・クラスの作成 (.NET)』
特定の Web サービス API 操作では、クライアント・アプリケーションが「document/literal」スタイルのラップ・エレメントを使用する必要があります。クライアント・アプリケーションは、 ヘルパー・クラスに、必要なラッパー・エレメントの生成を担当させます。

セキュリティーの追加 (.NET):

Web サービス通信は、クライアント・アプリケーションにセキュリティー・メカニズムを組み込むことにより保護できます。

このセキュリティー・メカニズムには、ユーザー名トークン (ユーザー名とパスワード)、またはカスタム・バイナリーおよび XML ベースのセキュリティー・トークンが使用できます。

1. Web Services Enhancements (WSE) 2.0 SP3 for Microsoft .NET をダウンロードしてインストールします。 入手先は以下のとおりです。

   http://www.microsoft.com/downloads/details.aspx?familyid=1ba1f631-c3e7-420a-bc1e-ef18bab66122&displaylang=en
2. 生成されたプロキシー・クライアント・コードを以下のように変更します。

   変更前:

```
public class Export1_MyMicroflowHttpService : System.Web.Services.Protocols.SoapHttpClientProtocol {
```

   変更後:

```
public class Export1_MyMicroflowHttpService : Microsoft.Web.Services2.WebServicesClientProtocol {
```

   注: この変更内容は、WSDL.exe ツールを実行してプロキシー・クライアントを再生成すると失われます。
3. ファイルの先頭に以下の行を追加して、クライアント・アプリケーション・コードを変更します。

```
using System.Web.Services.Protocols;
using Microsoft.Web.Services2;
using Microsoft.Web.Services2.Security.Tokens;
...
```
4. 必要なセキュリティー・メカニズムをインプリメントするコードを追加します。例えば、ユーザー名とパスワード保護を追加するコードは次のとおりです。

```
string user = "U1";
string pwd = "password";
UsernameToken token =
   new UsernameToken(user, pwd, PasswordOption.SendPlainText);

me._proxy.RequestSoapContext.Security.Tokens.Clear();
me._proxy.RequestSoapContext.Security.Tokens.Add(token);
```

ビジネス・プロセスおよびタスク関連のオブジェクトの照会

Web サービス API を使用すると、Business Process Choreographer データベース内のビジネス・プロセス・オブジェクトおよびタスク関連オブジェクトを照会して、これらのオブジェクトの特定のプロパティーを取得することができます。

Business Process Choreographer データベースは、ビジネス・プロセスおよびタスクの管理用のテンプレート (モデル) とインスタンス (ランタイム) のデータを保管します。

クライアント・アプリケーションは、Web サービス API 経由で照会を発行し、データベースからビジネス・プロセスおよびビジネス・タスクに関する情報を取得することができます。

クライアント・アプリケーションは、1 回限りの照会を発行して、オブジェクトの特定のプロパティーを取得することができます。使用頻度の高い照会は、保管しておくことができます。クライアント・アプリケーションは、このような保管照会文を後で取り出して使用することができます。

ビジネス・プロセスおよびタスク関連オブジェクトに対する照会:

Web サービス API の QUERY インターフェースを使用して、ビジネス・プロセスおよびタスクに関する情報を取得します。

クライアント・アプリケーションは、SQL 形式の構文を使用して、データベースを照会します。

Java Web サービスの例

```
string processTemplateName = "ProcessCustomerLR";
query query1 = new query();
query1.selectClause = "DISTINCT PROCESS_INSTANCE.STARTED, PROCESS_INSTANCE.PIID";
query1.whereClause =
       "PROCESS_INSTANCE.TEMPLATE_NAME = '" + processTemplateName + "'";
query1.orderByClause = "PROCESS_INSTANCE.STARTED";
query1.threshold = null;
query1.timeZone = "UTC"; query1.skipTuples = null;
queryResponse queryResponse1 = proxy.query(query1);
```

データベースから取り出した情報は、Web サービス API を使用して*照会結果セット*として戻されます。

以下に例を挙げます。

```
QueryResultSetType queryResultSet = queryResponse1.queryResultSet;
if (queryResultSet != null) {
   Console.WriteLine("--> QueryResultSetType");
   Console.WriteLine(" . size= " + queryResultSet.size);
   Console.WriteLine(" . numberColumns= " + queryResultSet.numberColumns);
   string indent = " . ";

   // -- the query column info
   QueryColumnInfoType[] queryColumnInfo = queryResultSet.QueryColumnInfo;
   if (queryColumnInfo.Length > 0) {
      Console.WriteLine();
      Console.WriteLine("= . QueryColumnInfoType size= " + queryColumnInfo.Length);
      Console.Write( " | tableName ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         Console.Write( " | " + queryColumnInfo[i].tableName.PadLeft(20) );
      }
      Console.WriteLine();
      Console.Write( " | columnName ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         Console.Write( " | " + queryColumnInfo[i].columnName.PadLeft(20) );
      }
      Console.WriteLine();
      Console.Write( " | data type ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         QueryColumnInfoTypeType tt = queryColumnInfo[i].type;
         Console.WriteLine( " | " + tt.ToString());
      }
         Console.WriteLine();
   }
   else {
      Console.WriteLine("--> queryColumnInfo= <null>");
   }

   // - the query result values
   string[][] result = queryResultSet.result;
   if (result !=null) {
      Console.WriteLine();
      Console.WriteLine("= . result size= " + result.Length);
      for (int i = 0; i &lt; result.Length; i++) {
        Console.Write(indent +i );
        string[] row = result[i];
        for (int j = 0; j &lt; row.Length; j++ ) {
```

```
            Console.Write(" | " + row[j]);
          }
          Console.WriteLine();
        }
    }
    else {
       Console.WriteLine("--> result= <null>");
    }
}
else {
    Console.WriteLine("--> QueryResultSetType= <null>");
}
```

照会関数は、呼び出し元の権限に応じてオブジェクトを戻します。照会結果セットには、呼び出し元が表示を許可されているオブジェクトのプロパティーのみが含まれます。

オブジェクトのプロパティーを照会するために、事前定義データベース・ビューが提供されています。プロセス・テンプレートの場合、照会関数には以下の構文があります。

```
ProcessTemplateData[] queryProcessTemplates
                       (java.lang.String whereClause,
                        java.lang.String orderByClause,
                        java.lang.Integer threshold,
                        java.util.TimeZone timezone);
```

タスク・テンプレートの場合、照会関数には以下の構文があります。

```
TaskTemplate[] queryTaskTemplates
                (java.lang.String whereClause,
                 java.lang.String orderByClause,
                 java.lang.Integer threshold,
                 java.util.TimeZone timezone);
```

他のビジネス・プロセスおよびタスク関連オブジェクトの場合、照会関数には以下の構文があります。

```
QueryResultSet query (java.lang.String selectClause,
                      java.lang.String whereClause,
                      java.lang.String orderByClause,
                      java.lang.Integer skipTuples
                      java.lang.Integer threshold,
                      java.util.TimeZone timezone);
```

QUERY インターフェースには、queryAll メソッドも含まれています。このメソッドを使用して、オブジェクトに関係のあるデータすべてを、モニターなどの目的で取得することができます。queryAll メソッドの呼び出し元には、 Java 2 Platform Enterprise Edition (J2EE) ロールの、BPESystemAdministrator、BPESystemMonitor、TaskSystemAdministrator、または TaskSystemMonitor のいずれかが必要です。オブジェクトの対応する作業項目を使用した許可検査は適用されません。

.NET の例

```
ProcessTemplateType[] templates = null;

   try {
        queryProcessTemplates iW = new queryProcessTemplates();
        iW.whereClause = "PROCESS_TEMPLATE.STATE=PROCESS_TEMPLATE.STATE.STATE_STARTED";
        iW.orderByClause = null;
```

```
  iW.threshold = null;
  iW.timeZone = null;

  Console.WriteLine("--> queryProcessTemplates ... ");
  Console.WriteLine("--> query: WHERE " + iW.whereClause + " ORDER BY " +
    iW.orderByClause + " THRESHOLD " + iW.threshold + " TIMEZONE" + iW.timeZone);

  templates = proxy.queryProcessTemplates(iW);

  if (templates.Length < 1) {
    Console.WriteLine("--> No templates found :-(");
  }
  else {
    for (int i = 0; i < templates.Length ; i++) {
      Console.Write("--> found template with ptid: " + templates[i].ptid);
      Console.WriteLine(" and name: " + templates[i].name);
     /*  ... other properties of ProcessTemplateType ... */
    }
  }
}
catch( Exception e ) {
  Console.WriteLine("exception= " + e);
}
```

照会パラメーター:

各照会では、複数の SQL 形式の文節およびパラメーターを指定する必要があります。

照会は以下のもので構成されます。

- select 文節
- where 文節
- order-by 文節
- スキップ・タプル・パラメーター
- しきい値パラメーター
- 時間帯パラメーター

ビジネス・プロセス・オブジェクトおよびヒューマン・タスク・オブジェクトの照会のための事前定義ビュー:

ビジネス・プロセス・オブジェクトおよびヒューマン・タスク・オブジェクト用に、事前定義データベース・ビューが提供されています。

これらのオブジェクトの参照データを照会する場合は、これらのビューを使用します。これらのビューを使用する場合は、ビューの列に明示的に述部を結合する必要はありません。これらの構成要素は自動的に追加されます。Web サービス API の照会機能を使用して、このデータを照会できます。

保管照会文の管理:

保管照会文は、頻繁に実行される照会を保管する方法を提供します。この保管照会文は、すべてのユーザーに対して使用できる照会 (共通照会) にすることも、特定のユーザーに属する照会 (専用照会) にすることもできます。

保管照会文は、データベースに保管され、名前で識別される照会のことです。専用の保管照会文と共通の保管照会文の名前を同じにすることができます。異なる複数の所有者の専用保管照会文を同じ名前にすることもできます。

保管照会文は、ビジネス・プロセス・オブジェクト、タスク・オブジェクト、またはこの 2 つのオブジェクト・タイプの組み合わせたものを対象とします。

共通保管照会文の管理
共通保管照会文はシステム管理者によって作成されます。 この照会は、 全ユーザーが使用できます。

他のユーザーの専用保管照会文の管理
専用照会はどのユーザーでも作成できます。この照会は、 照会の所有者とシステム管理者しか使用できません。

専用保管照会文の操作
システム管理者でなくても、自分専用の保管照会文は作成、実行、 および削除できます。また、システム管理者が作成した共通の保管照会文を 使用することもできます。

JSF コンポーネントを使用した、ビジネス・プロセスおよびヒューマン・タスク用 Web アプリケーションの開発

Business Process Choreographer Explorer は、いくつかの JavaServer Faces (JSF) コンポーネントを提供します。これらのコンポーネントを拡張および統合して、ビジネス・プロセスおよびヒューマン・タスク機能を Web アプリケーションに追加することができます。

WebSphere Integration Developer を使用して Web アプリケーションを作成することができます。

1. 動的プロジェクトを作成し、Web プロジェクトの Web プロジェクト・フィーチャー・プロパティーを変更して JSF 基本コンポーネントを組み込みます。

   Web プロジェクトの作成の詳細については、WebSphere Integration Developer インフォメーション・センターにアクセスしてください。
2. 前提条件である Business Process Choreographer Explorer Java アーカイブ (JAR ファイル) を追加します。

   以下のファイルをプロジェクトの WEB-INF/lib ディレクトリーに追加してください。

   - bpcclientcore.jar
   - bfmclientmodel.jar
   - htmclientmodel.jar
   - bpcjsfcomponents.jar

   これらのファイルは *install_root*/ProcessChoreographer/client ディレクトリーにあります。
3. 必要な EJB 参照を、Web アプリケーション・デプロイメント記述子 web.xml ファイルに追加します。

```
<ejb-ref id="EjbRef_1">
  <ejb-ref-name>ejb/BusinessProcessHome</ejb-ref-name>
   <ejb-ref-type>Session</ejb-ref-type>
   <home>com.ibm.bpe.api.BusinessFlowManagerHome</home>
   <remote>com.ibm.bpe.api.BusinessFlowManager</remote>
</ejb-ref>
<ejb-ref id="EjbRef_2">
```

```
    <ejb-ref-name>ejb/HumanTaskManagerEJB</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
     <home>com.ibm.task.api.HumanTaskManagerHome</home>
     <remote>com.ibm.task.api.HumanTaskManager</remote>
  </ejb-ref>
  <ejb-local-ref id="EjbLocalRef_1">
    <ejb-ref-name>ejb/LocalBusinessProcessHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
     <local-home>com.ibm.bpe.api.LocalBusinessFlowManagerHome</local-home>
     <local>com.ibm.bpe.api.LocalBusinessFlowManager</local>
  </ejb-local-ref>
  <ejb-local-ref id="EjbLocalRef_2">
    <ejb-ref-name>ejb/LocalHumanTaskManagerEJB</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
     <local-home>com.ibm.task.api.LocalHumanTaskManagerHome</local-home>
     <local>com.ibm.task.api.LocalHumanTaskManager</local>
  </ejb-local-ref>
```

4. Business Process Choreographer Explorer JSF コンポーネントを JSF アプリケーションに追加します。
   a. アプリケーションに必要なタグ・ライブラリーを JavaServer Pages (JSP) ファイルに追加します。 通常、JSF および HTML タグ・ライブラリーと、JSF コンポーネントに必要とされるタグ・ライブラリーが必要です。
      - `<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>`
      - `<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>`
      - `<%@ taglib uri="http://com.ibm.bpe.jsf/taglib" prefix="bpe" %>`
   b. JSP ページの本体に `<f:view>` タグを追加し、`<f:view>` タグに `<h:form>` タグを追加します。
   c. JSP ファイルに JSF コンポーネントを追加します。

      アプリケーションに応じて、List コンポーネント、Details コンポーネント、CommandBar コンポーネント、または Message コンポーネントを JSP ファイル内に追加します。各コンポーネントの複数のインスタンスを追加することができます。
   d. JSF 構成ファイル内で管理対象 Bean を構成します。

      デフォルトでは、構成ファイルは faces-config.xml ファイルです。このファイルは、Web アプリケーションの WEB-INF ディレクトリーにあります。 JSP ファイルに追加するコンポーネントに応じて、照会およびその他のラッパー・オブジェクトへの参照を JSF 構成ファイルに追加する必要もあります。
   e. JSF コンポーネントをサポートするために必要なカスタム・コードをインプリメントします。
5. アプリケーションをデプロイします。

   EJB 参照を Java Naming and Directory Interface (JNDI) 名へマップするか、参照を ibm-web-bnd.xmi ファイルへ手動で追加します。

   以下の表に、参照バインディングおよびそのデフォルト・マッピングを示します。

表 30. 参照バインディングから JNDI 名へのマッピング

参照バインディング	JNDI 名	コメント
ejb/BusinessProcessHome	com/ibm/bpe/api/BusinessFlowManagerHome	リモート・セッション Bean

表 30. 参照バインディングから JNDI 名へのマッピング (続き)

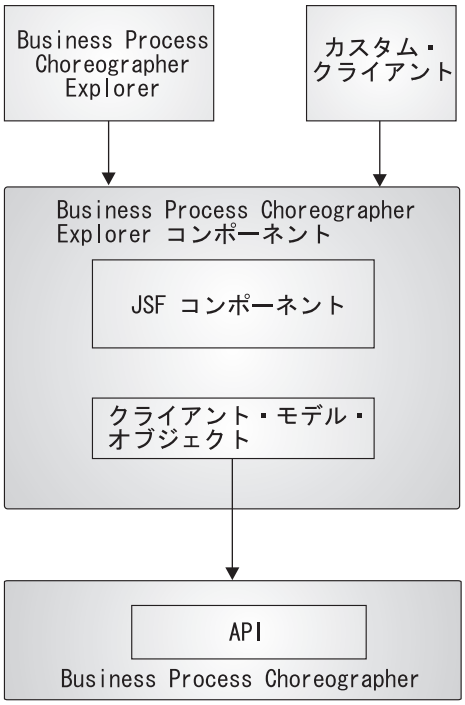
参照バインディング	JNDI 名	コメント
ejb/LocalBusinessProcessHome	com/ibm/bpe/api/BusinessFlowManagerHome	ローカル・セッション Bean
ejb/HumanTaskManagerEJB	com/ibm/task/api/HumanTaskManagerHome	リモート・セッション Bean
ejb/LocalHumanTaskManagerEJB	com/ibm/task/api/HumanTaskManagerHome	ローカル・セッション Bean

デプロイした Web アプリケーションには、Business Process Choreographer Explorer コンポーネントが提供する機能が含まれています。

Business Process Choreographer Explorer コンポーネント

Business Process Choreographer Explorer コンポーネントは、JavaServer Faces (JSF) テクノロジーに基づく構成可能かつ再使用可能なエレメントの集合です。これらのエレメントを Web アプリケーションに組み込むことができます。これにより、Web アプリケーションは、インストール済みビジネス・プロセスおよびヒューマン・タスク・アプリケーションにアクセスできるようになります。

コンポーネントは、JSF コンポーネントのセットおよびクライアント・モデル・オブジェクトのセットで構成されています。コンポーネントから Business Process Choreographer、Business Process Choreographer Explorer、およびその他のカスタム・クライアントへの関係を、次の図に示します。



JSF コンポーネント

Business Process Choreographer Explorer コンポーネントには、以下の JSF コンポーネントが含まれます。ビジネス・プロセスおよびヒューマン・タスクを操作するための Web アプリケーションをビルドするときに、これらの JSF コンポーネントを JavaServer Pages (JSP) ファイルに組み込みます。

- List コンポーネント

  List コンポーネントは、例えば、タスク、アクティビティー、プロセス・インスタンス、プロセス・テンプレート、作業項目、またはエスカレーションなどの、アプリケーション・オブジェクトのリストをテーブル内に表示します。このコンポーネントには、関連付けられたリスト・ハンドラーがあります。
- Details コンポーネント

  Details コンポーネントは、タスク、作業項目、アクティビティー、プロセス・インスタンス、およびプロセス・テンプレートのプロパティーを表示します。このコンポーネントには、関連付けられた詳細ハンドラーがあります。
- CommandBar コンポーネント

  CommandBar コンポーネントは、ボタンを含むバーを表示します。これらのボタンは、詳細ビュー内のオブジェクトまたはリスト内の選択されたオブジェクトのいずれかに作動するコマンドを表します。これらのオブジェクトは、リスト・ハンドラーまたは詳細ハンドラーによって提供されます。
- Message コンポーネント

  Message コンポーネントは、サービス・データ・オブジェクト (SDO) または単純型のいずれかを含むことのできるメッセージを表示します。

クライアント・モデル・オブジェクト

クライアント・モデル・オブジェクトは、JSF コンポーネントと共に使用されます。これらのオブジェクトは、基盤となる Business Process Choreographer API のインターフェースの一部をインプリメントし、元のオブジェクトをラップします。クライアント・モデル・オブジェクトは、ラベルの各国語サポートと、一部のプロパティーのコンバーターを提供します。

List コンポーネントでのリスト処理:

List コンポーネントのインスタンスはすべて com.ibm.bpe.jsf.handler.BPCListHandler クラスのインスタンスに関連しています。

このリスト・ハンドラーは、関連するリスト内で選択された項目をトラッキングし、通知メカニズムを提供します。リスト・ハンドラーは、`bpe:list` タグの **model** 属性を介して List コンポーネントにバインドされます。

リスト・ハンドラーの通知メカニズムは、 com.ibm.bpe.jsf.handler.ItemListener インターフェースを使用してインプリメントされます。 Business Process Choreographer Explorer はこの通知メカニズムを使用して、さまざまな種類の項目用の詳細ページにリスト項目を関連付けます。通知イベントのトリガーは、通常、現在のリストで表示される項目のプロパティーの 1 つです。

通知メカニズムを活用するには、プロパティーの `bpe:column` タグの **action** 属性の値を、通知イベントがトリガーされたときにアプリケーションが動作を続行する JavaServer Faces (JSF) ナビゲーション・ターゲットに設定します。List コンポーネントは、列内の項目を JSF コマンド・リンクとしてレンダリングします。リンクがトリガーされると、リスト内の項目を表すオブジェクトが判別され、すべての登録

済み項目リスナーに渡されます。このインターフェースのインプリメンテーションは、 JSF アプリケーションの構成ファイルに登録できます。

BPCListHandler クラスは、refreshList メソッドを提供します。このメソッドを JSF メソッド・バインディングで使用して、照会を再実行するためのユーザー・インターフェース制御をインプリメントすることができます。

照会のインプリメンテーション

リスト・ハンドラーを使用すると、すべての種類のオブジェクトおよびそれらのプロパティーを表示できます。表示されるリストの内容は、リスト・ハンドラー用に構成された com.ibm.bpc.clientcore.Query インターフェースのインプリメンテーションによって戻されるオブジェクトのリストによって異なります。照会は、BPCListHandler クラスの setQuery メソッドを使用してプログラマチックに設定することもできますし、アプリケーションの JSF 構成ファイルで構成することもできます。

照会は、Business Process Choreographer API に対してだけでなく、コンテンツ管理システムやデータベースなど、アプリケーションからアクセスできるその他の情報ソースに対しても実行できます。要件は、照会の結果が execute メソッドでオブジェクトの `java.util.List` として戻されることだけです。

戻されるオブジェクトのタイプは、照会が定義されたリストの列に表示されるすべてのプロパティーに対して適切な getter メソッドを使用できることを保証する必要があります。戻されるオブジェクトのタイプがリスト定義に適合することを確認するには、 faces 構成ファイルで定義されている BPCListHandler インスタンス上のタイプ・プロパティーの値を、戻されるオブジェクトの完全修飾クラス名に設定します。この名前は、照会インプリメンテーションの getType 呼び出しで戻すことができます。実行時に、リスト・ハンドラーはオブジェクト・タイプが定義に準拠していることを確認します。

リスト内の特定の項目にエラー・メッセージをマップするには、照会によって戻されたオブジェクトが署名 `public Object getID()` でメソッドをインプリメントする必要があります。

エラー処理

次のエラー状態では、BPCListHandler クラスが提供するエラー処理機能を利用できます。

- 照会の実行時またはコマンドの実行時に発生するエラー

  照会の実行中にエラーが発生した場合、 BPCListHandler クラスは、不十分なアクセス権限によるエラーとその他の例外とを区別します。不十分なアクセス権限によるエラーをキャッチするには、照会の execute メソッドによってスローされる ClientException の **rootCause** パラメーターが com.ibm.bpe.api.EngineNotAuthorizedException または com.ibm.task.api.NotAuthorizedException 例外である必要があります。 List コンポーネントは、照会の結果の代わりにエラー・メッセージを表示します。

  エラーが不十分なアクセス権限によるものでない場合、 BPCListHandler クラスは例外オブジェクトを、 JSF アプリケーション構成ファイルの BPCError キーで定

義した com.ibm.bpc.clientcore.util.ErrorBean インターフェースのインプリメンテーションに渡します。例外が設定されている場合は、エラー・ナビゲーション・ターゲットが呼び出されます。

- リストに表示される項目の処理時に発生するエラー

  BPCListHandler クラスは、 com.ibm.bpe.jsf.handler.ErrorHandler インターフェースをインプリメントします。 setErrors メソッドでタイプ java.util.Map のマップ・パラメーターを使用して、これらのエラーに関する情報を提供することができます。このマップには、キーとして ID が、値として例外が含まれています。 ID は、エラーの原因となったオブジェクトの getID メソッドによって戻された値である必要があります。マップが設定されていて、リスト内に表示されている項目のいずれかに ID のいずれかが一致する場合は、リスト・ハンドラーによって、エラー・メッセージを含む列が自動的にリストに追加されます。

  リスト内のエラー・メッセージが古くなるのを避けるため、エラー・マップをリセットしてください。次の状況では、マップは自動的にリセットされます。

  – refreshList メソッドの BPCListHandler クラスが呼び出される。
  – BPCListHandler クラスで新規照会が設定されている。
  – CommandBar コンポーネントを使用して、リストの項目でアクションがトリガーされている。 CommandBar コンポーネントは、エラー処理のメソッドの 1 つとしてこのメカニズムを使用します。

CommandBar コンポーネント:

CommandBar コンポーネントを使用して、アプリケーションにアクション・ボタンを追加します。コンポーネントは、ユーザー・インターフェースでのアクション用のボタンを作成し、ボタンがクリックされたときに作成されるイベントを処理します。

これらのボタンは、BPCListHandler クラスや BPCDetailsHandler クラスなど、com.ibm.bpe.jsf.handler.ItemProvider インターフェースによって戻されるオブジェクトで動作する機能を起動します。 CommandBar コンポーネントは、 `bpe:commandbar` タグで **model** 属性の値によって定義された項目プロバイダーを使用します。

コマンドの処理方法

アプリケーションのユーザー・インターフェースのコマンド・バー・セクションにあるボタンをクリックすると、関連するイベントが CommandBar コンポーネントによって次のように処理されます。

1. CommandBar コンポーネントは、イベントを生成したボタンに対して指定された com.ibm.bpc.clientcore.Command インターフェースのインプリメンテーションを示します。
2. CommandBar コンポーネントに関連するモデルが com.ibm.bpe.jsf.handler.ErrorHandler インターフェースをインプリメントすると、前のイベントからのエラー・メッセージを削除するため、clearErrorMap メソッドが呼び出されます。
3. ItemProvider インターフェースの getSelectedItems メソッドが呼び出されます。戻された項目のリストは、コマンドの execute メソッドに渡され、コマンドが呼び出されます。

4. CommandBar コンポーネントは、 JavaServer Faces (JSF) ナビゲーション・ターゲットを決定します。 bpe:commandbar タグで **action** 属性が指定されていない場合は、 execute メソッドの戻り値によってナビゲーション・ターゲットが指定されます。 **action** 属性が JSF メソッド・バインディングに設定されている場合は、メソッドによって戻されたストリングがナビゲーション・ターゲットと解釈されます。 **action** 属性は、明示的なナビゲーション・ターゲットも指定します。

エラー処理

bpe:commandbar タグにおいて **action** 属性によって指定された action メソッドは、次のいずれかの条件が満たされる場合に呼び出されます。

- 例外がスローされない。
- 例外がスローされる場合は、ErrorsInCommandException 例外である。

CommandBar コンポーネントでエラー処理をインプリメントする方法をいくつか示します。

- CommandBar コンポーネントの機能を使用しないことが可能です。例えば、選択したコマンドに固有のページにエラーを表示する場合は、コマンドのインプリメンテーションによって、発生した例外をキャッチし、エラー・ページに使用するページ Bean にそれを伝搬することができます。 bpe:commandbar タグの **context** 属性を使用して、ページ Bean をコマンド・インプリメンテーションに使用できるようにすることができます。ページ Bean で例外が設定された後、コマンドは、エラー・ページ用に定義された JSF ナビゲーション・ルールのストリングを戻します。
- ユーザー・インターフェースのコマンド・バー・セクションの下にエラー・メッセージを表示する場合は、com.ibm.bpc.clientcore.exception.CommandBarMessage マーク文字インターフェースをインプリメントする例外クラスを作成します。このインターフェースは、エラー・メッセージのメッセージ・カタログを提供します。
- コマンドが項目のリストで動作する場合は、リスト内の各項目についてコマンドの成功をトラッキングすることができます。エラーをトラッキングするには、操作が失敗した項目に各例外をマップします。 CommandBar コンポーネントは、キーとしての ID と値としての例外を含むマップを、 CommandBar コンポーネントに対して定義されたモデル・オブジェクトに渡すことができます。

  このメカニズムを機能させるには、モデル・オブジェクトが com.ibm.bpe.jsf.handler.ErrorHandler インターフェースをインプリメントし、コマンドが com.ibm.bpc.clientcore.exception.ErrorsInCommandException 例外をスローする必要があります。次に、CommandBar コンポーネントは、例外に含まれているマップをエラー・ハンドラーに渡します。エラーは発生しましたが、このアクション・メソッドがトリガーされ、現在のビューが更新されます。 Business Process Choreographer Explorer アプリケーションは、このメソッドを利用して、リストで例外を表示します。
- CommandBarMessage インターフェースをインプリメントしない ClientException 例外をスローし、例外が ErrorsInCommandException ではない場合、 CommandBar コンポーネントは、アプリケーションの構成ファイルで定義された BPCError エラー Bean に例外を伝搬します。エラー・ページでエラー処理が続けられます。

Business Process Choreographer Explorer JSF コンポーネントによって提供されるユーティリティー:

JavaServer Faces (JSF) コンポーネントは、ユーザー指定の時間帯情報およびエラー処理のためのユーティリティーを提供します。

ユーザー固有の時間帯情報

BPCListHandler クラスは、 com.ibm.bpc.clientcore.util.User インターフェースを使用して、各ユーザーの時間帯およびロケールに関する情報を取得します。 List コンポーネントは、JavaServer Faces (JSF) 構成ファイルでインターフェースのインプリメンテーションが **user** を管理対象 Bean 名として設定されていると予期します。構成ファイル内でこの項目が欠けている場合は、 WebSphere Process Server が動作している時間帯が戻されます。

com.ibm.bpc.clientcore.util.User インターフェースは次のように定義されています。

```
public interface User {

    /**
     * The locale used by the client of the user.
     * @return Locale.
     */
    public Locale getLocale();
   /**
    * The time zone used by the client of the user.
    * @return TimeZone.
    */
    public TimeZone getTimeZone();

   /**
    * The name of the user.
    * @return name of the user.
    */
    public String getName();
}
```

エラー処理用の ErrorBean インターフェース

JSF コンポーネントはエラー処理の際に、事前定義された管理対象 Bean である BPCError を活用します。 この Bean は、com.ibm.bpc.clientcore.util.ErrorBean インターフェースをインプリメントします。エラー・ページをトリガーするエラー状態では、例外がエラー Bean で設定されます。エラー・ページが表示されるのは、次のような状態のときです。

- リスト・ハンドラー用に定義された照会の実行中にエラーが発生し、エラーがコマンドの execute メソッドによって ClientException エラーとして生成された場合
- ClientException エラーがコマンドの execute メソッドによって生成され、このエラーが ErrorsInCommandException エラーではなく、 CommandBarMessage インターフェースのインプリメントもしない場合
- コンポーネント内でエラー・メッセージが表示され、メッセージのハイパーリンクに従っている場合

com.ibm.bpc.clientcore.util.ErrorBeanImpl インターフェースのデフォルト・インプリメンテーションを使用できます。

インターフェースは次のように定義されます。

```
public interface ErrorBean {

    public void setException(Exception ex);

    /*
     * This setter method call allows a locale and
     * the exception to be passed. This allows the
     * getExceptionMessage methods to return localized Strings
     *
     */
    public void setException(Exception ex, Locale locale);

    public Exception getException();
    public String getStack();
    public String getNestedExceptionMessage();
    public String getNestedExceptionStack();
    public String getRootExceptionMessage();
    public String getRootExceptionStack();

    /*
     * This method returns the exception message
     * concatenated recursively with the messages of all
     * the nested exceptions.
     */
    public String getAllExceptionMessages();

    /*
     * This method is returns the exception stack
     * concatenated recursively with the stacks of all
     * the nested exceptions.
     */
    public String getAllExceptionStacks();
}
```

JSF アプリケーションへの List コンポーネントの追加

Business Process Choreographer Explorer List コンポーネントを使用して、例えばビジネス・プロセス・インスタンスやタスク・インスタンスなどの、クライアント・モデル・オブジェクトのリストを表示します。

1. List コンポーネントを JavaServer Pages (JSP) ファイルに追加します。

   bpe:list タグを h:form タグに追加します。 bpe:list タグには、モデル属性が含まれていなければなりません。bpe:column タグを bpe:list に追加して、リスト内の各行に表示されるオブジェクトのプロパティーを追加します。

   以下の例では、List コンポーネントを追加してタスク・インスタンスを表示する方法を示します。

   ```
   <h:form>

      <bpe:list model="#{TaskPool}">
         <bpe:column name="name" action="taskInstanceDetails" />
         <bpe:column name="state" />
         <bpe:column name="kind" />
         <bpe:column name="owner" />
         <bpe:column name="originator" />
      </bpe:list>

   </h:form>
   ```

   モデル属性は、TaskPool という管理対象 Bean を参照します。管理対象 Bean は、リストが操作を繰り返す対象となる Java オブジェクトのリストを提供し、次に個々の行を表示します。

2. bpe:list タグで参照されている管理対象 Bean を構成します。

   List コンポーネントの場合、この管理対象 Bean は、com.ibm.bpe.jsf.handler.BPCListHandler クラスのインスタンスでなければなりません。

   以下の例では、TaskPool 管理対象 Bean を構成ファイルに追加する方法を示します。

```
<managed-bean>
<managed-bean-name>TaskPool</managed-bean-name>
<managed-bean-class>com.ibm.bpe.jsf.handler.BPCListHandler</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>

   <managed-property>
      <property-name>query</property-name>
      <value>#{TaskPoolQuery}</value>
   </managed-property>

   <managed-property>
      <property-name>type</property-name>
      <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
   </managed-property>

</managed-bean>
<managed-bean>
<managed-bean-name>htmConnection</managed-bean-name>
<managed-bean-class>com.ibm.task.clientmodel.HTMConnection</managed-bean-class>
<managed-bean-scope>application</managed-bean-scope>

   <managed-property>
      <property-name>jndiName</property-name>
      <value>java:comp/env/ejb/LocalHumanTaskManagerEJB</value>
   </managed-property>
</managed-bean>
```

   例では、照会およびタイプの 2 つの構成可能プロパティーが TaskPool に含まれていることを示しています。照会プロパティーの値は、TaskPoolQuery という別の管理対象 Bean を参照しています。タイプ・プロパティーの値は、Bean クラスを指定します。そのクラスのプロパティーは、表示されたリストの列に示されます。関連する照会インスタンスは、プロパティー型を持つことも可能です。プロパティー型が指定された場合、それはリスト・ハンドラーに指定された型と同一でなければなりません。

   TaskPool 管理対象 Bean は、Human Task Manager への接続を提供するため、htmConnection 管理対象 Bean を使用してインプリメントされます。

3. リスト・ハンドラーによって参照される管理対象 Bean 用のカスタム・コードを追加します。

   以下の例では、TaskPool 管理対象 Bean 用のカスタム・コードを追加する方法を示します。

```
public class MyTaskQuery implements Query {

  public List execute throws ClientException {

    // Examine the faces-config file for a managed bean "htmConnection".
    //
    FacesContext ctx = FacesContext.getCurrentInstance();
    Application  app = ctx.getApplication();
```

```
    ValueBinding htmVb = app.createValueBinding("#{htmConnection}");
    htmConnection = (HTMConnection) htmVb.getValue(ctx);
    HumanTaskManagerService taskService =
         htmConnection.getHumanTaskManagerService();

    // Then call the actual query method on the Human Task Manager service.
    //
    QueryResultSet queryResult = taskService.query(
     "DISTINCT TASK.TKIID, TASK.NAME, TASK.KIND, TASK.STATE, TASK.TYPE,"
      + "TASK.STARTED, TASK.ACTIVATED, TASK.DUE, TASK.EXPIRES, TASK.PRIORITY" ,
      "TASK.KIND IN(101,102,105) AND TASK.STATE IN(2)
        AND WORK_ITEM.REASON IN (1)",
      null,
      null,
      null);
    List applicationObjects = transformToTaskList ( queryResult );
    return applicationObjects ;
  }


  private List transformToTaskList(QueryResultSet result) {

ArrayList array = null;
int entries = result.size();
array = new ArrayList( entries );

// Transforms each row in the QueryResultSet to a task instance beans.
 for (int i = 0; i < entries; i++) {
   result.next();
   array.add( new TaskInstanceBean( result, connection ));
 }
 return array ;
  }
}
```

TaskPoolQuery Bean は、Java オブジェクトのプロパティーを照会します。この Bean は、com.ibm.bpc.clientcore.Query インターフェースをインプリメントする必要があります。リスト・ハンドラーは、内容を最新表示するときに、照会の execute メソッドを呼び出します。呼び出しによって、Java オブジェクトのリストが戻されます。getType メソッドは、戻された Java オブジェクトのクラス名を戻す必要があります。

これで、JSF アプリケーションは、例えば状態、種類、所有者、ユーザーが使用可能なタスク・インスタンスのオリジネーターなどの、要求されたオブジェクトのリストのプロパティーを表示する JavaServer ページを含むようになります。

List コンポーネント: タグ定義:

Business Process Choreographer Explorer List コンポーネントは、例えば、タスク、アクティビティー、プロセス・インスタンス、プロセス・テンプレート、作業項目、およびエスカレーションなどの、オブジェクトのリストをテーブル内に表示します。

List コンポーネントは、JSF コンポーネント・タグである `bpe:list` と `bpe:column` から構成されます。`bpe:column` タグは、`bpe:list` タグのサブエレメントです。

コンポーネント・クラス

com.ibm.bpe.jsf.component.ListComponent

構文例

```
<bpe:list model="#{ProcessTemplateList}">
          rows="20"
          styleClass="list"
          headerStyleClass="listHeader"
          rowClasses="normal">

    <bpe:column name="name" action="processTemplateDetails"/>
    <bpe:column name="validFromTime"/>
    <bpe:column name="executionMode" label="Execution mode"/>
    <bpe:column name="state" converterID="my.state.converter"/>
    <bpe:column name="autoDelete"/>
    <bpe:column name="description"/>

</bpe:list>
```

タグ属性

bpe:list タグの本体には、bpe:column タグのみを含めることができます。テーブルがレンダリングされるとき、List コンポーネントは、アプリケーション・オブジェクトのリストについて処理を繰り返し、列ごとに特定のオブジェクトを提供します。

表 31. bpe:list 属性

属性	必須	説明
model	はい	com.ibm.bpe.jsf.handler.BPCListHandler クラスの管理対象 Bean 用の値バインディング。
styleClass	いいえ	タイトル、行、およびページ送りボタンを含むテーブル全体のレンダリング用のカスケーディング・スタイル・シート (CSS) スタイル。
headerStyleClass	いいえ	テーブル・ヘッダーのレンダリング用の CSS スタイル・クラス。
cellStyleClass	いいえ	個々のテーブル・セルのレンダリング用の CSS スタイル・クラス。
buttonStyleClass	いいえ	フッター領域内のボタンのレンダリング用の CSS スタイル・クラス。
rowClasses	いいえ	テーブル内の行のレンダリング用の CSS スタイル・クラス。
rows	いいえ	ページに表示される行数。項目数が行数を超える場合は、テーブルの最後にページ送りボタンが表示されます。この属性では、値の式はサポートされていません。
checkbox	いいえ	複数の項目を選択するためのチェック・ボックスを提供するかどうかを決定します。この属性には true または false のいずれかの値が使用されます。

表 32. bpe:column 属性

属性	必須	説明
name	はい	この列に表示されるオブジェクト・プロパティーの名前。この名前は、対応するクライアント・モデル・クラスで定義されているように、名前付きプロパティーに対応していなければなりません。
action	いいえ	この属性は、結果ストリングとして指定された場合、JavaServer Faces (JSF) ナビゲーション・ハンドラーによって使用される結果を定義して、次のページを決定します。 この属性がメソッド・バインディング (#{.....}) として指定された場合、呼び出されるメソッドのシグニチャーは `String method()` であり、その戻り値は、JSF ナビゲーション・ハンドラーによって次のページを決定するために使用されます。
label	いいえ	列のヘッダー内またはテーブル・ヘッダー行のセル内に表示されるラベル。この属性が設定されていない場合、クライアント・モデル・クラスによってデフォルト・ラベルが提供されます。
converterID	いいえ	JSF 構成ファイルでコンバーターを登録するために使用される ID。コンバーター ID が指定されない場合、リストに表示されるオブジェクトのインプリメンテーションには、現在のプロパティー用のコンバーターの定義が含まれます。List コンポーネントはこのコンバーターを使用します。

JSF アプリケーションへの Details コンポーネントの追加

Business Process Choreographer Explorer Details コンポーネントを使用して、タスク、作業項目、アクティビティー、プロセス・インスタンス、およびプロセス・テンプレートのプロパティーを表示します。

1. Details コンポーネントを JavaServer Pages (JSP) ファイルに追加します。

   `bpe:details` タグを `<h:form>` タグに追加します。 `bpe:details` タグには、モデル属性が含まれていなければなりません。`bpe:property` タグを使用して Details コンポーネントにプロパティーを追加することができます。 Details コンポーネントにプロパティーが含まれていない場合は、オブジェクトのすべてのプロパティーが表示されます。

   以下の例では、Details コンポーネントを追加して、タスク・インスタンスのプロパティーのいくつかを表示する方法を示します。

```
<h:form>

   <bpe:details model="#{TaskInstanceDetails}">
      <bpe:property name="displayName" />
```

```
      <bpe:property name="owner" />
      <bpe:property name="kind" />
      <bpe:property name="state" />
      <bpe:property name="escalated" />
      <bpe:property name="suspended" />
      <bpe:property name="originator" />
      <bpe:property name="activationTime" />
      <bpe:property name="expirationTime" />
   </bpe:details>

</h:form>
```

モデル属性は、TaskInstanceDetails という管理対象 Bean を参照します。Bean は、Java オブジェクトのプロパティーを提供します。

2. bpe:details タグで参照されている管理対象 Bean を構成します。

   Details コンポーネントの場合、この管理対象 Bean は、com.ibm.bpe.jsf.handler.BPCDetailsHandler クラスのインスタンスでなければなりません。このハンドラー・クラスは、Java オブジェクトをラップし、そのパブリック・プロパティーを Details コンポーネントに公開します。

   以下の例では、TaskInstanceDetails 管理対象 Bean を構成ファイルに追加する方法を示します。

```
<managed-bean>
   <managed-bean-name>TaskInstanceDetails</managed-bean-name>
   <managed-bean-class>com.ibm.bpe.jsf.handler.BPCDetailsHandler</managed-bean-class>
   <managed-bean-scope>session</managed-bean-scope>
   <managed-property>
      <property-name>type</property-name>
      <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
   </managed-property>
</managed-bean>
```

   例では、TaskInstanceDetails Bean に構成可能な type プロパティーが含まれることを示しています。タイプ・プロパティーの値は、Bean クラス (com.ibm.task.clientmodel.bean.TaskInstanceBean) を指定します。そのクラスのプロパティーは、表示された詳細の行に示されます。

これで、JSF アプリケーションは、例えばタスク・インスタンスの詳細などの、指定されたオブジェクトの詳細を表示する JavaServer ページを含むようになります。

Details コンポーネント: タグ定義:

Business Process Choreographer Explorer Details コンポーネントは、タスク、作業項目、アクティビティー、プロセス・インスタンス、およびプロセス・テンプレートのプロパティーを表示します。

Details コンポーネントは、JSF コンポーネント・タグである bpe:details と bpe:property から構成されます。bpe:property タグは、bpe:details タグのサブエレメントです。

コンポーネント・クラス

com.ibm.bpe.jsf.component.DetailsComponent

構文例

```
<bpe:details model="#{MyActivityDetails}">
    <bpe:property name="name"/>
    <bpe:property name="owner"/>
    <bpe:property name="activated"/>
</bpe:details>
```

```
<bpe:details model="#{MyActivityDetails}" style="style" styleClass="cssStyle">
             style="style"
             styleClass="cssStyle"
</bpe:details>
```

タグ属性

bpe:property タグを使用して、表示される属性のサブセットおよびこれらの属性が表示される順序の両方を指定します。詳細タグに属性タブが含まれていない場合、モデル・オブジェクトの使用可能な属性がすべてレンダリングされます。

表 33. bpe:details 属性

属性	必須	説明
model	はい	com.ibm.bpe.jsf.handler.BPCDetailsHandler クラスの管理対象 Bean 用の値バインディング。
styleClass	いいえ	HTML エレメントのレンダリング用の、カスケーディング・スタイル・シート・スタイル (CSS) クラス。
columnClasses	いいえ	列のレンダリング用の、コンマで区切られた CSS スタイルのリスト。
rowClasses	いいえ	行のレンダリング用の、コンマで区切られた CSS スタイルのリスト。

表 34. bpe:property 属性

属性	必須	説明
name	はい	表示されるプロパティーの名前。この名前は、対応するクライアント・モデル・クラスで定義されているように、名前付きプロパティーに対応していなければなりません。
label	いいえ	プロパティーのラベル。この属性が設定されていない場合、クライアント・モデル・クラスによってデフォルト・ラベルが提供されます。
converterID	いいえ	JavaServer Faces (JSF) 構成ファイルでコンバーターを登録するために使用される ID。

JSF アプリケーションへの CommandBar コンポーネントの追加

Business Process Choreographer Explorer CommandBar コンポーネントを使用して、ボタンを含むバーを表示します。これらのボタンは、オブジェクトの詳細ビューまたはリスト内の選択されたオブジェクトで作動するコマンドを表します。

ユーザーがユーザー・インターフェースのボタンをクリックすると、対応するコマンドが選択されたオブジェクトで実行されます。JSF アプリケーション内の CommandBar コンポーネントを追加および拡張することができます。

1. CommandBar コンポーネントを JavaServer Pages (JSP) ファイルに追加します。

   bpe:commandbar タグを <h:form> タグに追加します。 bpe:commandbar タグには、モデル属性が含まれていなければなりません。

   以下の例では、タスク・インスタンス・リストの refresh および claim コマンドを提供する CommandBar コンポーネントを追加する方法を示します。

   ```
   <h:form>

      <bpe:commandbar model="#{TaskInstanceList}">
         <bpe:command commandID="Refresh" >
                      action="#{TaskInstanceList.refreshList}"
                      label="Refresh"/>

         <bpe:command commandID="MyClaimCommand" >
                      label="Claim" >
                      commandClass="<customcode>"/>
      </bpe:commandbar>

   </h:form>
   ```

   model 属性は、管理対象 Bean を参照します。この Bean は、ItemProvider インターフェースをインプリメントし、選択された Java オブジェクトを提供する必要があります。CommandBar コンポーネントは、通常、同一の JSP ファイル内の List コンポーネントまたは Details コンポーネントのいずれかと共に使用されます。一般に、タグで指定されたモデルは、同一ページの List コンポーネントまたは Details コンポーネントで指定されたモデルと同じです。そのため、例えば List コンポーネントの場合、コマンドはリスト内の選択された項目に対して作動します。

   この例では、 **model** 属性は TaskInstanceList 管理対象 Bean を参照します。この Bean は、選択されたオブジェクトをタスク・インスタンス・リストに示します。この Bean は ItemProvider インターフェースをインプリメントする必要があります。 このインターフェースは、BPCListHandler クラスおよび BPCDetailsHandler クラスによってインプリメントされます。

2. **オプション:** bpe:commandbar タグで参照されている管理対象 Bean を構成します。

   CommandBar **model** 属性が、例えばリスト・ハンドラーまたは詳細ハンドラー用に既に構成済みの管理対象 Bean を参照する場合、それ以上の構成は必要ありません。これらのハンドラーのいずれかの構成を変更する場合、または異なる管理対象 Bean を使用する場合は、ItemProvider インターフェースをインプリメントする管理対象 Bean を JSF 構成ファイルに追加してください。

3. カスタム・コマンドをインプリメントするコードを JSF アプリケーションに追加します。

   以下のコード断片は、コマンド・バーを拡張するコマンド・クラスの作成方法を示します。このコマンド・クラス (MyClaimCommand) は、JSP ファイル内の bpe:command タグで参照されます。

このコマンドは、正しい数の項目が選択されていることなどの前提条件を検査します。次に、ヒューマン・タスク API の HumanTaskManagerService への参照を検索します。コマンドは、選択されたオブジェクトに対して操作を繰り返し、それらの処理を試みます。HumanTaskManagerService API を通じて、ID によってタスクが要求されます。例外が発生しなければ、対応する TaskInstanceBean オブジェクトの状態が更新されます。このアクションでは、オブジェクトの値を再度サーバーから検索することが避けられます。

```
public class MyClaimCommand implements Command {

  public String execute(List selectedObjects) throws ClientException {
     if( selectedObjects != null && selectedObjects.size() > 0 ) {
        try {
         // Determine HumanTaskManagerService from an HTMConnection bean.
         // Configure the bean in the faces-config.xml for easy access
         // in the JSF application.
         FacesContext ctx = FacesContext.getCurrentInstance();
         ValueBinding vb =
           ctx.getApplication().createValueBinding("{htmConnection}");
         HTMConnection htmConnection = (HTMConnection) htmVB.getValue(ctx);
         HumanTaskManagerService htm =
            htmConnection.getHumanTaskManagerService();

         Iterator iter = selectedObjects.iterator() ;
         while( iter.hasNext() ) {
           try {
                TaskInstanceBean task = (TaskInstanceBean) iter.next() ;
                TKIID tiid = task.getID() ;

                htm.claim( tiid ) ;
                task.setState( new Integer(TaskInstanceBean.STATE_CLAIMED ) ) ;

           }
           catch( Exception e ) {
             ;   // Error while iterating or claiming task instance.
                  // Ignore for better understanding of the sample.
           }
         }
       }
       catch( Exception e ) {
         ;  // Configuration or communication error.
           // Ignore for better understanding of the sample
       }
     }
     return null;
  }

 // Default implementations
 public boolean isMultiSelectEnabled() { return false; }
 public boolean[] isApplicable(List itemsOnList) {return null; }
 public void setContext(Object targetModel) {; // Not used here }
}
```

コマンドは以下のように処理されます。

a. ユーザーがコマンド・バーの対応するボタンをクリックすると、コマンドが起動されます。CommandBar コンポーネントは、**model** 属性で指定された項目プロバイダーから選択された項目を検索し、選択されたオブジェクトのリストを commandClass インスタンスの execute メソッドに渡します。

b. **commandClass** 属性は、コマンド・インターフェースをインプリメントするカスタム・コマンド・インプリメンテーションを参照します。つまり、コマンドは public String execute(List selectedObjects) throws ClientException メソ

ッドをインプリメントする必要があります。コマンドが戻した結果は、JSF アプリケーションの次のナビゲーション規則を決定するために使用されます。

c. コマンドの完了後、CommandBar コンポーネントは **action** 属性を評価します。 **action** 属性は、静的ストリングである場合も、`public String Method()` というシグニチャーの JSF アクション・メソッドへのメソッド・バインディングである場合もあります。 **action** 属性を使用して、コマンド・クラスの結果をオーバーライドするか、またはナビゲーション規則の結果を明示的に指定します。コマンドが ErrorsInCommandException 例外以外の例外を生成した場合、 **action** 属性は処理されません。

これで、JSF アプリケーションは、カスタマイズされたコマンド・バーをインプリメントする JavaServer ページを含むようになります。

CommandBar コンポーネント: タグ定義:

Business Process Choreographer Explorer CommandBar コンポーネントは、ボタンを含むバーを表示します。これらのボタンは、詳細ビュー内のオブジェクトまたはリスト内の選択されたオブジェクトに作動します。

CommandBar コンポーネントは、JSF コンポーネント・タグである `bpe:commandbar` と `bpe:command` から構成されます。 `bpe:command` タグは、 `bpe:commandbar` タグのサブエレメントです。

コンポーネント・クラス

com.ibm.bpe.jsf.component.CommandBarComponent

構文例

```
<bpe:commandbar model="#{TaskInstanceList}">

    <bpe:command
       commandID="Work on"
       label="Work on..."
       commandClass="com.ibm.bpc.explorer.command.WorkOnTaskCommand"
       context="#{TaskInstanceDetailsBean}"/>

    <bpe:command
       commandID="Cancel"
       label="Cancel"
       commandClass="com.ibm.task.clientmodel.command.CancelClaimTaskCommand"
       context="#{TaskInstanceList}"/>

</bpe:commandbar>
```

タグ属性

表 35. bpe:commandbar 属性

属性	必須	説明
model	はい	ItemProvider インターフェースをインプリメントする管理対象 Bean に対する値バインディング式。通常、この管理対象 Bean は、同一の JavaServer Pages (JSP) ファイル内の List コンポーネントまたは Details コンポーネントによって CommandBar コンポーネントとして使用される com.ibm.bpe.jsf.handler.BPCListHandler クラスまたは com.ibm.bpe.jsf.handler.BPCDetailsHandler クラスです。
styleClass	いいえ	バーのレンダリング用のカスケーディング・スタイル・シート (CSS) スタイル。
buttonStyleClass	いいえ	コマンド・バー内のボタンのレンダリング用の CSS スタイル。

表 36. bpe:command 属性

属性	必須	説明
commandID	はい	コマンドの ID。
commandClass	はい	起動されるコマンド・クラス。
action	いいえ	シグニチャーが `String method()` である JavaServer Faces (JSF) アクション・メソッド。この **action** メソッドによって戻される値、または直接リテラルとして指定された値が、コマンドの execute メソッドによって戻されるターゲットをオーバーライドします。コマンドが ErrorsInCommandException 例外以外の例外を生成した場合、 **action** 属性は処理されません。 この属性は、結果ストリングとして指定された場合、JSF ナビゲーション・ハンドラーによって使用される結果を定義して、ナビゲーション規則と次に表示するページを決定します。 この属性がメソッド・バインディング (#{.....}) として指定された場合、呼び出されるメソッドのシグニチャーは `String method()` です。その戻り値が JSF ナビゲーション・ハンドラーによって使用され、ナビゲーション規則と次に表示するページを決定します。
label	はい	コマンド・バーでレンダリングされるボタンのラベル。

表 36. bpe:command 属性 (続き)

属性	必須	説明
styleClass	いいえ	ボタンのレンダリング用の CSS スタイル。このスタイルは、コマンド・バーに定義されたボタン・スタイルをオーバーライドします。
context	いいえ	管理対象 Bean を参照する値バインディング式。コマンドがターゲット・ページまたは Bean を初期化する必要がある場合に、この属性を使用します。

JSF アプリケーションへの Message コンポーネントの追加

Business Process Choreographer Explorer Message コンポーネントを使用して、JavaServer Faces (JSF) アプリケーション内で、データ・オブジェクトおよびプリミティブ型をレンダリングします。

メッセージ型がプリミティブ型である場合、ラベルおよび入力フィールドがレンダリングされます。メッセージ型がデータ・オブジェクトである場合、コンポーネントはオブジェクトを全探索し、オブジェクト内のエレメントをレンダリングします。

1. Message コンポーネントを JavaServer Pages (JSP) ファイルに追加します。

   `bpe:form` タグを `<h:form>` タグに追加します。 `bpe:form` タグには、 model 属性が含まれていなければなりません。

   以下の例では、Message コンポーネントを追加する方法を示します。

   ```
   <h:form>

      <h:outputText value="Input Message" />
      <bpe:form model="#{MyHandler.inputMessage}" readOnly="true" />

      <h:outputText value="Output Message" />
      <bpe:form model="#{MyHandler.outputMessage}" />

   </h:form>
   ```

   Message コンポーネントの **model** 属性は、com.ibm.bpc.clientcore.MessageWrapper オブジェクトを参照します。このラッパー・オブジェクトは、サービス・データ・オブジェクト (SDO) オブジェクトか、または `int` や `boolean` などの Java プリミティブ型のいずれかをラップします。例では、メッセージは MyHandler 管理対象 Bean のプロパティーによって提供されます。

2. `bpe:form` タグで参照されている管理対象 Bean を構成します。

   以下の例では、MyHandler 管理対象 Bean を構成ファイルに追加する方法を示します。

   ```
   <managed-bean>
   <managed-bean-name>MyHandler</managed-bean-name>
   <managed-bean-class>com.ibm.bpe.sample.jsf.MyHandler</managed-bean-class>
   <managed-bean-scope>session</managed-bean-scope>

      <managed-property>
         <property-name>type</property-name>
   ```

```
      <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
   </managed-property>

</managed-bean>
```

3. JSF アプリケーションにカスタム・コードを追加します。

   以下の例では、入力メッセージおよび出力メッセージをインプリメントする方法を示します。

```
public class MyHandler implements ItemListener {

  private TaskInstanceBean taskBean;
  private MessageWrapper inputMessage, outputMessage

  /* Listener method, e.g. when a task instance was selected in a list handler.
   * Ensure that the handler is registered in the faces-config.xml or manually.
   */
  public void itemChanged(Object item) {
    if( item instanceof TaskInstanceBean ) {
        taskBean = (TaskInstanceBean) item ;
    }
  }

  /* Get the input message wrapper
   */
  public MessageWrapper getInputMessage() {
    try{
        inputMessage = taskBean.getInputMessageWrapper() ;
    }
    catch( Exception e ) {
        ;       //...ignore errors for simplicity
    }
    return inputMessage;
  }

  /* Get the output message wrapper
   */
  public MessageWrapper getOutputMessage() {
    // Retrieve the message from the bean. If there is no message, create
    // one if the task has been claimed by the user. Ensure that only
    // potential owners or owners can manipulate the output message.
    try{
        outputMessage = taskBean.getOutputMessageWrapper();
        if( outputMessage == null
         && taskBean.getState() == TaskInstanceBean.STATE_CLAIMED ) {
            HumanTaskManagerService htm = getHumanTaskManagerService();
            outputMessage = new MessageWrapperImpl();
            outputMessage.setMessage(
               htm.createOutputMessage( taskBean.getID() ).getObject()
            );
         }
    }
    catch( Exception e ) {
        ;       //...ignore errors for simplicity
    }
    return outputMessage
  }
}
```

   MyHandler 管理対象 Bean は、リスト・ハンドラーへの項目リスナーとして登録できるように、com.ibm.jsf.handler.ItemListener インターフェースをインプリメントします。ユーザーがリスト内の項目をクリックすると、選択された項目について MyHandler Bean が itemChanged(Object item) メソッドで通知されます。ハ

ンドラーは、項目タイプを検査してから、関連した TaskInstanceBean オブジェクトへの参照を保管します。このインターフェースを使用するには、faces-config.xml ファイル内の適切なリスト・ハンドラーにエントリーを追加します。

MyHandler Bean は、getInputMessage および getOutputMessage メソッドを提供します。これらのメソッドはどちらも、MessageWrapper オブジェクトを戻します。メソッドは、参照されたタスク・インスタンス Bean への呼び出しを委任します。例えばメッセージが設定されていないなどの理由で、タスク・インスタンス Bean がヌルを戻した場合、ハンドラーは新規に空のメッセージを作成して保管します。Message コンポーネントは MyHandler Bean が提供するメッセージを表示します。

これで、JSF アプリケーションは、データ・オブジェクトおよびプリミティブ型をレンダリング可能な JavaServer ページを含むようになります。

Message コンポーネント: タグ定義:

Business Process Choreographer Explorer Message コンポーネントは、JavaServer Faces (JSF) アプリケーション内で、`commonj.sdo.DataObject` オブジェクトと、整数およびストリングなどのプリミティブ型をレンダリングします。

Message コンポーネントは、JSF コンポーネント・タグである `bpe:form` から構成されます。

コンポーネント・クラス

com.ibm.bpe.jsf.component.MessageComponent

構文例

```
<bpe:form model="#{TaskInstanceDetailsBean.inputMessageWrapper}"
          simplification="true" readOnly="true"
          styleClass4table="messageData"
          styleClass4output="messageDataOutput">
</bpe:form>
```

タグ属性

表 37. bpe:form 属性

属性	必須	説明
model	はい	`commonj.sdo.DataObject` オブジェクトまたは `com.ibm.bpc.clientcore.MessageWrapper` オブジェクトを参照する値バインディング式。
simplification	いいえ	この属性が `true` に設定されている場合は、カーディナリティーがゼロまたは 1 のプロパティーが表示されます。デフォルトでは、この属性は `true` に設定されています。

表 37. bpe:form 属性 (続き)

属性	必須	説明
readOnly	いいえ	この属性が true に設定されている場合は、読み取り専用フォームのみがレンダリングされます。デフォルトでは、この属性は false に設定されています。
style4validinput	いいえ	有効な入力のレンダリング用のカスケーディング・スタイル・シート (CSS) スタイル。
style4invalidinput	いいえ	無効な入力のレンダリング用の CSS スタイル。
styleClass4validInput	いいえ	有効な入力のレンダリング用の CSS クラス名。
styleClass4invalidInput	いいえ	無効な入力のレンダリング用の CSS クラス名。
styleClass4output	いいえ	出力エレメントのレンダリング用の CSS スタイル・クラス名。
styleClass4table	いいえ	Message コンポーネントによって提供されるテーブルのレンダリング用の、CSS テーブル・スタイルのクラス名。
buttonStyleClass	いいえ	配列またはリストに作動するボタン用の CSS スタイル。

クライアント・モデル・オブジェクトのマッピング

クライアント・モデル・オブジェクトは、Business Process Choreographer API の対応するインターフェースをインプリメントします。

インターフェースをこのようにラップすることにより、ロケールを区別するラベルと、プロパティーのセット用のコンバーターが提供されます。以下の表に、Business Process Choreographer インターフェースから対応するクライアント・モデル・オブジェクトへのマッピングを示します。

表 38. Business Process Choreographer インターフェースからクライアント・モデル・オブジェクトへのマッピング

Business Process Choreographer インターフェース	クライアント・モデル・オブジェクト・クラス
com.ibm.bpe.api.ActivityInstanceData	com.ibm.bpe.clientmodel.bean.ActivityInstanceBean
com.ibm.bpe.api.ActivityServiceTemplateData	com.ibm.bpe.clientmodel.bean.ActivityServiceTemplateBean
com.ibm.bpe.api.ProcessInstanceData	com.ibm.bpe.clientmodel.bean.ProcessInstanceBean
com.ibm.bpe.api.ProcessTemplateData	com.ibm.bpe.clientmodel.bean.ProcessTemplateBean
com.ibm.task.api.Escalation	com.ibm.task.clientmodel.bean.EscalationBean
com.ibm.task.api.Task	com.ibm.task.clientmodel.bean.TaskInstanceBean
com.ibm.task.api.TaskTemplate	com.ibm.task.clientmodel.bean.TaskTemplateBean

第 6 章 デプロイ

ビジネス・プロセスおよびヒューマン・タスク・アプリケーションのインストール

ビジネス・プロセスまたはヒューマン・タスク、あるいはこの両方を含む Service Component Architecture (SCA) Enterprise JavaBeans (EJB) モジュールをデプロイメント・ターゲットに配布することができます。 サーバーまたはクラスターをデプロイメント・ターゲットとすることができます。

アプリケーションのインストール先とするアプリケーション・サーバーまたはクラスターごとに、ビジネス・プロセス・コンテナーまたはタスク・コンテナーがインストールされ、構成されていることを確認してください。

ビジネス・プロセスまたはヒューマン・タスク・アプリケーションをインストールする前に、以下の条件をすべて満たしていることを確認してください。

- アプリケーションのインストール先とするサーバーが稼動している。
- 各クラスターで、プロセスまたはタスクを使用して Enterprise JavaBeans モジュールをインストールする対象のサーバーが少なくとも 1 台実行されている。

ビジネス・プロセスおよびタスク・アプリケーションを、管理コンソールやコマンド行から、または管理スクリプトを実行してインストールすることができます。管理スクリプトを実行してビジネス・プロセス・アプリケーションまたはヒューマン・タスク・アプリケーションをインストールする場合、サーバー接続が必要です。 `-conntype NONE` オプションをインストール・オプションとして使用しないでください。

1. アプリケーションをクラスター上にインストールする場合、アプリケーションがクラスターにちなんで名前を付けられたデータ・ソースを使用していることを確認します。

   例えば、アプリケーションがデフォルト・データ・ソース `BPEDB` を使用して生成された場合、アプリケーションのデータ・ソースを BPEDB_*cluster_name* に変更します。ここで、*cluster_name* は、アプリケーションのインストール先のクラスターの名前です。
2. アプリケーションをインストールします。

すべてのビジネス・プロセス・テンプレートおよびヒューマン・タスク・テンプレートが、開始状態になります。これらのテンプレートからプロセス・インスタンスおよびタスク・インスタンスを作成できます。

プロセス・インスタンスまたはタスク・インスタンスを作成するには、アプリケーションを始動する必要があります。

モデルのデプロイ

WebSphere Integration Developer またはサービス・デプロイがプロセスのデプロイメント・コードを生成するとき、プロセスまたはタスク・モデルの構成要素は、さまざまな Java 2 Enterprise Edition (J2EE) 構成要素および成果物にマップされます。すべてのデプロイメント・コードは、エンタープライズ・アプリケーション (EAR) ファイルにパッケージ化されます。デプロイするモデルの新規バージョンごとに、新しいエンタープライズ・アプリケーションにパッケージ化する必要があります。

ビジネス・プロセス・モデルまたはヒューマン・タスク・モデルの J2EE 構成要素で構成されるエンタープライズ・アプリケーションをインストールすると、モデルの構成要素が必要に応じて、*プロセス・テンプレート* または*タスク・テンプレート*として Business Process Choreographer データベースに格納されます。データベース・システムが稼働していない場合、またはデータベース・システムにアクセスできない場合、デプロイは失敗します。デフォルトでは、新しくインストールされたテンプレートは、開始済み状態となります。ただし、新しくインストールされたエンタープライズ・アプリケーションの場合は、停止状態になります。インストール済みのエンタープライズ・アプリケーションは、個々に開始したり停止したりすることができます。

新バージョンのプロセス・テンプレートまたはタスク・テンプレートの名前は同じですが、有効開始日属性が異なります。プロセス・テンプレートやタスク・テンプレートの多くの異なるバージョンを、それぞれ、異なるエンタープライズ・アプリケーションにデプロイできます。ただし、1 つのプロセスの 2 つのバージョンが同じ有効開始日を持つことはできません。1 つのプロセスの異なるバージョンをインストールする場合は、バージョンごとに異なる有効開始日を指定してください。データベースには、異なるプロセス・バージョンのすべてが格納されます。

有効開始日を指定しない場合、日付は次のように決定されます。

- ヒューマン・タスクの場合、有効開始日はアプリケーションがインストールされた日付です。
- ビジネス・プロセスの場合、有効開始日はプロセスがモデル化された日付です。

ビジネス・プロセス・アプリケーションの対話式デプロイ

wsadmin ツールおよび installInteractive スクリプトを使用して、実行時に対話式にアプリケーションをインストールできます。このスクリプトを使用すると、管理コンソールを使用してアプリケーションをインストールした場合に変更できない設定を変更できます。

次のステップを実行して、ビジネス・プロセス・アプリケーションを対話式にインストールします。

1. wsadmin ツールを開始します。

   profile_root/bin ディレクトリーで、 `wsadmin` と入力します。
2. アプリケーションをインストールします。

   wsadmin コマンド行プロンプトで、次のコマンドを入力します。

   ```
   $AdminApp installInteractive application.ear
   ```

ここで、*application.ear* は、処理アプリケーションを含むエンタープライズ・アーカイブ・ファイルの修飾名です。 アプリケーションの値を変更できる一連のタスクのためのプロンプトが出されます。

3. 構成変更を保管します。

   wsadmin コマンド行プロンプトで、次のコマンドを入力します。

   ```
   $AdminConfig save
   ```

   更新をマスター構成リポジトリーに転送するためには、変更を保管する必要があります。スクリプト処理が終了したときに変更を保管していない場合、変更は廃棄されます。

処理アプリケーションのデータ・ソースおよび設定参照設定の構成

特定のデータベース・インフラストラクチャーで SQL ステートメントを実行する処理アプリケーションを構成しなければならない場合があります。これらの SQL ステートメントは情報サービス・アクティビティーから来たものであるか、プロセスのインストールまたはインスタンスの開始時に実行するステートメントです。

アプリケーションをインストールする場合、次のタイプのデータ・ソースを指定できます。

- プロセスのインストール時に SQL ステートメントを実行するデータ・ソース
- プロセス・インスタンスの開始時に SQL ステートメントを実行するデータ・ソース
- SQL 断片アクティビティーを実行するデータ・ソース

SQL 断片アクティビティーの実行に必要なデータ・ソースは、タイプ tDataSource の BPEL 変数で定義されます。 SQL 断片アクティビティーが必要とするデータベース・スキーマおよびテーブル名は、タイプ tSetReference の BPEL 変数で定義されます。これら両方の変数の初期値を構成できます。

wsadmin ツールを使用してデータ・ソースを指定できます。

1. wsadmin ツールを使用して処理アプリケーションを対話式にインストールします。
2. データ・ソースおよび設定参照を更新するタスクまでステップスルーします。

   環境に合わせてこれらの設定を構成します。次の例は、以下の各タスクで変更できる例を示しています。

3. 変更を保管します。

例: wsadmin ツールを使用したデータ・ソースと設定参照の更新

「**データ・ソースの更新**」タスクでは、プロセスのインストール時またはプロセスの開始時に使用される初期変数値およびステートメントのデータ・ソース値を変更できます。「**設定参照の更新**」タスクでは、データベース・スキーマとテーブル名に関連した設定を構成できます。

```
Task[24]: データ・ソースの更新

//プロセス開始時の初期変数値のデータ・ソース値を変更する

Process name:  Test
```

```
// プロセス・テンプレートの名前
Process start or installation time:  Process start
// プロセスの開始またはプロセスのインストール時に
//指定した値を評価するかどうかを指示する
Statement or variable:  Variable
// データ・ソース変数を変更することを指示する
Data source name:  MyDataSource
// 変数の名前
JNDI name:[jdbc/sample]:jdbc/newName
// JNDI 名を jdbc/newName に設定する
Task[25]: 設定参照の更新

// BPEL 変数の初期値として使用される設定参照値を変更する

Process name:  Test
// プロセス・テンプレートの名前
Variable:  SetRef
// BPEL 変数名
JNDI name:[jdbc/sample]:jdbc/newName
// 設定参照のデータ・ソースの JNDI 名を jdbc/newName に設定する
Schema name: [IISAMPLE]
// データベース・スキーマの名前
Schema prefix: []:
// スキーマ名のプレフィックス。
// この設定はスキーマ名が生成された場合にのみ適用される。
Table name: [SETREFTAB]: NEWTABLE
// データベース表の名前を NEWTABLE に設定する。
Table prefix: []:
// テーブル名のプレフィックス。
// この設定は表の名前が生成された場合にのみ適用される。
```

サーバーが稼動していないクラスターにプロセス・アプリケーションをインストール可能な場合

このトピックでは、サーバーが稼動していないクラスターにアプリケーションをインストールする必要があるような、例外的な状況について説明します。

サーバー上のビジネス・プロセス・アプリケーションのインストール時に、対応するビジネス・プロセス・コンテナーのデータ・ソースの Java Naming and Directory Interface (JNDI) 名を解決する必要があります。そのため、サーバー接続がなければ、アプリケーションをインストールできません。Network Deployment (ND) 環境では、このサーバーはデプロイメント・マネージャーです。

撤廃された制約事項

ND 環境でクラスターにビジネス・プロセス・アプリケーションをインストールする場合、以下の条件が真である場合は、クラスター内のサーバーが稼動している必要はありません。

- 必要なデータ・ソースがセル・レベルで定義されている。
- プロセス・アプリケーションがヒューマン・タスクを指定していない。

ヒューマン・タスクを持たないプロセス・アプリケーションでは、アプリケーション・サーバーのネーム・スペースでの検索操作が前に失敗した場合、データ・ソース検索操作はデプロイメント・マネージャーのネーム・スペース内で実行されます。アプリケーションが正常にインストールされた場合は、アプリケーション・サーバー・ネーム・スペース内部のデータ・ソース検索操作の失敗を示す SystemOut.log ファイル内のエラー・メッセージは無視してください。

機能する場合

- デプロイメント・マネージャー・ネーム・スペース内部の検索操作は、データ・ソース JNDI 名がセル・レベルで定義されている場合のみ、正常に実行されます。
- ウィザードを使用して、スタンドアロン・サーバー上のビジネス・プロセス・コンテナーまたはヒューマン・タスク・コンテナーを構成した場合、データ・ソースはサーバー・レベルで定義されます。アプリケーション・サーバーのインストール済み環境の ProcessChoreographer/config ディレクトリーで提供されている構成スクリプト bpeconfig.jacl を使用した場合にも、同じことが当てはまります。この場合、手動でデータ・ソースをセル・レベルで定義し、ビジネス・プロセス・コンテナーのインストール時にこのデータ・ソースを使用する必要があります。
- クラスター・メンバーでウィザードを使用してビジネス・プロセス・コンテナーを構成した場合、データ・ソースは自動的にセル・レベルで定義されます。JNDI 名は、クラスター名によってスコープが決まります。アプリケーション・サーバーのインストール済み環境の ProcessChoreographer/config ディレクトリーで提供されている構成スクリプト bpeconfig.jacl を使用した場合にも、同じことが当てはまります。この場合、何も手動で変更する必要はありません。

機能しない場合

ヒューマン・タスクを持つプロセス・アプリケーションでは、追加の JNDI 名検索操作によって、スタッフ・プラグイン・プロバイダーを見つける必要があります。したがって、このようなアプリケーションのインストールを確実に正常に実行するため、稼動しているサーバーがクラスターに含まれるようにしてください。

スコープによる副次作用

名前検索の副次作用としては、アプリケーション・サーバーが稼動しておらず、データ・ソースが、サーバーまたはノード・レベルでもセル・レベルのデータ・ソースと同じ名前で定義されている場合、セル・レベルのデータ・ソースが優先されます。これは、結果的に、実行時に使用するものとは異なるデータ・ソースをデプロイメント時に使用する場合があることを意味します。

重要: 名前の競合を回避してください。手動により、セル・レベルでデータ・ソースを定義する場合、クラスター名またはサーバー名、およびノード名のスコープが追加された JNDI 名 (例えば jdbc/BPEDB_*cluster_name*) を使用します。

管理コンソールを使用した、ビジネス・プロセスおよびヒューマン・タスク・アプリケーションのアンインストール

ビジネス・プロセスまたはヒューマン・タスクが含まれるエンタープライズ・アプリケーションをアンインストールするには、以下のアクションを実行します。

1. アプリケーションのすべてのプロセスおよびタスクのテンプレートを停止します。

   このアクションにより、プロセスおよびタスクのインスタンスの作成が回避されます。

a. 管理コンソールのナビゲーション・ペインで、「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」をクリックします。
b. 停止するアプリケーションを選択します。
c. 「関連項目」の下で、「**EJB モジュール**」をクリックして、Enterprise JavaBeans モジュールを選択します。 複数の EJB モジュールがある場合は、ビジネス・プロセスまたヒューマン・タスクが含まれる Service Component Architecture (SCA) モジュールに対応する EJB モジュールを選択してください。 SCA モジュール名の後に EJB を付けると、対応する EJB モジュールを見つけることができます。例えば、SCA モジュールの名前が `TestProcess` であれば、 EJB モジュールは `TestProcessEJB.jar` になります。
d. 「追加プロパティー」の下で、「**ビジネス・プロセス**」か「**ヒューマン・タスク**」、または両方を必要に応じてクリックします。
e. 適切なチェック・ボックスをクリックして、プロセスおよびタスクのテンプレートをすべて選択します。
f. 「**停止**」をクリックします。

ビジネス・プロセス・テンプレートまたはタスク・テンプレートが含まれるすべての EJB モジュールで、このステップを繰り返します。

2. データベース、クラスターごとに少なくとも 1 つのアプリケーション・サーバー、アプリケーションがデプロイされているスタンドアロン・サーバーが動作していることを確認してください。

Network Deployment (ND) 環境では、デプロイメント・マネージャー、すべての ND 管理対象スタンドアロン・アプリケーション・サーバー、および少なくとも 1 台のアプリケーション・サーバーが、アプリケーションがインストールされているそれぞれのクラスターごとに実行されている必要があります。

3. 実行中のプロセス・インスタンスまたはタスク・インスタンスがないこと、および終了状態で autoDelete フラグが false に設定されていないことを確認します。

必要であれば、管理者は、Business Process Choreographer Explorer を使用して、残存するプロセス・インスタンスまたはタスク・インスタンスを削除することができます。

4. アプリケーションを停止してアンインストールするには、以下のようにします。
a. 管理コンソールのナビゲーション・ペインで、「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」をクリックします。
b. アンインストールするアプリケーションを選択し、「**停止**」をクリックします。

アプリケーションでプロセス・インスタンスまたはタスク・インスタンスがまだ存在する場合は、このステップは失敗します。

c. アンインストールするアプリケーションを再度選択し、「**アンインストール**」をクリックします。
d. 「**保管**」をクリックして、変更を保管します。

アプリケーションはアンインストールされます。

管理コマンドを使用した、ビジネス・プロセスおよびヒューマン・タスク・アプリケーションのアンインストール

管理コマンドには、ビジネス・プロセスまたはヒューマン・タスクを含むアプリケーションをアンインストールするための、管理コンソールの代替方法が用意されています。

グローバル・セキュリティーが使用可能に設定されている場合は、使用するユーザー ID にオペレーター権限があることを確認します。

管理クライアントが接続しているサーバー・プロセスが動作していることを確認します。

- ND 環境では、サーバー・プロセスはデプロイメント・マネージャーです。
- スタンドアロン環境では、サーバー・プロセスはアプリケーション・サーバーです。

管理クライアントが自動的にサーバー・プロセスに接続できるよう、 `-conntype NONE` オプションをコマンド・オプションとして使用しないでください。

次の手順で、bpcTemplates.jacl スクリプトを使用して、ビジネス・プロセス・テンプレートまたはヒューマン・タスク・テンプレートを含むアプリケーションをアンインストールする方法を示します。また、アプリケーションをアンインストールするには、そのアプリケーションに属しているテンプレートを停止する必要があります。 bpcTemplates.jacl スクリプトを使用して、1 つの手順でテンプレートを停止およびアンインストールできます。

アプリケーションをアンインストールする前に、 Business Process Choreographer Explorer などを使用して、アプリケーション内のテンプレートに関連したプロセス・インスタンスやタスク・インスタンスを削除できます。bpcTemplates.jacl スクリプトと一緒に **-force** オプションを使用して、テンプレートと関連したインスタンスの削除、テンプレートの停止、テンプレートのアンインストールを 1 ステップで行うこともできます。

注意:
このオプションはすべてのプロセス・インスタンスおよびタスク・インスタンス・データを削除するため、注意して使用する必要があります。

1. Business Process Choreographer サンプル・ディレクトリーに移動します。 次のように入力します。

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. テンプレートを停止して、対応するアプリケーションをアンインストールします。

   ```
   install_root/bin/wsadmin -f bpcTemplates.jacl
                               [-user user_name]
                               [-password user password]
                               -uninstall application_name
                               [-force]
   ```

   各部の意味は、次のとおりです。

user_name
グローバル・セキュリティーが使用可能な場合は、認証用のユーザー ID を提供します。

user_password
グローバル・セキュリティーが使用可能な場合は、認証用のユーザー・パスワードを提供します。

application_name
グローバル・セキュリティーが使用可能な場合は、認証用のユーザー・パスワードを提供します。

-force
アプリケーションがアンインストールされる前に、実行中のインスタンスが停止および削除されます。

アプリケーションはアンインストールされます。

第 7 章 モニター

ビジネス・プロセスとヒューマン・タスクのモニター

プロセスとヒューマン・タスクのモニターは、WebSphere Integration Developer のモニター・ペインから制御されます。監査証跡が有効になっているかどうか、あるいはイベントが発行されるかどうかに関係なく、この方法に従う必要があります。

WebSphere Process Server に組み込まれている Common Event Infrastructure は、イベント・データの管理用の標準の形式およびメカニズムを提供します。

Business Process Choreographer は、監視を必要とする状態が起きて、Common Event Infrastructure サービスが使用可能であるときはいつでもイベントを送信します。このイベントは、Common Base Event 仕様に準拠しています。これらのイベントの処理には、汎用ツールが使用できます。

ユーザー・データ・イベントの作成および送信には、 Java 断片も使用できます。詳しくは、イベント送信に関する Common Event Infrastructure の資料を参照してください。

状態非依存イベント・データ

ビジネス・プロセスとヒューマン・タスクの代わりに発行される Common Base Event には、イベントが作成されたときの状態に依存しない情報が含まれます。このイベント・データは、すべてのビジネス・プロセス・イベントとヒューマン・タスク・イベントで同じです。

次の表に、これらのイベントに含まれる CommonBaseEvent エレメントと sourceComponentID エレメントの属性値を示します。

属性	説明
CommonBaseEvent エレメント	
creationTime	イベントが作成された時刻 (協定世界時 (UTC) 形式)。
globalInstanceId	Common Base Event インスタンスの ID。この ID は自動的に生成されます。
sequenceNumber	イベント・ファクトリーによって発行されるシーケンス番号。
severity	ビジネス・プロセスまたはヒューマン・タスクにイベントが及ぼす影響。この属性は 10 (情報) に設定されます。
version	1.0.1 に設定します。
extensionName	値は、イベントを作成するオブジェクトまたはイベントに依存します。
sourceComponentId エレメント	
component	ビジネス・プロセスとヒューマン・タスクの場合、WPS# と、その後に現行プラットフォームの ID と基盤となるソフトウェア・スタックのバージョン ID を設定します。

属性	説明
componentIdType	ストリング「ProductName」に設定します。
executionEnvironment	オペレーティング・システムを示すストリング。
instanceId	サーバーの ID。この ID のフォーマットは、*cell name/node name/server name* です。区切り文字はプラットフォームに依存します。
location	実行中のサーバーのホスト名に設定します。
locationType	IP アドレスまたはホスト名に設定します。
processId	オペレーティング・システムのプロセス ID。
subcomponent	ビジネス・プロセスの場合、`BFM` に設定します。 ヒューマン・タスクの場合、`HTM` に設定します。
threadId	Java 仮想マシン (JVM) のスレッド ID。
componentType	ビジネス・プロセスの場合、次のように設定します。 www.ibm.com/namespaces/autonomic/Workflow_Engine ヒューマン・タスクの場合、次のように設定します。 www.ibm.com/xmlns/prod/websphere/scdl/human-task

ビジネス・プロセス・イベント

ビジネス・プロセスのために発行されるイベントは、状態非依存データとビジネス・プロセス・イベントに固有のデータで構成されています。ここでは、ビジネス・プロセス・イベントに固有の属性とエレメントについて説明します。

ビジネス・プロセス・イベントには、次に示すイベント内容のカテゴリーがあります。

ビジネス・プロセス固有のイベント・データ

ビジネス・プロセスでは、イベントは、プロセス、アクティビティー、スコープ、リンク、および変数に関連します。これらの各イベント・タイプのオブジェクト固有の内容について説明します。

特に明記されていない限り、オブジェクト固有の内容は、タイプがストリングの *extendedDataElement* XML エレメントとして書き込まれます。

プロセス

プロセス・インスタンスのイベントには、以下のようなオブジェクト固有のイベント内容が含まれています。

属性	説明
processTemplateName	インスタンスが派生したプロセス・テンプレートの名前
processTemplateValidFrom	テンプレートの有効日
processTemplateId	プロセス・テンプレートの ID
processInstanceDescription	**オプション:** プロセス・インスタンスの説明

属性	説明
processInstanceExecutionState	プロセスの状態を表すストリング値。フォーマットは次のとおりです。*state number-state description*。この属性は、以下の値のうちのいずれかをとることができます。 1 - STATE_READY 2 - STATE_RUNNING 3 - STATE_FINISHED 4 - STATE_COMPENSATING 5 - STATE_FAILED 6 - STATE_TERMINATED 7 - STATE_COMPENSATED 8 - STATE_TERMINATING 9 - STATE_FAILING 10 - STATE_INDOUBT 11 - STATE_SUSPENDED 12 - STATE_COMPENSATION_FAILED
PayloadType	完全ストリング

アクティビティーおよびスコープ

アクティビティーおよびスコープには、以下のようなオブジェクト固有のイベント内容が含まれています。

属性	説明
processTemplateName	インスタンスが派生したプロセス・テンプレートの名前。
processTemplateValidFrom	テンプレートの有効日。
activityTemplateName	**オプション:** インスタンスが派生したアクティビティー・テンプレートの名前。
activityInstanceDescription	**オプション:** アクティビティー・インスタンスの説明。

属性	説明
activityKind	アクティビティーの種類を示すストリング値。この値のフォーマットは、次のとおりです。 *kind number-kind description*。この属性は、以下の値のうちのいずれかをとることができます。 3 - KIND_EMPTY 21 - KIND_INVOKE 23 - KIND_RECEIVE 24 - KIND_REPLY 25 - KIND_THROW 26 - KIND_TERMINATE 27 - KIND_WAIT 29 - KIND_COMPENSATE 30 - KIND_SEQUENCE 32 - KIND_SWITCH 34 - KIND_WHILE 36 - KIND_PICK 38 - KIND_FLOW 42 - KIND_SCRIPT 43 - KIND_STAFF 44 - KIND_ASSIGN 45 - KIND_CUSTOM 46 - KIND_RETHROW 47 - KIND_FOR_EACH_SERIAL 48 - KIND_FOR_EACH_PARALLEL 1000 - SQLSnippet 1001 - RetrieveSet 1002 - InvokeInformationService 1003 - AtomicSQLSnippetSequence
state	アクティビティーの状態を表すストリング値。フォーマットは次のとおりです。*state number-state description*。アクティビティーの状態コードは、プロセスで使用する状態コードとは異なります。この属性は、以下の値のうちのいずれかをとることができます。 1 - STATE_INACTIVE 2 - STATE_READY 3 - STATE_RUNNING 4 - STATE_SKIPPED 5 - STATE_FINISHED 6 - STATE_FAILED 7 - STATE_TERMINATED 8 - STATE_CLAIMED 9 - STATE_TERMINATING 10 - STATE_FAILING 11 - STATE_WAITING 12 - STATE_EXPIRED 13 - STATE_STOPPED
bpelId	アクティビティーの wpc:id 属性を表すストリング値。

属性	説明
PayloadType	有効搭載量タイプ。ストリングの値は、none、digest、または full のいずれかです。値は、WebSphere Integration Developer の設定と、ビジネス・オブジェクト (BO) の内容がイベントに書き込まれるかどうかによって異なります。イベントにビジネス・オブジェクトが含まれない場合、値は必ず full に設定されます。

リンク

リンクには、以下のようなオブジェクト固有のイベント内容が含まれています。

属性	説明
processTemplateName	インスタンスが派生したプロセス・テンプレートの名前
processTemplateValidFrom	テンプレートの有効日
flowBpelId	リンクを含むフロー・アクティビティーの wpc:id 属性を表すストリング値
elementName	評価されたリンクの名前
description	リンクの説明。この属性は、プロセス・モデルに指定された場合にのみ含まれます。
PayloadType	完全ストリング

変数

変数には、以下のようなオブジェクト固有のイベント内容が含まれています。

属性	説明
processTemplateName	インスタンスが派生したプロセス・テンプレートの名前。
processTemplateValidFrom	テンプレートの有効日。
variableName	変更された変数の名前。
variableData	WBI モニター互換イベントが要求される場合に出力されます。変数の内容の XML 表現。データ・オブジェクトの各プロパティーは、ネストされた拡張データ・エレメントの形式で報告されます。エレメント・タイプは、「ブール」または「ストリング」タイプで、適切な値を持ちます。
variableData_BO	非 WBI モニター互換イベントが要求される場合に出力されます。このエレメントのタイプは「noValue」で、値には変数の内容の XML 表現が含まれます。データ・オブジェクトの各プロパティーは、ネストされた拡張データ・エレメントの形式で報告されます。
bpelId	アクティビティーの wpc:id 属性を表すストリング値。
PayloadType	有効搭載量タイプ。ストリングの値は、none、digest、または full のいずれかです。値は、WebSphere Integration Developer の設定と、ビジネス・オブジェクトの内容がイベントに書き込まれるかどうかによって異なります。イベントにビジネス・オブジェクトが含まれない場合、値は必ず full に設定されます。

ビジネス・プロセス・イベントの状態

ビジネス・プロセス・イベントは、さまざまな状態で発行可能です。これらの状態のデータについては、状態エレメントで説明されています。

ビジネス・プロセス・イベントには、次の状態エレメントのいずれかが含まれます。

状態名	Common Base Event の内容	
開始	categoryName は StartSituation に設定されます。	
	situationType	
	Type	StartSituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
	situationQualifier	START_COMPLETED
停止	categoryName は StopSituation に設定されます。	
	situationType	
	Type	StopSituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
	situationQualifier	STOP_COMPLETED
破棄	categoryName は DestroySituation に設定されます。	
	situationType	
	Type	DestroySituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
失敗	categoryName は StopSituation に設定されます。	
	situationType	
	Type	StopSituation
	reasoningScope	EXTERNAL
	successDisposition	UNSUCCESSFUL
	situationQualifier	STOP_COMPLETED
レポート	categoryName は ReportSituation に設定されます。	
	situationType	
	Type	ReportSituation
	reasoningScope	EXTERNAL
	reportCategory	STATUS

ビジネス・プロセス・イベント

WebSphere Integration Developer 内のビジネス・プロセス・エレメントのモニターが要求された場合、ビジネス・プロセス・イベントが送信されます。ここでは、ビジネス・プロセスによって発行可能なすべてのイベントのリストを示します。

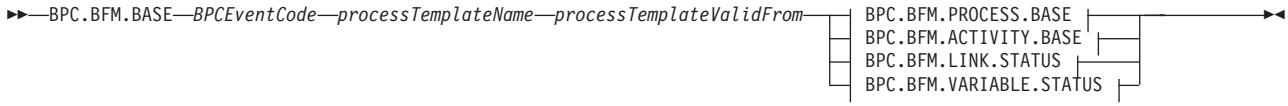
ビジネス・プロセスによって発生するイベントには以下のタイプがあります。

- 354 ページの『プロセス・イベント』

- 356 ページの『アクティビティー・イベント』
- 358 ページの『アクティビティー・スコープ・イベント』
- 359 ページの『リンク・イベント』
- 360 ページの『変数イベント』

XML 構文

ビジネス・プロセス・イベントの有効搭載量の構文は以下のとおりです。



各部の意味は、次のとおりです。

BPCEventCode
: イベント・タイプの番号を識別する Business Process Choreographer イベント・コード。考えられるイベント・コードが次の表にリストされています。

processTemplateName
: プロセス・テンプレートの名前。

processTemplateValidFrom
: プロセス・テンプレートの有効開始日属性。

イベント・エレメントの名前は大文字 (例えば、BPC.BFM.BASE) で、拡張データ・エレメントの名前は大/小文字混合 (例えば、*BPCEventCode*) です。明示されている場合を除き、すべてのデータ・エレメントのタイプは string です。

テーブル列へのキー

次の表の列には、以下の内容が含まれています。

コード イベントの番号。この値は、すべての BPC.BFM.BASE エレメントについて *BPCEventCode* 拡張データ・エレメントとして提供されます。

拡張子名
: Common Base Event の *extensionName* 属性の値として使用されるストリング値。これは、XML 拡張データ・エレメントの名前でもあり、イベントに関する追加データを提供します。

  WebSphere Business Integration Modeler を使用して Business Process Execution Language (BPEL) およびモニター仕様を生成する場合は、拡張名にハッシュ文字 (#) を付け、その後に追加の文字を続けることによって拡張することができます。また、メッセージ・データを発行するイベントには、追加の *extendedDataElements* が含まれています。詳しくは、WebSphere Business Integration Modeler の資料を参照してください。

状態 ビジネス・プロセス・イベントの状態名を指します。状態の詳細については、352 ページの『ビジネス・プロセス・イベントの状態』を参照してください。

イベント性質

WebSphere Integration Developer に表示されるときの、EventNature パラメーター内のビジネス・プロセス・エレメントのイベント状態を指すポインター。

プロセス・イベント

以下の表に、すべてのプロセス・イベントについてまとめます。

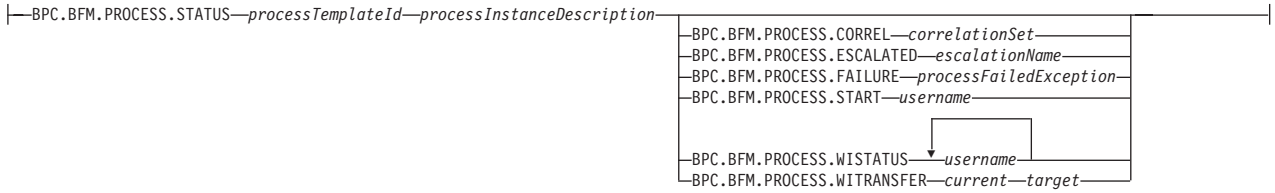
コード	説明	拡張子名	状態	イベント性質
21000	プロセスが開始された	BPC.BFM.PROCESS.START	開始	ENTRY
21001	プロセスが中断された	BPC.BFM.PROCESS.STATUS	レポート	SUSPENDED
21002	プロセスが再開された	BPC.BFM.PROCESS.STATUS	レポート	RESUMED
21004	プロセスが完了した	BPC.BFM.PROCESS.STATUS	停止	EXIT
21005	プロセスが強制終了された	BPC.BFM.PROCESS.STATUS	停止	TERMINATED
21019	プロセスが再始動した	BPC.BFM.PROCESS.START	レポート	RESTARTED
21020	プロセスが削除された	BPC.BFM.PROCESS.STATUS	破棄	DELETED
42001	プロセスが失敗した	BPC.BFM.PROCESS. FAILURE	失敗	FAILED
42003	プロセスが補正中	BPC.BFM.PROCESS.STATUS	レポート	COMPENSATING
42004	プロセスが補正された	BPC.BFM.PROCESS.STATUS	停止	COMPENSATED
42009	プロセスが強制終了中	BPC.BFM.PROCESS.STATUS	レポート	TERMINATING
42010	プロセスが失敗する	BPC.BFM.PROCESS.STATUS	レポート	FAILING
42027	相関セットが初期化された	BPC.BFM.PROCESS.CORREL	レポート	CORRELATION
42041	プロセス作業項目が削除された	BPC.BFM.PROCESS. WISTATUS	レポート	WI_DELETED
42042	プロセス作業項目が作成された	BPC.BFM.PROCESS. WISTATUS	レポート	WI_CREATED
42046	プロセス補正が失敗した	BPC.BFM.PROCESS.STATUS	失敗	COMPFAILED
42047	プロセス・イベントを受信	BPC.BFM.PROCESS.STATUS	レポート	EV_RECEIVED
42049	プロセス・イベントがエスカレートされた	BPC.BFM.PROCESS.ESCALATED	レポート	EV_ESCALATED

コード	説明	拡張子名	状態	イベント性質
42056	プロセス作業項目が転送された	BPC.BFM.PROCESS. WITRANSFER	レポート	WI_TRANSFERRED

プロセス・イベントの有効搭載量の構文は以下のとおりです。

BPC.BFM.PROCESS.BASE

BPC.BFM.PROCESS.STATUS:



各部の意味は、次のとおりです。

processInstanceExecutionState
プロセスの現在の実行状態。形式は次のとおりです。<*state code*>-<*state name*>

processTemplateId
プロセス・テンプレートの ID。

processInstanceDescription
プロセス・インスタンスの説明。

correlationSet
相関セット・インスタンス。形式は次のとおりです。

```
<?xml version="1.0"?>
<correlationSet name="correlation set name">
  <property name="property name"
            value="property value"/>*
</correlationSet>
```

escalationName
エスカレーションの名前。

processFailedException
プロセスの失敗につながる例外メッセージ。

username
BPC.BFM.PROCESS.START の場合、これは、プロセスの開始または再開を要求したユーザーの名前です。 BPC.BFM.PROCESS.WISTATUS の場合、これは、作業項目が作成または削除されたユーザーのリストです。

current
作業項目の現在の所有者のユーザー名。これは、作業項目が転送されたユーザーです。

target 作業項目の新規所有者のユーザー名。

プロセス・イベントの場合、以下のイベント相関範囲 ID も、コンテキスト・データ・エレメントとして Common Base Event に書き込まれます。

- ECSCurrentID は、プロセス・インスタンスの ID です。
- ECSParentID は、現行プロセスのプロセス・インスタンス開始イベントの前の ECSCurrentID の値を提供します。

アクティビティー・イベント

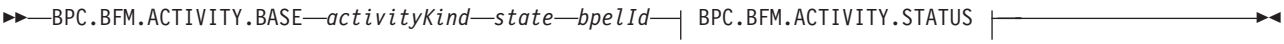
以下の表で、すべてのアクティビティー・イベントについて説明します。

コード	説明	拡張子名	状態	イベント性質
21006	アクティビティーが作動可能	BPC.BFM.ACTIVITY.STATUS	開始	CREATED
21007	アクティビティーが開始した	invoke アクティビティーの場合、BPC.BFM.ACTIVITY. MESSAGE。その他すべてのアクティビティー・タイプの場合、BPC.BFM.ACTIVITY.STATUS	開始	ENTRY
21011	アクティビティーが完了した	invoke、staff、receive、reply の各アクティビティーの場合: BPC.BFM.ACTIVITY. MESSAGE。その他すべてのアクティビティー・タイプの場合、BPC.BFM.ACTIVITY.STATUS	停止	EXIT
21021	要求がキャンセルされた	BPC.BFM.ACTIVITY.STATUS	レポート	DEASSIGNED
21022	アクティビティーが要求された	BPC.BFM.ACTIVITY.CLAIM	レポート	ASSIGNED
21027	アクティビティーが強制終了された	BPC.BFM.ACTIVITY.STATUS	停止	TERMINATED
21080	アクティビティーが失敗した	BPC.BFM.ACTIVITY.FAILURE	失敗	FAILED
21081	アクティビティーの期限切れ	BPC.BFM.ACTIVITY.STATUS	レポート	EXPIRED
42005	アクティビティーがスキップされた	BPC.BFM.ACTIVITY.STATUS	レポート	SKIPPED
42012	アクティビティー出力メッセージが設定された	BPC.BFM.ACTIVITY.MESSAGE	レポート	OUTPUTSET
42013	アクティビティー障害メッセージが設定された	BPC.BFM.ACTIVITY.MESSAGE	レポート	FAULTSET
42015	アクティビティーが停止した	BPC.BFM.ACTIVITY.STATUS	停止	STOPPED
42031	アクティビティーが強制再試行された	BPC.BFM.ACTIVITY.STATUS	レポート	FRETRIED
42032	アクティビティーが強制完了した	BPC.BFM.ACTIVITY.STATUS	停止	FCOMPLETED

コード	説明	拡張子名	状態	イベント性質
42036	アクティビティーがメッセージを受信	BPC.BFM.ACTIVITY.MESSAGE	レポート	EXIT
42037	ループ条件が true	BPC.BFM.ACTIVITY.STATUS	レポート	CONDTRUE
42038	ループ条件が false	BPC.BFM.ACTIVITY.STATUS	レポート	CONDFALSE
42039	作業項目が削除された	BPC.BFM.ACTIVITY. WISTATUS	レポート	WI_DELETED
42040	作業項目が作成された	BPC.BFM.ACTIVITY. WISTATUS	レポート	WI_CREATED
42050	アクティビティーがエスカレートされた	BPC.BFM.ACTIVITY.ESCALATED	レポート	ESCALATED
42054	アクティビティー作業項目が更新された	BPC.BFM.ACTIVITY. WISTATUS	レポート	WI_REFRESHED
42055	作業項目が転送された	BPC.BFM.ACTIVITY. WITRANSFER	レポート	WI_TRANSFERRED
42057	For each - アクティビティー分岐が開始された	BPC.BFM.ACTIVITY. FOREACH	レポート	BRANCHES_STARTED

アクティビティー・イベントの有効搭載量の構文は以下のとおりです。

BPC.BFM.ACTIVITY.BASE



BPC.BFM.ACTIVITY.STATUS:

各部の意味は、次のとおりです。

activityKind
アクティビティーの種類 (例えば、sequence や invoke)。形式は、*<kind code>-<kind name>* です。

state アクティビティー・インスタンスの現在の状態。形式は次のとおりです。*<state code>-<state name>*

bpelId BPEL ファイル内のアクティビティーの wpc:id 属性。これは、プロセス・モデル内部のアクティビティーに固有です。

activityTemplateName
アクティビティー・テンプレートの名前。

activityTemplateId
アクティビティー・テンプレートの内部 ID。

activityInstanceDescription
アクティビティー・インスタンスの説明。

username
BPC.BFM.ACTIVITY.CLAIM の場合、これはタスクが要求されたユーザーです。 BPC.BFM.ACTIVITY.WISTATUS の場合、これは作業項目に関連付けられたリスト・ユーザーです。

principal
アクティビティーを要求したユーザーの名前。

escalationName
エスカレーションの名前。

activityFailedException
アクティビティーが失敗する原因となった例外

parallelBranchesStarted
開始された分岐の数。

message **または** *message_BO*
ストリングまたはビジネス・オブジェクト (BO) 表現としてのサービスの入力または出力メッセージ。形式は、WebSphere Integration Developer の「**イベント・モニター (Event Monitor)**」タブで「**互換性のあるイベントのモニター (Monitor Compatible Events)**」オプションが選択されたかどうかによって異なります。

current
作業項目の現在の所有者のユーザー名。これは、作業項目が転送されたユーザーです。

target 作業項目の新規所有者のユーザー名。

アクティビティー・イベントの場合、以下のイベント相関範囲 ID も、コンテキスト・データ・エレメントとして Common Base Event に書き込まれます。

- *ECSCurrentID* は、アクティビティーの ID です。
- *ECSParentID* は、収容プロセスの ID です。

アクティビティー・スコープ・イベント

以下の表で、すべてのアクティビティー・スコープ・イベントについて説明します。

コード	説明	拡張子名	状態	イベント性質
42020	スコープが開始した	BPC.BFM.ACTIVITY.STATUS	開始	ENTRY
42021	スコープがスキップされた	BPC.BFM.ACTIVITY.STATUS	レポート	SKIPPED
42022	スコープが失敗した	BPC.BFM.ACTIVITY.FAILURE	失敗	FAILED

コード	説明	拡張子名	状態	イベント性質
42023	スコープが強制終了中	BPC.BFM.ACTIVITY.STATUS	レポート	FAILING
42024	スコープが強制終了した	BPC.BFM.ACTIVITY.STATUS	停止	TERMINATED
42026	スコープが完了した	BPC.BFM.ACTIVITY.STATUS	停止	EXIT
42043	スコープが補正中	BPC.BFM.ACTIVITY.STATUS	レポート	COMPENSATING
42044	スコープが補正された	BPC.BFM.ACTIVITY.STATUS	停止	COMPENSATED
42045	スコープ補正が失敗した	BPC.BFM.ACTIVITY.STATUS	失敗	COMPFAILED
42048	スコープ・イベントを受信	BPC.BFM.ACTIVITY.STATUS	レポート	EV_RECEIVED
42051	スコープ・イベントがエスカレートされた	BPC.BFM.ACTIVITY.ESCALATED	レポート	EV_ESCALATED

アクティビティー・スコープ・イベントは、アクティビティー・イベントのタイプで、その構文については、上述の BPC.BFM.ACTIVITY.STATUS で説明されています。

アクティビティー・スコープ・イベントの場合、以下のイベント相関範囲 ID も、コンテキスト・データ・エレメントとして Common Base Event に書き込まれます。

- ECSCurrentID は、スコープの ID です。
- ECSParentID は、収容プロセスの ID です。

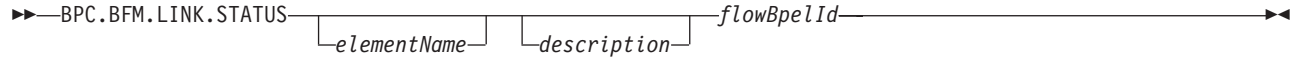
リンク・イベント

以下の表は、すべてのリンク・イベントについて説明しています。

コード	説明	拡張子名	状態	イベント性質
21034	リンクが true と評価された	BPC.BFM.LINK.STATUS	レポート	CONDTRUE
42000	リンクが false と評価された	BPC.BFM.LINK.STATUS	レポート	CONDFALSE

リンク・イベントの有効搭載量の構文は以下のとおりです。

BPC.BFM.LINK.STATUS



各部の意味は、次のとおりです。

elementName
リンクの名前。

description
リンクの説明。

flowBpelId
リンクが定義されているフロー・アクティビティーの ID。

リンク・イベントの場合、以下のイベント相関範囲 ID も、コンテキスト・データ・エレメントとして Common Base Event に書き込まれます。

- ECSCurrentID は、リンクのソース・アクティビティーの ID です。
- ECSParentID は、収容プロセスの ID です。

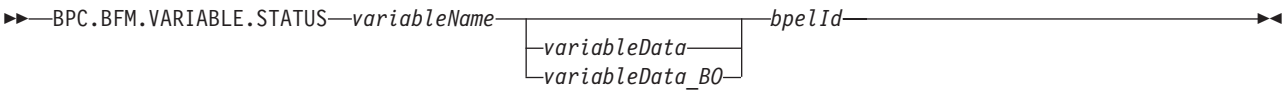
変数イベント

以下の表で、変数イベントについて説明します。

コード	説明	拡張子名	状態	イベント性質
21090	変数の更新	BPC.BFM.VARIABLE.STATUS	レポート	CHANGED

変数イベントの有効搭載量の構文は以下のとおりです。

BPC.BFM.VARIABLE.STATUS



各部の意味は、次のとおりです。

variableName
変数の名前。

variableData **または** *variableData_BO*
変数 *variableName* が初期化されていない場合、 *variableData* も *VariableData_BO* エレメントも存在しません。ストリングまたはビジネス・オブジェクト (BO) 表現としての変数のデータ内容。形式は、WebSphere Integration Developer の「**イベント・モニター (Event Monitor)**」タブで「**互換性のあるイベントのモニター (Monitor Compatible Events)**」オプションが選択されたかどうかによって異なります。

bpelId 変数の Business Process Choreographer ID。

変数イベントの場合、以下のイベント相関範囲 ID が、コンテキスト・データ・エレメントとして Common Base Event に書き込まれます。

- ECSCurrentID は、収容スコープまたは収容プロセスの ID です。
- ECSParentID は、現行プロセスのプロセス・インスタンス開始イベントの前の ECSCurrentID です。

ヒューマン・タスク・イベント

ヒューマン・タスクのために発行されるイベントは、状態非依存データとヒューマン・タスク・イベントに固有のデータで構成されています。ここでは、ヒューマン・タスク・イベントに固有の属性とエレメントについて説明します。

ヒューマン・タスク・イベントには、次に示すイベント内容のカテゴリーがあります。

ヒューマン・タスク固有のイベント・データ

タスクおよびエスカレーションのためにイベントが作成されます。これらの各イベント・タイプのオブジェクト固有の内容について説明します。

タスク

特に明記されていない限り、内容は、タイプがストリングの extendedDataElements として書き込まれます。

タスク・イベントには、以下のようなオブジェクト固有のイベント内容が含まれています。

属性	説明
taskTemplateName	インスタンスが派生したタスク・テンプレートの名前。
taskTemplateValidFrom	テンプレートの有効日。
taskTemplateId	インスタンスが派生したタスク・テンプレートの ID。
taskInstanceDescription	オプション: タスク・インスタンスの説明。
PayloadType	有効搭載量タイプ。ストリングの値は、none、digest、または full のいずれかです。値は、WebSphere Integration Developer の設定と、ビジネス・オブジェクトの内容がイベントに書き込まれるかどうかによって異なります。イベントにビジネス・オブジェクトが含まれない場合、値は必ず full に設定されます。

エスカレーション

エスカレーションには、以下のようなオブジェクト固有のイベント内容が含まれています。

属性	説明
taskTemplateName	インスタンスが派生したタスク・テンプレートの名前。
taskTemplateValidFrom	テンプレートの有効日。
taskTemplateId	インスタンスが派生したタスク・テンプレートの ID。
escalationName	エスカレーションの名前。
escalationInstanceDescription	オプション: エスカレーション・インスタンスの説明。
PayloadType	有効搭載量タイプ。ストリングの値は、none、digest、または full のいずれかです。値は、WebSphere Integration Developer の設定と、ビジネス・オブジェクトの内容がイベントに書き込まれるかどうかによって異なります。イベントにビジネス・オブジェクトが含まれない場合、値は必ず full に設定されます。

ヒューマン・タスク・イベントの状態

ヒューマン・タスク・イベントは、さまざまな状態で発行可能です。これらの状態のデータについては、状態エレメントで説明されています。

ヒューマン・タスク・イベントには、次の状態エレメントのいずれかが含まれます。

状態名	Common Base Event の内容	
開始	categoryName は StartSituation に設定されます。	
	situationType	
	Type	StartSituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
	situationQualifier	START_COMPLETED
停止	categoryName は StopSituation に設定されます。	
	situationType	
	Type	StopSituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
	situationQualifier	STOP_COMPLETED
破棄	categoryName は DestroySituation に設定されます。	
	situationType	
	Type	DestroySituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
失敗	categoryName は StopSituation に設定されます。	
	situationType	
	Type	StopSituation
	reasoningScope	EXTERNAL
	successDisposition	UNSUCCESSFUL
	situationQualifier	STOP_COMPLETED
レポート	categoryName は ReportSituation に設定されます。	
	situationType	
	Type	ReportSituation
	reasoningScope	EXTERNAL
	reportCategory	STATUS

ヒューマン・タスク・イベント

WebSphere Integration Developer 内のタスクのエレメントについてモニターが要求された場合、ヒューマン・タスク・イベントが送信されます。ここでは、ヒューマン・タスクによって発行可能なすべてのイベントのリストを示します。

ヒューマン・タスクによって発生するイベントには以下のタイプがあります。

- 364 ページの『タスク・イベント』
- 366 ページの『エスカレーション・イベント』

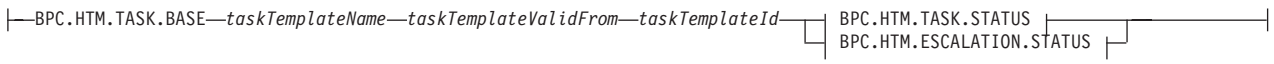
注: タスク・モデルにおいてビジネス関連性フラグが true に設定されている場合、イベントは随時タスクの場合のみ発行されます。

XML 構文

ヒューマン・タスク・イベントの有効搭載量の構文は以下のとおりです。

BPC.HTM.BASE

BPC.HTM.TASK.BASE:



各部の意味は、次のとおりです。

HTMEventCode
: イベント・タイプの番号を識別する Business Process Choreographer イベント・コード。考えられるイベント・コードが次の表にリストされています。

taskTemplateName
: タスク・テンプレートの名前。

taskTemplateValidFrom
: タスク・テンプレートが有効になる日時。

taskTemplateId
: テンプレートの ID。

イベント・エレメントの名前は大文字 (例えば、BPC.HTM.BASE) で、拡張データ・エレメントの名前は大/小文字混合 (例えば、*HTMEventCode*) です。明示されている場合を除き、すべてのデータ・エレメントのタイプは string です。

テーブル列へのキー

コード の欄には、イベントの番号が記載されています。値は、*HTMEventCode* という名前の拡張データ・エレメントとして Common Base Event に書き込まれます。各欄には以下のような内容が記載されています。

拡張子名
: Common Base Event の extensionName 属性の値として使用されるストリング値。

  WebSphere Business Integration Modeler を使用して基本のタスク・モデルを作成する場合、有効搭載量にメッセージ・データを含むイベントの拡張名は、ハッシュ文字 (#) とそれに続く追加文字によって拡張できます。これらの追加文字は、いろいろなメッセージ・オブジェクトを運ぶ Common Base Events の識別に使用されます。メッセージ・データを出力するイベントにも、データ・オブジェクトの内容を報告するために、追加のネストされた extendedDataElements が含まれます。詳しくは、WebSphere Business Integration Modeler の資料を参照してください。

状態
: ヒューマン・タスク・イベントの状態名を指します。状態の詳細については、361 ページの『ヒューマン・タスク・イベントの状態』を参照してください。

イベント性質

WebSphere Integration Developer に表示されるときの、EventNature パラメーター内のビジネス・プロセス・エレメントのイベント状態を指すポインター。

タスク・イベント

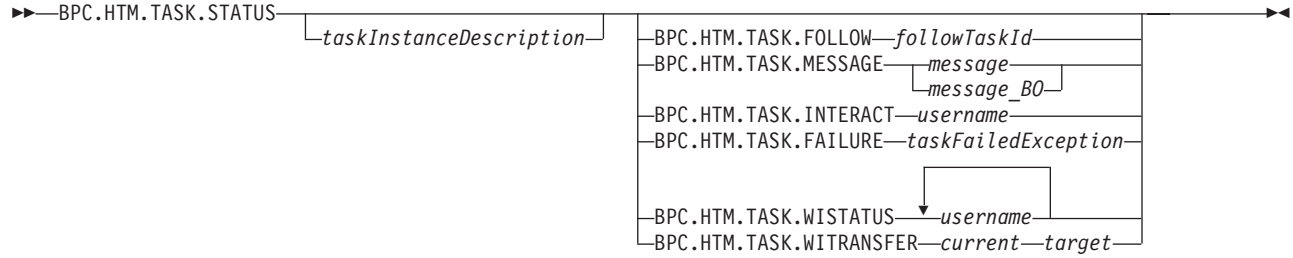
以下の表で、すべてのタスク・イベントについて説明します。

コード	説明	拡張子名	状態	イベント性質
51001	タスクが作成された	BPC.HTM.TASK. INTERACT	レポート	CREATED
51002	タスクが削除された	BPC.HTM.TASK.STATUS	破棄	DELETED
51003	タスクが開始された	BPC.HTM.TASK.STATUS	開始	ENTRY
51004	タスクが完了した	BPC.HTM.TASK.STATUS	停止	EXIT
51005	要求がキャンセルされた	BPC.HTM.TASK.STATUS	レポート	DEASSIGNED
51006	タスクが要求された	BPC.HTM.TASK. INTERACT	レポート	ASSIGNED
51007	タスクが強制終了された	BPC.HTM.TASK.STATUS	停止	TERMINATED
51008	タスクが失敗した	BPC.HTM.TASK. FAILURE	失敗	FAILED
51009	タスクの期限切れ	BPC.HTM.TASK.STATUS	レポート	EXPIRED
51010	サブタスクを待機中	BPC.HTM.TASK.STATUS	レポート	WAITFORSUBTASK
51011	サブタスクが完了した	BPC.HTM.TASK.STATUS	停止	SUBTASKCOMPLETED
51012	タスクが再始動した	BPC.HTM.TASK.STATUS	レポート	RESTARTED
51013	タスクが一時停止した	BPC.HTM.TASK.STATUS	レポート	SUSPENDED
51014	タスクが再開した	BPC.HTM.TASK.STATUS	レポート	RESUMED
51015	タスクが完了し、追加タスクが開始した	BPC.HTM.TASK. FOLLOW	レポート	COMPLETEDFOLLOW
51101	タスク・プロパティーが更新された	BPC.HTM.TASK.STATUS	レポート	UPDATED
51103	出力メッセージが更新された	BPC.HTM.TASK. MESSAGE	レポート	OUTPUTSET
51104	障害メッセージが更新された	BPC.HTM.TASK. MESSAGE	レポート	FAULTSET
51201	作業項目が削除された	BPC.HTM.TASK. WISTATUS	破棄	WI_DELETED
51202	作業項目が作成された	BPC.HTM.TASK. WISTATUS	レポート	WI_CREATED

コード	説明	拡張子名	状態	イベント性質
51204	作業項目が転送された	BPC.HTM.TASK. WITRANSFER	レポート	WI_TRANSFERRED
51205	作業項目が更新された	BPC.HTM.TASK. WISTATUS	レポート	WI_REFRESHED

タスク・イベントの有効搭載量の構文は以下のとおりです。

BPC.HTM.TASK.STATUS



各部の意味は、次のとおりです。

taskInstanceDescription
タスクの説明。

followTaskId
追加タスクとして開始されたタスクの ID

message **または** *message_BO*
入力または出力メッセージを含むストリングまたはビジネス・オブジェクトの表現。形式は、WebSphere Integration Developer の「**イベント・モニター (Event Monitor)**」タブで「**互換性のあるイベントのモニター (Monitor Compatible Events)**」オプションが選択されたかどうかによって異なります。

taskFailedException
セミコロン (;) で区切られた *faultNameSpace* および *faultName* を含むストリング。

username
BPC.HTM.TASK.INTERACT の場合、これは、タスクに関連したユーザーの名前です。 BPC.BPC.TASK.WISTATUS の場合、これは、作業項目が作成または削除されたユーザーのリストです。

current
現行ユーザーの名前。これは、作業項目が転送されたユーザーです。

target 作業項目の受け取り側のユーザー名。

タスク・イベントの場合、以下のイベント相関範囲の ID が、コンテキスト・データ・エレメントとして Common Base Event に書き込まれます。

- ESCcurrentID は、タスク・インスタンスの ID です。
- ECSParentID は、タスク・インスタンス・イベントの前の ECSCurrentID です。

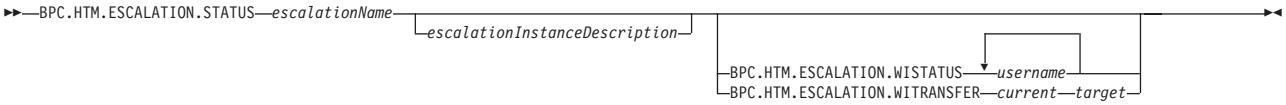
エスカレーション・イベント

以下の表で、すべてのタスク・エスカレーション・イベントについて説明します。

コード	説明	拡張子名	状態
53001	エスカレーションが起動された	BPC.HTM.ESCALATION. STATUS	レポート
53201	作業項目が削除された	BPC.HTM.ESCALATION. WISTATUS	破棄
53202	作業項目が作成された	BPC.HTM.ESCALATION. WISTATUS	レポート
53204	エスカレーションが転送された	BPC.HTM.ESCALATION. WITRANSFER	レポート
53205	作業項目が更新された	BPC.HTM.ESCALATION. WISTATUS	レポート

エスカレーション・イベントの有効搭載量の構文は以下のとおりです。

BPC.HTM.ESCALATION.STATUS



各部の意味は、次のとおりです。

EscalationName
エスカレーションの名前。

escalationInstanceDescription
エスカレーションの説明。

username
これは、作業項目がエスカレートされたユーザーのリストです。

current
現行ユーザーの名前。これは、作業項目が転送されたユーザーです。

target 作業項目の受け取り側のユーザー名。

タスク・イベントの場合、以下のイベント相関範囲の ID が、コンテキスト・データ・エレメントとして Common Base Event に書き込まれます。

- ESCcurrentID は、エスカレーションの ID です。
- ECSParentID は、関連したタスク・インスタンスの ID です。

第 8 章 ビジネス・プロセスのチューニング

このタスクを使用して、ビジネス・プロセスのパフォーマンスを改善します。

ビジネス・プロセスが正常に実行されたら、このタスクを実行してパフォーマンスを改善することができます。

1. 基本的パフォーマンスの測定方法、および最適化する測定値を定義します。

   例えば、ビジネス・アプリケーションによっては、負荷のピーク時にエンド・ユーザーの応答時間を減らす方法を好むものもあります。その他のアプリケーションの場合は、システムで可能なトランザクションの処理速度のほうが、各トランザクションの実際の期間よりも重要になります。

2. 基本的な測定を行います。

   アプリケーションのチューニングに適した負荷、時刻、および曜日の条件下で、基本的な測定を行います。通常、最も重要な基本的な測定値は、スループットおよび応答時間です。例えば CPU 負荷が 100% に、ディスク I/O が最大値に、またはネットワーク I/O が 100% に達するなど、特定のボトルネック容量に達した後で、スループット値が測定されます。信頼性の高い応答時間の値は、単一のプロセス・インスタンスで、サーバー使用率が低いときに、最も適切に測定されます。

3. プロセスを調整します。

   アプリケーションが長期間にわたって実行するプロセスと microflow のどちらを使用するかに応じて、以下のステップを実行します。

   - 長期間にわたって実行するプロセスを調整するには、368 ページの『長期間にわたって実行するプロセスのチューニング』で説明されているステップを実行します。これらのプロセスは、長時間実行が継続する傾向がありますが、イベントまたは人との対話で中断される場合もあります。したがってそのパフォーマンスは、Business Process Choreographer データベースとメッセージング・サービスのパフォーマンスによって異なります。
   - microflow を調整するには、377 ページの『microflow のチューニング』で説明されているステップを実行します。このプロセスは、どちらかといえば短期間のみ実行するプロセスです。このプロセスは、監査ロギング用のみにデータベースを使用して (監査ロギングが使用可能な場合)、テンプレート情報を取得します。このプロセスでは、永続データの保管のためにメッセージング・サポートが使用されません。このプロセスは、人との対話を必要としません。

4. アプリケーションを調整します。

   さまざまなオプションを使用して、1 つのアプリケーションにおいて同じ機能を実現することができます。パフォーマンスが重要なコードすべてを識別して検討します。非同期性を最大化し、アクションが不要にシリアライズされることのないようにします。シリアライズとシリアライズ解除のコストはプロセスで使用されたデータ・オブジェクトのサイズおよび複雑度に直接関連するので、プロセスにサブミットされるデータのサイズおよび複雑度を最小限にしてみてください。

エラー状態にならない範囲でタイムアウトの短縮について考慮します。データベース照会の結果をキャッシュに入れる機会を識別します。

5. 除去可能なパフォーマンスのボトルネックがないかどうか、現在の構成を検討します。

   考慮するべき可能性のある項目は以下のとおりです。

   - より多くのプロセッサー、メモリー、およびより高速なディスクをインストールします。
   - データベースのログをデータとは異なる物理ディスクに保管して、データをいくつかのディスクに分散させます。
   - 最適なパフォーマンスを得るために、Cloudscape ではなく、DB2を使用します。

6. 基本的な測定項目に関して、同様の負荷条件でベンチマーク測定を繰り返します。

   アプリケーションのパフォーマンス測定の永続レコードを保持して、パフォーマンスにおける将来の変更点を客観的に測定します。

ビジネス・プロセスは、はっきりと分かるほど高速に実行されるように構成されます。

長期間にわたって実行するプロセスのチューニング

このタスクを使用して、長期間にわたって実行するビジネス・プロセスのパフォーマンスを改善します。

長期間にわたって実行するプロセスには、ユーザーとの対話、非同期呼び出し、複数の受信、ピック、イベント・ハンドラーなどがあります。これらのプロセスでは、永続状態を保管するためにデータベースとメッセージング・サブシステムを使用します。以下のトピックでは、長期間にわたって実行するプロセスのパフォーマンスを向上させる方法を説明します。

初期データベース設定の指定

このタスクを使用して、初期 DB2 データベース設定を指定します。この情報は、例を示す目的で提供しています。

重要: 以下に示すのは、Business Process Choreographer データベースに関する情報です。WebSphere のデフォルトのメッセージング・データベースのチューニングについて詳しくは、WebSphere Application Server for z/OS インフォメーション・センターの『メッセージング・エンジンのデータ・ストアのチューニングと問題解決』を参照してください。

WebSphere Process Server for z/OS でのデータベース作成について詳しくは、「Installing and configuring WebSphere Process Server for z/OS」の PDF で、『*データベースを作成する際の考慮事項*』および『*データベースおよびストレージ・グループの作成*』を参照してください。

データベースを良好に運用するには、データベースの初期設定値を指定します。設定値の細密チューニングは、後述の 375 ページの『データベースの細密チューニング』で行います。

1. ログ・ファイルをデータ・ファイルから分離します。

   データから分離されたディスク・ドライブにデータベースのログ・ファイルを配置すると、十分な数のディスク・ドライブが使用可能であれば、パフォーマンスが向上する傾向があります。使用可能なディスク・ドライブがほとんどない場合は、前のセクションで説明したようにテーブル・スペースを分散するほうが、データベースのログ・ファイルを別々のドライブに配置するよりも一般に利点があります。

   例えば、Windows システムで DB2 を使用する場合は、次のコマンドを入力することによって、BPEDB という名前のデータベースのログ・ファイルの場所を F:¥db2logs ディレクトリーに変更することができます。

   ```
   db2 UPDATE DB CFG FOR BPEDB USING NEWLOGPATH F:¥db2logs
   ```

2. テーブル・スペースを作成します。

   データベースを作成したら、テーブル・スペースを明示的に作成します。テーブル・スペースを作成するためのサンプル・スクリプトは、Business Process Choreographer により、WebSphere Application Serverインストール・サブディレクトリーの ProcessChoreographer に準備されています。それらのスクリプトをカスタマイズして、特定のシナリオのニーズに対処します。テーブル・スペースを作成する場合の目標は、DB2 で使用可能なできるだけ多くのディスク・ドライブ全体に入出力操作を分散させることです。デフォルトでは、これらのスクリプトにより次のテーブル・スペースが作成されます。

   AUDITLOG
   プロセスおよびタスクの監査証跡テーブルを格納します。使用される監査の程度に応じて、このテーブル・スペース内のテーブルへのアクセス数が多くなる可能性があります。監査をオフにすると、このテーブル・スペース内のテーブルにアクセスできなくなります。

   COMP
   Business Process Choreographer バージョン 5 のビジネス・プロセス用の補正テーブルを格納します。補正可能なプロセスとアクティビティーの割合によっては、このテーブル・スペース内のテーブルでディスク高帯域幅が必要になる場合があります。ビジネス・プロセス内で補正が使用されない場合は、このテーブル・スペース内のテーブルは使用されません。

   INSTANCE
   プロセス・インスタンスおよびタスク・テーブルを格納します。実行される長期実行プロセスの種類に関係なく、常時集中的に使用されます。可能であれば、このテーブル・スペースを数台のディスク・ドライブに広げます。

   SCHEDTS
   WebSphere スケジューリング・コンポーネントが使用するテーブルを格

納します。スケジューラー・テーブル・スペース内のテーブルへのアクセスは、スケジューラーで使用されるキャッシング機構のために通常低速で行われます。

STAFFQRY
Lightweight Directory Access Protocol (LDAP) のような社員レジストリーから入手される社員照会結果を一時的に保管するために使用するテーブルを含みます。ビジネス・プロセスに数多くの person アクティビティーが含まれている場合は、このテーブル・スペース内のテーブルへのアクセスが頻繁に行われます。

TEMPLATE
プロセスおよびタスクのテンプレート情報を保管するテーブルを格納します。アプリケーションのデプロイメント中に、テーブルにデータが取り込まれます。実行時には、アクセス速度が低下します。デプロイメント中は、データは更新されず、新規データのみが挿入されます。

WORKITEM
作業項目の処理に必要なテーブルを保持します。作業項目は、ヒューマン・タスクの対話に使用されます。ビジネス・プロセス内のヒューマン・タスクの数に応じて、このテーブル・スペースのテーブルへのアクセス速度が、低速からかなりの高速まで変わる場合があります。使用されている明示的なヒューマン・タスクがない場合でも、アクセス速度はゼロになりません。これは、長期実行プロセスの管理をサポートするために、作業項目も生成されるからです。

ハイパフォーマンスを生み出すデータベースを作成するには、以下のアクションを実行します。

a. データベースを作成します。
b. 希望するディスクにテーブル・スペースを作成します。

例えば、以下のスクリプトは、WebSphere Application Server インストール・サブディレクトリーの `ProcessChoreographer` にある `createTablespaceDb2.ddl` ファイルを基にしています。Windows システムで異なる 3 つのディスク・ドライブを使用してテーブル・スペースを作成します。

```
-- Scriptfile to create tablespaces for DB2 UDB
-- Replace occurence or @location@ in this file with the location
-- where you want the tablespace containers to be stored, then run:
-- db2 connect to BPEDB
-- db2 -tf createTablespaceDb2.ddl


CREATE TABLESPACE TEMPLATE MANAGED BY SYSTEM USING( 'D:/BPE/TEMPLATE' );

CREATE TABLESPACE STAFFQRY MANAGED BY SYSTEM USING( 'D:/BPE/STAFFQRY' );

CREATE TABLESPACE AUDITLOG MANAGED BY SYSTEM USING( 'E:/BPE/AUDITLOG' );

CREATE TABLESPACE COMP MANAGED BY SYSTEM USING( 'D:/BPE/COMP' );

CREATE TABLESPACE INSTANCE MANAGED BY SYSTEM USING( 'D:/BPE/INSTANCE', 'E:/BPE/INSTANCE');
```

```
CREATE TABLESPACE WORKITEM MANAGED BY SYSTEM USING( 'F:/BPE/WORKITEM' );

CREATE TABLESPACE SCHEDTS MANAGED BY SYSTEM USING( ' F:/BPE/SCHEDTS' );
```

INSTANCE テーブル・スペースが 2 つのドライブに分散されることに注意してください。

c. テーブルを作成します。

個々のデータベース用に提供されたスクリプトを実行して、Business Process Choreographer テーブルを作成します。例えば DB2 の場合、`ProcessChoreographer` ディレクトリーの `createSchemaDb2.ddl` ファイルを使用します。

3. データベースを調整します。

キャパシティー・プランニング・ツールを使用して、データベースの初期設定を行います。

DB2 を使用している場合は、Business Process Choreographer データベースのポップアップ・メニューから 「**DB2 configuration advisor**」 を選択して、DB2 コントロール・センターから DB2 Configuration Advisor を始動します。以下のアクションを実行します。

a. メモリーを DB2 割り振ります。

「**サーバー**」では、スワッピングしないで物理的に使用可能な最大限のメモリーのみを DB2 に割り振ります。

b. ワークロードのタイプを指定します。

「**ワークロード (Workload)**」では、「**混合 (Mixed)**」 (照会とトランザクション) を選択します。

c. 「**トランザクション**」では、トランザクションの長さと、毎分処理されるトランザクションの推定数を指定します。

「**10 より大きい (More than 10)**」を選択し、長いトランザクションが使用されることを示します。

次に、「**毎分のトランザクション (Transactions per minute)**」フィールドで、毎分処理されるトランザクションの推定数を選択します。この数値を判別するには、プロセスの各アクティビティーにトランザクションが 1 つあると想定します。このとき、1 分間に実行されるトランザクション数は次のようになります。

*毎分実行されるトランザクション数 = 毎分完了するプロセス数 * 各プロセス内のアクティビティー数*

d. データベースを調整して、トランザクション・パフォーマンスがより高速になり、リカバリーが遅くなるようにします。

「**優先順位**」では、「**高速トランザクション・パフォーマンス (Faster transaction performance)**」を選択します。

e. 可能な場合、実動中の標準的な量のデータを取り込んだデータベースを調整します。 「**データ取り込み済み (Populated)**」では、「**はい**」を選択します。それ以外の場合は、「**いいえ**」を選択します。

f. 並列接続設定を調整します。

「**接続**」では、アプリケーション・サーバーに対して確立可能な並列接続の最大数を指定します。この値を決定する場合は、以下の点を指針にしてください。

- 必要なデータベース接続数は、WebSphere Application Server への JDBC (Java DataBase Connectivity) の数によって決まります。JDBC 接続は、WebSphere Application Server にある JDBC 接続プールによって準備されます。*p* JDBC 接続では、*p* * 1.1 本のデータベース接続が必要です。*p* の現実的な値の見積もり方法については、374 ページの『アプリケーション・サーバーのチューニング』で説明します。
- Business Process Choreographer とデータベースが同じ物理サーバーにインストールされている場合、Business Process Choreographer ではリモート・データベース接続が不要です。ただし、リモート・データベース管理でリモート接続が必要になる可能性があるので、ゼロではなく低い値を指定します。
- Business Process Choreographer と DB2 が別々のサーバーにインストールされている場合は、ローカル接続に関して前に説明した規則に従って、リモート・アプリケーションの数を設定します。

g. 「**分離 (Isolation)**」では、「**読み取り固定**」を選択します。 Business Process Choreographer では、この分離レベルが必要です。

構成アドバイザーに提案される変更点が表示されます。すぐに変更点を適用することも、ファイルに保管して後で適用することもできます。

ユーザーの長期間にわたって実行するプロセスは、現在の環境とロード条件において可能な限り高速実行されます。

メッセージング・エンジンの設定計画

メッセージング・エンジンの初期設定の計画を立てるには、このタスクを使用します。

長期間にわたって実行するプロセスの最善のパフォーマンスを引き出すには、永続メッセージのパフォーマンスが最大化するようにメッセージ・キューイング・システムを調整します。

WebSphere Platform Messaging を使用する場合は、WebSphere Application Server for z/OS インフォメーション・センターの『サービス統合』の説明に従って、メッセージング・エンジン用のデータ・ストアをセットアップおよび調整してください。

デフォルトのメッセージング・サービスではなく、IBM WebSphere MQ メッセージング製品を使用する場合は、以下の手順を最後まで実行してください。

1. MQ パラメーター設定を調整します。

以下の MQ パラメーター設定を調整します。

- ログ・ファイル・ページ
- ログ・バッファー・ページ
- ログ 1 次ファイル
- ログ 2 次ファイル
- ログのデフォルト・パス
- 最大チャネル数
- チャネル・アプリケーションのバインド・タイプ

永続キュー・データと MQ ログの両方のデフォルト・ロケーションは、MQ インストール・ディレクトリーです。永続キューと WebSphere MQ ログのデータ・ストレージは、別々のディスク・ドライブに配置します。別のディスク・ドライブを参照するログ・ファイルへのパスを変更することにより、MQ ログの場所を変更できます。これらの変更は、Business Process Choreographer のキュー・マネージャーを作成する前に行います。

2. Business Process Choreographer の WebSphere MQ サービス・プロパティーを調整します。

   これらの値は、Business Process Choreographer が使用するキュー・マネージャーを作成する前に設定する必要があります。各パラメーターの値は、以下の表に示す最大値に設定します。

表 39. Business Process Choreographer の WebSphere MQ サービス・プロパティーのチューニング

パラメーター	値	コメント

ログ・ファイル・ページ	16384	Windows システムの場合、WebSphere MQ のバージョンによっては、MQ 管理ツールを使用してログ・ファイル・ページの数を 16384 に設定することをサポートしないものもあります。この場合は、Windows レジストリー・キー、
		`HKEY_LOCAL_MACHINE¥SOFTWARE¥IBM¥MQSeries¥ CurrentVersion¥Configuration¥LogDefaults`
		の値を、次のように変更します。
		`16384`

ログ 1 次ファイル	10	
ログ 2 次ファイル	53	
ログ・バッファー・ページ	512	

3. キュー・マネージャー・プロパティーを調整します。

   以下の表に示すように、チャネル数が最大の場合のキュー・マネージャー・プロパティー、およびチャネル・アプリケーションのバインディング・タイプを指定します。

表 40. キュー・マネージャーのプロパティーのチューニング

キュー・マネージャーのプロパティー	値
最大チャネル数	デフォルトを使用
チャネル・アプリケーションのバインド・タイプ	FASTPATH

ご使用のキュー・マネージャーは最適な状態で作動しています。

アプリケーション・サーバーのチューニング

このタスクを使用して、アプリケーション・サーバーを調整します。

このタスクを始める前に、 368 ページの『初期データベース設定の指定』で説明されるとおりに、データベースの初期設定値を指定する必要があります。

ビジネス・プロセス・コンテナーが確実に最適な状態で動作できるようにするため、サーバー設定を調整する必要があります。

1. ビジネス・プロセス・コンテナーごとに必要なアプリケーション・サーバーのリソースを見積もります。
   a. ビジネス・プロセスの状態情報をデータベースに読み書きするための 1 つのデータ・ソース: BPEDDataSourceDb2 (サーバー・スコープ DB2 Universal JDBC Driver Provider (XA))
   b. 以下を追加して、プロセス・ナビゲーションの最大並行トランザクション数 *t* を計算します。
      - Business Process Choreographer API を通じて並行して接続されるクライアントの最大数
      - JMS 活動化仕様 BPEInternalActivationSpec で定義される並行エンドポイント数
      - JMS 活動化仕様 HTMInternalActivationSpec で定義される並行エンドポイント数

      注: プロセス・サーバーの活動化仕様を表示するには、管理コンソールの **「リソース」** → **「JMS プロバイダー」** → **「デフォルトのメッセージング」** → **「JMS 活動化仕様」**をクリックします。
   c. プロセス・サーバー・データベースに必要な並列 JDBC 接続の数 $p = 1.1 * t$ を計算します。

      注: *p* の値は、そのデータベースで許可される接続数以下にしてください。
   d. メッセージング・データベースに必要な並列 JDBC 接続数 $m = t + x$ を計算します。*x* は、追加メッセージが生成され処理される必要がある過負荷状態を許容できるようにするための、追加 JMS セッション数です。
   e. SQL 「ステートメント・キャッシュ・サイズ」を 30 に設定します。
2. プロセス・サーバー・データベース (BPEDB) の JDBC プロバイダー設定を調整します。
   a. **「最大接続数」**を値 *p* に設定します。 *p* の値は、そのデータベースで許可される接続数以下にしてください。
   b. SQL **「ステートメント・キャッシュ・サイズ」** を 300 に設定します。

3. メッセージング・データベースの JDBC プロバイダー設定を調整します。

   「**最大接続数**」を値 *m* に設定します。
4. ヒープ・サイズを調整します。

   サーバー・ヒープのサイズを決める場合の指針を以下に示します。
   - 256 MB では少なすぎ、ローパフォーマンスになる。
   - 512 MB は多くのシステムの初期ヒープ・サイズとして適当。
   - 1024 MB は合理的な上限。
5. ビジネス・プロセスが使用するサービスを調整します。 サポート用サービスが、サービスでの Business Process Choreographer の並行性の程度および負荷要求に対処できるように調整されていることを確認してください。

アプリケーション・サーバーが調整され、パフォーマンスが改善されます。

メッセージング・プロバイダーの細密チューニング

メッセージング・プロバイダーのパフォーマンスを向上させるには、このタスクを使用します。

WebSphere Platform Messaging を使用する場合は、『メッセージング・エンジンのデータ・ストアのチューニングと問題解決』を参照してください。

メッセージング・プロバイダーのパフォーマンスが向上します。

データベースの細密チューニング

このタスクを使用して、データベースを細密チューニングします。

ビジネス・プロセス・コンテナーおよびビジネス・プロセスが実行中であることが必要です。

共通問題は、データベースがロック・リスト・スペースを使い尽くすことによりロック状態が増大し、パフォーマンスに重大な影響が出ることです。そのため、実行されているビジネス・プロセスの構造に応じて、データベース管理システムの特定のパフォーマンスに関連したパラメーターの設定をカスタマイズする必要が生じることもあります。

注: DB2 を使用していない場合は、データベースのパフォーマンスのモニター、ボトルネックの識別と除去、およびパフォーマンスの細密チューニングについて、データベース管理システムの文書を参照してください。このトピックの以下の部分では、DB2 ユーザーへの提案を示します。

1. ロック・リスト・スペースを調整して、最適なパフォーマンスを保証します。

   DB2 インスタンスの db2diag.log ファイルを確認します。次の例にあるような項目を探します。

```
2005-07-24-15.53.42.078000 Instance:DB2 Node:000
PID:2352(db2syscs.exe)   TID:4360   Appid:*LOCAL.DB2.027785142343
data management  sqldEscalateLocks Probe:4   Database:BPEDB

ADM5503E The escalation of "10" locks on table "DB2ADMIN.ACTIVITY_INSTANCE_B_T"
to lock intent "X" has failed.  The SQLCODE is "-911".
```

このタイプのメッセージは、ビジネス・プロセス・アプリケーションの並列処理が改良され、使用可能なロックの数が小さくなりすぎるほどになったことを示します。LOCKLIST 値を概算で 10 * *p* に増やしてください。ここで、*p* は、任意の時点で必要になる並列 JDBC 接続の最大数の推定値です。 例えば、Business Process Choreographer データベースである BPEDB のサイズを、値として *p*=50 を使用して変更した場合は、次のコマンドを入力します。

```
db2 UPDATE DB CFG FOR BPEDB USING LOCKLIST 500
```

2. DB2 構成アドバイザーを使用した場合、データベースのスループットは通常良好な状態になっています。ただし、以下のようにしてさらにパフォーマンスを改善することができます。
   - DB2 オンライン文書、ブック、および記事で説明されている最良事例に従って、データベースのチューニングを行ってください。
   - DB2 モニターを使用し、データベース内部のボトルネックについての詳しい情報を db2diag.log ファイルで調べます。
   - データベースに対して定期的に runstats を実行します。
   - 以下の DB2 パラメーターを調整します。

     LOCKLIST
     ステップ 1 (375 ページ) の説明を参照してください。

     AVG_APPLS
     このパラメーターは、低すぎるよりも高すぎる方が、望ましい結果が得られます。例えば、最大 20 個の接続されたアプリケーションがある場合は、AVG_APPLS を 50 に設定します。

     LOGBUFSZ
     DB2 ログのバッファーのサイズを大きくすると、ログ・バッファー全体をディスクに書き込まなければならない頻度が減少します。

     LOG_FILSIZ
     ログ・ファイルのサイズを大きくすると、ログ・ファイルが切り替えられる頻度が減少します。

3. データベースおよびデータベース・マネージャーの設定をワークロード要件に応じて調整します。 構成アドバイザーがデータベースを構成したあと、以下の設定を調整することもできます。

   MINCOMMIT
   1 の値を強くお勧めします。DB2 Configuration Advisor は他の値を提示する場合があります。

   NUM_IOSERVERS
   データベースが常駐している物理ディスクの数に一致しなければなりません。少なくともディスクの数に等しい IOSERVER を持つ必要があります。IOSERVER はシステム・リソースをあまり使用しないので、低すぎる値を設定するよりは、高すぎる値を設定してください。

ユーザーの長期間にわたって実行するプロセスは、現在の環境とロード条件において可能な限り高速実行されます。

microflow のチューニング

microflow のパフォーマンスを向上させるには、このタスクを使用します。

microflow は、ユーザーとの対話や永続メッセージングのサポートなしに、メモリー内で実行されます。監査ロギングまたは Common Event Infrastructure (CEI) が microflow で使用可能になっている場合のみ、データベース・アクセスが必要です。microflow の処理が、単一スレッド内で行われ、また通常は単一トランザクション内で行われます。 microflows のパフォーマンスは、主として呼び出されるサービスによって決まります。ただし、サーバーが使用可能なメモリーが小さすぎる場合、microflow のパフォーマンスは低下します。

1. Java 仮想マシン (JVM) ヒープ・サイズを調整します。

   Java ヒープ・サイズを増やすことにより、microflow のスループットを向上させることができます。これは、ヒープ・サイズを大きくするほど、必要なガーベッジ・コレクション・サイクル数を減らすことができるからです。値は、ディスクへのヒープ・スワッピングを避けるため、十分に低い値に維持します。サーバー・ヒープのサイズの指針については、 374 ページの『アプリケーション・サーバーのチューニング』の関係するステップを参照してください。
2. JVM のガーベッジ・コレクションを調整します。 Throughput Garbage Collector を使用すると最高のスループットが得られますが、ヒープ・サイズによっては、ガーベッジ・コレクションが 100 から 1000 ミリ秒の間一時停止する場合があります。スループットより応答時間の方を重視する場合は、Low Pause Garbage Collector を使用してください。
3. データベース接続数が十分であることを確認します。 現在実行中の microflow ごとに、プロセス・データベースへの JDBC 接続が少なくとも 1 つは必要です。データベース自体への接続だけでなく、データ・ソースの接続プール内の接続についても、十分な数が必要です。
4. オブジェクト・リクエスト・ブローカー (ORB) のスレッド・プール・サイズを調整します。 リモート・クライアントがサーバー・サイド ORB に接続する場合は、ORB スレッド・プールに使用できるスレッドが十分あることを確認してください。
5. デフォルトのスレッド・プール・サイズを調整します。 同時に実行できる microflow の数を増やすには、デフォルトのスレッド・プール・サイズを大きくする必要があります。値を変更するには、管理コンソールで、「**アプリケーション・サーバー**」 → 「***server***」 → 「**スレッド・プール (Thread pools)**」 → 「**デフォルト**」をクリックします。

microflow は、現在の環境とロード条件の下で可能な限り高速に実行されます。

ヒューマン・タスクを含むビジネス・プロセスのチューニング

ヒューマン・タスクを含むビジネス・プロセスのパフォーマンスを向上させるには、さまざまな方法があります。

以下のトピックでは、ヒューマン・タスクを含むビジネス・プロセスを調整する方法を説明します。

ヒューマン・タスクへの同時アクセス数の削減

複数のユーザーが同じヒューマン・タスクを要求しようとした場合、1 人のユーザーだけが要求に成功します。他のユーザーのアクセスは拒否されます。

1 つのヒューマン・タスクを要求できるのは、1 人のユーザーのみです。何人かのユーザーが同時に同じヒューマン・タスクで作業を行おうとすると、衝突が起きる可能性が増します。衝突が起きると、データベースやロールバックでのロック待機のため、遅延が発生します。衝突の発生を回避または削減する方法として、以下のものがあります。

- 同時アクセス数が多い場合、特定のヒューマン・タスクにアクセスできるユーザー数を制限します。
- インテリジェントな要求機構を使用して、クライアントからの不必要なヒューマン・タスクの照会をなくします。例えば、以下の手順のいずれかを実行します。
  - 最初の要求に失敗した場合、リストから別の項目を要求します。
  - ヒューマン・タスクを常に無作為に要求します。
  - メンバー数の少ないグループにタスクを割り当てるなどして、タスクの潜在的な所有者の数を減らします。
  - リストの検索に使用する照会にしきい値を指定して、タスク・リストのサイズを制限します。ヒット数を制限するフィルター処理の使用も検討します。タスクのプロパティーをフィルターに掛けることによって、たとえば、優先順位が 1 番のタスクまたは 24 時間以内に期限の切れるタスクのみを表示できます。インライン・タスクの場合、カスタム・プロパティーまたは照会プロパティーを使用するタスクに関連するビジネス・データをフィルターに掛けることもできます。このようなフィルター処理を行うには、タスク・リストを検索する照会に適切な where 文節を指定する必要があります。
  - 動的スタッフ照会 (変数を使用する照会) を最小限に抑えるか、回避します。
  - ヒューマン・タスクの照会にクライアント・キャッシュ機構を使用して、一度に複数の照会が実行されないようにします。

照会の応答時間の短縮

データベースが照会への応答に必要とする時間を短縮します。

カスタム・クライアントを使用する場合は、必ず照会でしきい値を設定してください。使用可能度の観点から、一般的に、何百、何千もの項目の検索は望ましいことではありません。データベースの操作数が増えるほど、タスクの完了までにかかる時間が長くなること、および人は一度に少数の結果しか管理できないことがその理由です。しきい値を指定すると、データベースの負荷やネットワーク・トラフィックを最小限に押さえ、クライアントが確実にデータをすぐに提供できるようになります。

多数の項目を戻す照会の処理方法としてよりよい方法は、照会を作成し直して、戻される結果セットの項目数を減らすことです。これを行うには、特定のプロセス・インスタンスのみについて作業項目を照会するか、一定の日付を持つ作業項目のみを照会します。

フィルター基準を使用して、照会結果を削減することもできます。

全テーブルのスキャンニングの回避

照会アプリケーション・プログラミング・インターフェース (API) を使用して、データベース内のオブジェクトをリストする場合、フィルター操作を指定して、取得する結果の範囲を絞ることができます。このフィルターに、オブジェクト属性の値と範囲を指定することができます。

データベース照会の処理時に、フィルター情報が変換され、Structured Query Language (SQL) ステートメントの WHERE 文節に格納されます。この WHERE 文節は、影響を受けるデータベース・テーブル内の列名にオブジェクト属性をマップします。

照会に索引付きテーブル列への変換を行わないフィルターが指定されている場合、SQL ステートメントによってテーブルがスキャンされることになります。このスキャンニングによって、パフォーマンスに悪影響があり、デッドロックが発生するリスクが高くなります。このパフォーマンスの悪影響は、1 日に数回のみという程度であれば容認されますが、1 分間に数回という頻度になれば、効率に悪影響を及ぼします。

このような状況では、カスタム索引でこの影響を劇的に抑えることができます。実際のお客様の状況で、カスタム索引により、API の応答時間を 25 秒から 300 ミリ秒に減らすことができました。データベース・テーブルの 724 000 行を読み取る代わりに、6 行を読み取るだけでよいのです。

指定するフィルター基準により、一部の列が索引に含まれないように設定することができます。該当する場合、テーブル・スキャンを使用し、その結果照会のパフォーマンスが低下する場合は、例えば DB2 Explain を使用して、ステートメントのアクセス・パスを確認してください。必要であれば、新しい索引を定義してください。

第 9 章 Business Process Choreographer のトラブルシューティング

Business Process Choreographer 構成のトラブルシューティング

このトピックを参照して、ビジネス・プロセス・コンテナーまたはヒューマン・タスク・コンテナーの構成に関連した問題を解決します。

このセクションの目的は、ビジネス・プロセス・コンテナーまたはヒューマン・タスク・コンテナーの構成が期待どおりに機能しない理由を理解したり、問題を解決したりする際に役立つ情報を提供することです。以下のタスクは、ビジネス・プロセス・コンテナーまたはヒューマン・タスク・コンテナーの構成中に発生する可能性がある問題の判別および解決方法に焦点を当てています。

Business Process Choreographer のログ・ファイル

ここでは、Business Process Choreographer 構成のログ・ファイルの保管場所について説明します。

プロファイルの作成

Business Process Choreographer のプロファイル・アクションでは、logs ディレクトリーの bpcaugment.log ファイルに書き込みを行います。

プロファイル・ウィザードでサンプル構成オプションを選択すると、bpeconfig.jacl スクリプトが呼び出され、アクションは logs ディレクトリーの bpeconfig.log ファイルに記録されます。

管理スクリプト

wsadmin を使用して実行されるすべての Business Process Choreographer スクリプトは、wsadmin.traceout ファイルに記録されます。ただし、このファイルは wsadmin が起動されるごとに上書きされるので、wsadmin を再度起動する前に必ずこのログ・ファイルを保管してください。

構成関連スクリプト

bpeconfig.jacl、taskconfig.jacl、clientconfig.jacl、および bpeunconfig.jacl の各スクリプト・ファイルは、それぞれ bpeconfig.log、taskconfig.log、clientconfig.log、および bpeunconfig.log という名前でログ・ファイルを logs ディレクトリーに書き込みます。構成スクリプト setUpEventCollector.bat (UNIX システムでは .sh) および setupObserver.bat (UNIX システムでは .sh) は、ログ・ファイルを、それぞれ setupOberver.log および setupEventCollector.log ファイルとして logs ディレクトリーに書き込みます。また、wsadmin.traceout ファイルも確認します。

管理ユーティリティー・スクリプト

ProcessChoreographer ディレクトリーの util サブディレクトリーにある管理スクリプトは、独自のログ・ファイルを書き込みません。wsadmin.traceout ファイルおよびアプリケーション・サーバーのログ・ファイルを確認します。

構成チェッカー

ProcessChoreographer/config ディレクトリーにある bpecheck.jacl スクリプト・ファイルを使用して、共通の構成問題を確認できます。結果は、logs ディレクトリーの bpecheck.log ファイルに書き込まれます。

Business Process Choreographer のトレースの使用可能化

ここでは、サポートに連絡する前に行うべきことについて説明します。

トレースの使用可能化

Business Process Choreographer トレースでは、標準の WebSphere Process Server トレース・メカニズムを使用します。このメカニズムは通常の方法で使用可能に設定する必要があります。

トレース仕様は次のとおりです。

```
*=info:com.ibm.bpe.*=all:com.ibm.task.*=all:com.ibm.ws.staffsupport.*=all
```

ここで、`com.ibm.bpe.*=all` はビジネス・プロセスをトレースし、`com.ibm.task.*=all` はヒューマン・タスクをトレースします。ヒューマン・タスクの残りの局面であるスタッフ・プラグインは、`com.ibm.ws.staffsupport` によってトレースされます。

サポートに送信するもの

トレースを使用可能に設定後、問題シナリオを再作成して以下のファイルを準備します。

- SystemOut.log
- SystemErr.log
- trace.log

これらのファイルは次のディレクトリーにあります。*install_root*/profiles/*profile_name*/logs/

ffdc フォルダーにある WebSphere Application Server FFDC ログには、トレースが存在しないときのサポートに役立つ情報も含まれます。

タスク・コンテナー・アプリケーションの開始に失敗する

ejb/htm/TaskContainerStartUpBean という名前の開始 Bean がアプリケーションを強制的に停止しました。

症状

次のエラーが SystemOut.log ファイルに書き込まれます。

```
WSVR0037I: Starting EJB jar: taskejb.jar
NMSV0605W: A Reference object looked up from the context "java:"
with the name "comp/env/scheduler/DefaultUserCalendarHome"
was sent to the JNDI Naming Manager and an exception resulted.
Reference data follows:
Reference Factory Class Name: com.ibm.ws.naming.util.IndirectJndiLookupObjectFactory
Reference Factory Class Location URLs:
Reference Class Name: java.lang.Object
Type: JndiLookupInfo
Content: JndiLookupInfo:
jndiName="com/ibm/websphere/scheduler/calendar/DefaultUserCalendarHome";
providerURL=""; initialContextFactory=""
:
StartBeanInfo E STUP0005E: Startup bean named ejb/htm/TaskContainerStartUpBean
forced application to stop.
ApplicationMg W WSVR0101W: An error occurred starting, TaskContainer_utxtlb10Node01_server1
ApplicationMg A WSVR0217I: Stopping application: TaskContainer_utxtlb10Node01_server1
EJBContainerI I WSVR0041I: Stopping EJB jar: taskejb.jar
```

理由

このエラーは、TaskContainer アプリケーションの開始時に SchedulerCalendars アプリケーションが使用不可になっている場合に発生します。

解決方法

SchedulerCalendars アプリケーションを手動でインストールするか、既にインストール済みの場合は、新規のターゲット・マッピングを追加します。

デフォルト・プロファイルでは、SchedulerCalendars アプリケーションが、WebSphere システム・アプリケーションとして自動的に使用可能になります。ただし、カスタム・プロファイルでは自動的に使用可能にはなりません。

bpeconfig.jacl スクリプトは、SchedulerCalendars アプリケーションのインストールを試行しますが、常に可能とは限りません。

管理コンソールのインストール・ウィザードを使用して ND 環境の Business Process Choreographer を構成する場合は、SchedulerCalendars アプリケーションを手動でインストールする必要があります。

Business Process Choreographer データベースおよびデータ・ソースのトラブルシューティング

このタスクを使用して、Business Process Choreographer データベースおよびデータ・ソースの問題を解決します。

ビジネス・プロセス・コンテナーとヒューマン・タスク・コンテナーは、両方ともデータベースを必要とします。データベースがなければ、ビジネス・プロセスおよびヒューマン・タスクを含むエンタープライズ・アプリケーションは機能しません。

- DB2 を使用している場合:
  - DB2 Universal JDBC ドライバー・タイプ 4 を使用しており、Business Process Choreographer データ・ソースで接続をテストするとき、またはサーバーを始動するときに、"com.ibm.db2.jcc.a.re: XAER_RMERR : The DDM parameter value is not supported. DDM parameter code point having unsupported

value : 0x113f DB2ConnectionCorrelator: NF000001.PA0C.051117223022" などの DB2 内部エラーが発生する場合は、以下のアクションを実行します。

1. データ・ソースのクラスパス設定を確認します。デフォルト・セットアップでは、WebSphere 変数 ${DB2UNIVERSAL_JDBC_DRIVER_PATH} により、universalDriver_wbi ディレクトリーにある WebSphere Process Server 組み込み DB2 Universal JDBC ドライバーを指すことができます。
2. ドライバーのバージョンは、DB2 サーバー・バージョンと互換性がない場合があります。ご使用のデータベース・インストールのオリジナル db2jcc.jar ファイルを使用しており、WebSphere Process Server 組み込み DB2 Universal JDBC ドライバーではないことを確認します。必要な場合は、オリジナル db2jcc.jar ファイルを指すように、WebSphere 変数 ${DB2UNIVERSAL_JDBC_DRIVER_PATH} の値を変更します。
3. サーバーを再始動します。

– DB2 インスタンスの db2diag.log ファイルに、以下に示す ADM5503E のようなメッセージが含まれている場合:

```
2004-06-25-15.53.42.078000   Instance:DB2   Node:000
PID:2352(db2syscs.exe)   TID:4360   Appid:*LOCAL.DB2.027785142343
data management  sqldEscalateLocks Probe:4   Database:BPEDB

ADM5503E  The escalation of "10" locks on table "GRAALFS .ACTIVITY_INSTANCE_T"
          to lock intent "X" has failed. The SQLCODE is "-911"
```

LOCKLIST 値を増やします。例えば、この値を 500 に設定するには、以下の DB2 コマンドを入力します。

```
db2 UPDATE DB CFG FOR BPEDB USING LOCKLIST 500
```

これにより、パフォーマンスが大きく向上します。

– デッドロックを回避するには、データベース・システムが十分なメモリー (特にバッファー・プール用) を使用するように構成されていることを確認します。DB2 の場合は、DB2 Configuration Advisor を使用して、ご使用の構成に合った値を判別します。
– データ・ソースのインプリメンテーション・クラス COM.ibm.db2.jdbc.DB2XADataSource について述べるエラーが発生した場合:
  - server.policy ファイルで使用されているすべての WebSphere 環境変数が、正しく設定されていることを確認します。例えば、DB2_INSTALL_ROOT と DB2_JDBC_DRIVER_PATH です。
  - ご使用の JDBC プロバイダーのクラスパス定義が正しいこと、および項目が 2 つ存在しないことを確認します。
  - コンポーネント管理認証別名が、*cellName*/BPEAuthDataAlias*dbType_Scope* に設定されていることを確認します。ここで、*cellName* はセルの名前、*dbType* はデータベース・タイプ、および *Scope* は定義のスコープです。
– リモートの DB2 for z/OS データベースを使用している場合で、アプリケーション・サーバーがリモート・データベースとの最初の XA トランザクションを開始しようとするときに、SystemOut.log ファイルに SQL コード 30090N がある場合は、以下を実行します。
  - インスタンス構成変数 SPM_NAME が、8 文字以内のホスト名を持つローカル・サーバーを指していることを確認します。ホスト名が 8 文字より長い場合は、etc/hosts ファイルで短い別名を定義します。

- その他の場合、無効な同期点マネージャー・ログ項目が sqllib/spmlog ディレクトリーにある可能性があります。 sqllib/spmlog ディレクトリーの項目のクリアを試行して再始動します。
- SPM_LOG_FILE_SZ の値を増やすことを考慮します。

- Cloudscape を使用している場合:
  - Linux または UNIX システムで「開かれたファイルが多すぎます (Too many open files)」エラーが発生した場合は、使用可能なファイル・ハンドルの数を例えば 4000 以上に増やします。使用可能なファイル・ハンドルの数を増やす方法について詳しくは、ご使用のオペレーティング・システムの資料を参照してください。
  - Cloudscape ツールを起動しようとすると「Java クラスが見つかりません (Java class not found)」という例外が発生する場合は、 Java 環境がセットアップされているか、および classpath 環境変数に以下の JAR ファイルが含まれているか確認します。
    - db2j.jar
    - db2jtools.jar
    - db2jcc.jar
    - db2jcview.jar
  - Cloudscape ツール (ij または cview など) を使用して Cloudscape データベースに接続できず、以下の例外が発生する場合:

```
ERROR XJ040: Failed to start database 'c:¥WebSphere¥AppServer¥profiles¥profile_name¥databases¥BPEDB',
  see the next exception for details.
ERROR XSDB6: Another instance of Cloudscape may have already booted the database
  c:¥WebSphere¥AppServer¥profiles¥profile_name¥databases¥BPEDB.
```

    Cloudscape データベースにアクセスできるアプリケーションは一度に 1 つだけなので、これらのツールを使用する前に WebSphere Application Server を停止する必要があります。

- ビジネス・プロセスまたはヒューマン・タスクを含むエンタープライズ・アプリケーションのインストール中に、データベース・エラーが発生する場合。 エンタープライズ・アプリケーションをインストールすると、プロセス・テンプレートとタスク・テンプレートは Business Process Choreographer データベースに書き込まれます。ビジネス・プロセス・コンテナーが使用するデータベース・システムが稼動しており、アクセス可能であることを確認します。
- 国別文字の使用に問題がある場合。 Unicode 文字セットをサポートするようデータベースが作成されていることを確認します。
- データベースの中でテーブルまたはビューが検出されない場合。 データ・ソースの認証別名を構成するときには、データベース表の作成に使用したユーザー ID (またはデータベース表を作成するスクリプトの実行に使用したユーザー ID) と同じユーザー ID を指定する必要があります。

ビジネス・プロセスとヒューマン・タスクのトラブルシューティング

このトピックを参照して、ビジネス・プロセスおよびヒューマン・タスクに関連した問題を解決します。

以下のタスクは、ビジネス・プロセスまたはタスクの実行中に発生する可能性のある問題のトラブルシューティングに焦点を当てています。

ビジネス・プロセス・アプリケーションとヒューマン・タスク・アプリケーションのインストールのトラブルシューティング

ND 環境で、ビジネス・プロセス、ヒューマン・タスク、またはその両方を含むアプリケーションをインストールするときに、デプロイメント・マネージャーの SystemErr.log ファイルに例外が記録されます。

症状

ND 環境で、ビジネス・プロセス、ヒューマン・タスク、またはその両方を含むアプリケーションをインストールするときに、デプロイメント・マネージャーの SystemErr.log ファイルで以下の例外が検出されます。

```
SystemErr R com.ibm.ws.management.commands.sib.SIBAdminCommandException:
CWSJA0012E: Messaging engine not found.
at com.ibm.ws.management.commands.sib.SIBAdminCommandHelper.createDestination
 (SIBAdminCommandHelper.java:787)
at com.ibm.ws.management.commands.sib.CreateSIBDestinationCommand.afterStepsExecuted
 (CreateSIBDestinationCommand.java:459)
at com.ibm.websphere.management.cmdframework.provider.AbstractTaskCommand.execute
 (AbstractTaskCommand.java:547)
at com.ibm.ws.sca.internal.deployment.sib.SIBAdminHelper.call(SIBAdminHelper.java:136)
at com.ibm.ws.sca.internal.deployment.sib.SIBAdminHelper.createSIBDestination
 (SIBAdminHelper.java:112)
at com.ibm.ws.sca.internal.deployment.sib.SIBAdmin.createDestination(SIBAdmin.java:327)
at com.ibm.ws.sca.internal.deployment.sib.SIBDestinationTask.createDestination
 (SIBDestinationTask.java:263)
at com.ibm.ws.sca.internal.deployment.sib.SIBDestinationTask.preInstallModule
 (SIBDestinationTask.java:71)
at com.ibm.ws.sca.internal.deployment.SCATaskBase.installModule(SCATaskBase.java:57)
at com.ibm.ws.sca.internal.deployment.sib.SIBDestinationTask.processArtifacts
 (SIBDestinationTask.java:228)
at com.ibm.ws.sca.internal.deployment.sib.SIBDestinationTask.install
 (SIBDestinationTask.java:287)
at com.ibm.ws.sca.internal.deployment.SCAInstallTask.performInstallTasks
 (SCAInstallTask.java:116)
at com.ibm.ws.sca.internal.deployment.SCAInstallTask.performTask
 (SCAInstallTask.java:61)
at com.ibm.ws.management.application.SchedulerImpl.run(SchedulerImpl.java:253)
at java.lang.Thread.run(Thread.java:568)
```

理由

「SCA.SYSTEM.cellName.Bus」バスのバス・メンバーが欠落しています。

解決方法

管理コンソールで、**「サービス統合」 → 「バス」 → SCA.SYSTEM.*cellName*.Bus** をクリックします。「トポロジー」セクションで、**「バス・メンバー (Bus members)」** をクリックします。ビジネス・プロセス・アプリケーションまたはヒューマン・タスク・アプリケーションをバス・メンバーとしてインストールするサーバーまたはクラスターを追加してから、該当サーバーまたはクラスターを再始動し、再度アプリケーションのインストールを試行します。

ビジネス・プロセスの実行のトラブルシューティング

ここでは、ビジネス・プロセスの実行に関する共通の問題の解決方法について説明します。

Business Process Choreographer Explorer を使用し、IBM 技術サポート・ページでエラー・メッセージ・コードを検索できます。

1. エラー・ページで、「**詳しくは、検索してください**」リンクをクリックします。これにより、IBM 技術サポート・サイトでエラー・コードの検索を開始します。このサイトで提供される情報は英語のみです。
2. エラー・ページに表示されるエラー・メッセージ・コードをクリップボードにコピーします。エラー・コードの形式は `CWWBcnnnnc` で、`c` は文字を、`nnnn` は 4 桁の数値を示します。「WebSphere Process Server technical support」ページに進みます。
3. エラー・コードを「**追加検索項目 (Additional search terms)**」フィールドに貼り付けて、「**実行 (Go)**」をクリックします。

特定の問題に対する解決方法を以降のトピックで説明します。

Microflow を含むアプリケーションを停止するときの ClassCastException

Microflow を含むアプリケーションが停止した時間帯に、ClassCastException 例外が SystemOut.log ファイルに記述されます。

理由

アプリケーションが停止するときに、EAR ファイルに含まれるクラスはクラスパスから除去されます。ただし、それらのクラスを必要とする Microflow インスタンスがまだ実行されている可能性があります。

解決方法

以下のアクションを実行します。

1. まず、Microflow プロセス・テンプレートを停止します。この時点以降、そのテンプレートから新規の Microflow インスタンスを開始することはできなくなります。
2. 少なくとも Microflow が実行される最大期間が過ぎるのを待ち、実行中のインスタンスすべてを完了させます。
3. アプリケーションを停止します。

XPath 照会により配列から予期しない値が戻される

XPath 照会を使用して配列のメンバーにアクセスすると、予期しない値が戻されます。

理由

この問題の一般的な理由は、配列の最初のエレメントの指標値がゼロになっていると考えられます。配列の XPath 照会では、最初のエレメントの指標値は 1 になっています。

解決方法

配列で使用する指標値がエレメント 1 で始まっていることを確認します。

アクティビティーは処理不能の障害のため停止しました (メッセージ: CWWBE0057I)

システム・ログに CWWBE0057I メッセージがあります。プロセスは「実行」状態になっていますが、現行パスでナビゲーションが先に進みません。

理由

次のすべての状態が発生した場合は、invoke アクティビティー、インライン・ヒューマン・タスク、および Java 断片を停止状態にします。

- アクティビティーにより障害が発生した
- エンクロージング・スコープで障害が処理されない
- アクティビティーの continueOnError 属性が false に設定されている

解決方法

この問題を解決するには、次の 2 つのレベルの処置が必要です。

1. 管理者は、停止したアクティビティー・インスタンスを手動で修復する必要があります。例えば、停止したアクティビティー・インスタンスを強制完了するか、強制再試行します。
2. 障害の理由を調査する必要があります。場合によっては、モデリング・エラーによって障害が発生することもあり、その場合はモデル内で修正する必要があります。

例えば、WebSphere Scheduler デフォルト・カレンダーを使用し、アクティビティーに有効期限が定義されている場合は、時間が正しい形式で定義されていることを確認します。特に、時間の数字と単位の間に空白がないことを確かめます。正しく指定されたタイムアウト期間の例を次に示します。

- 1minute
- 2hours 4minutes 1second
- 1day 1hour

Microflow が補正されない

Microflow がサービスを呼び出したときにプロセスが失敗しましたが、元に戻すサービスが呼び出されません。

解決方法

Microflow の補正を起動するには、さまざまな条件を満たす必要があります。次の点を確認します。

1. Business Process Choreographer Explorer にログオンし、「**失敗した補正**」をクリックして、補正サービスが失敗しており、修復する必要があるかどうかを確認します。
2. Microflow の補正は、Microflow のトランザクションがロールバックした場合にのみ起動されます。この場合に該当するかどうか確認してください。

3. Microflow の compensationSphere 属性を「必須」に設定する必要があります。
4. 補正サービスが実行されるのは、対応する転送サービスが Microflow のトランザクションに関わっていない場合のみです。転送サービスがナビゲーション・トランザクションに関わっていないことを確認してください。例えば、プロセス・コンポーネントの参照時には、Service Component Architecture (SCA) の修飾子 suspendTransaction を True に設定します。

長期実行プロセスが停止しているように見える

長期実行プロセスは実行中の状態になっていますが、何も動作していないように見えます。

理由

このような振る舞いをすることにはさまざまな理由が考えられます。

1. ナビゲーション・メッセージを再試行した回数が多すぎるため、保存キューまたは保留キューに移動された。
2. Service Component Architecture (SCA) インフラストラクチャーからの応答メッセージが繰り返し失敗した。
3. プロセスが、イベント、タイムアウト、あるいは長期実行呼び出しまたはタスクが戻るのを待っている。
4. プロセスのアクティビティーが停止状態になっている。

解決方法

上述の理由それぞれに対して、異なる修正アクションが必要になります。

1. 管理についての PDF に記載のとおり、保存キューまたは保留キューにメッセージがあるかどうかを確認します。
2. 管理コンソールの失敗したイベント管理ビューに何か表示されているかどうかを確認します。
   - Service Component Architecture (SCA) 応答メッセージからの失敗したイベントがある場合は、そのメッセージを再活動化します。
   - それ以外は、長期実行アクティビティーを強制完了するか、強制再試行します。
3. 停止状態のアクティビティーが存在するかどうかを調べ、存在する場合はそれらのアクティビティーを修復します。システム・ログに CWWBE0057I メッセージがある場合は、『メッセージ: CWWBE0057I』に記載のとおり、モデルを訂正することも必要になります。

別の EAR ファイルの同期サブプロセスの呼び出しが失敗する

長期実行プロセスが別のプロセスを同期して呼び出し、サブプロセスが別のエンタープライズ・アーカイブ (EAR) ファイルに配置されている場合は、そのサブプロセスの呼び出しは失敗します。

例えば、次のような例外が発生します。

```
com.ibm.ws.sca.internal.ejb.util.EJBStubAdapter com.ibm.ws.sca.internal.ejb.util.EJBStubAdapter#003
Exception:
java.rmi.AccessException: CORBA NO_PERMISSION 0x49424307 No; nested exception is:
org.omg.CORBA.NO_PERMISSION: The WSCredential does not contain a forwardable token.
```

```
Please enable Identity Assertion for this scenario.
vmcid: 0x49424000 minor code: 307 completed: No
at com.ibm.CORBA.iiop.UtilDelegateImpl.mapSystemException(UtilDelegateImpl.java:202)
at javax.rmi.CORBA.Util.mapSystemException(Util.java:84)
```

理由

別の EAR ファイルの同期サブプロセスを呼び出す場合は、Common Secure Interoperability バージョン 2 (CSIv2) ID アサーションを使用可能に設定する必要があります。

解決方法

CSIv2 インバウンド認証と CSIv2 アウトバウンド認証を構成します。

実行中に予期しない例外が発生しました (メッセージ: CWWBA0010E)

キュー・マネージャーが実行されていないか、Business Process Choreographer 構成に誤ったデータベース・パスワードが含まれています。

解決方法

次の点を確認します。

1. `systemout.log` ファイルに `"javax.jms.JMSException: MQJMS2005: failed to create MQQueueManager"` が含まれている場合、キュー・マネージャーを開始してください。
2. Business Process Choreographer 構成に保管されているデータベース管理者パスワードが、データベースで設定されているパスワードと一致していることを確認してください。

不明のイベント (メッセージ: CWWBE0037E)

プロセス・インスタンスにイベントを送信、または新規プロセス・インスタンスを開始しようとして、「CWWBE0037E: 不明のイベント」例外が発生しました。

理由

このエラーの共通の理由は、メッセージをプロセスに送信しても、receive または pick アクティビティーが既にナビゲート済みであるために、メッセージを再度このプロセス・インスタンスでコンシュームできないというものです。

解決方法

この問題を訂正するには、次の指示に従います。

- イベントが既存のプロセス・インスタンスによってコンシュームされることになっている場合は、対応する receive または pick アクティビティーをまだナビゲートしていない既存のプロセス・インスタンスと一致する、相関セット値を渡す必要があります。
- イベントが新規プロセス・インスタンスを開始させることになっている場合は、相関セット値が既存のプロセス・インスタンスと一致してはいけません。

ビジネス・プロセスで相関セットを使用する方法について詳しくは、「technote 1171649」を参照してください。

プロセス・インスタンスを検索できないか、または作成できません (メッセージ: CWWBA0140E)

プロセス・インスタンスにイベントを送信しようとして、「'CreateRejectedException'」メッセージを受け取ります。

理由

このエラーの一般的な理由は、`createInstance` 属性が `no` に設定され、このアクティビティーで使用される相関セットのメッセージと一緒に渡される値が既存のプロセス・インスタンスと一致しないために、新規プロセス・インスタンスをインスタンス化できない receive または pick アクティビティーにメッセージが送信されるというものです。

解決方法

この問題を訂正するには、既存のプロセス・インスタンスと一致する相関セット値を渡す必要があります。

ビジネス・プロセスでの相関セットの使用法について詳しくは、「Correlation sets in BPEL processes」を参照してください。

Java 断片の未初期化変数または NullPointerException

ビジネス・プロセスで未初期化変数を使用すると、さまざまな例外が発生します。

症状

次のような例外が発生します。

- 変数の内容を読み取ったり操作したりする Java 断片または Java 式の実行時に、NullPointerException がスローされます。
- assign、invoke、reply、または throw アクティビティーの実行時に、BPEL の標準障害「uninitializedVariable」(メッセージ CWWBE0068E) がスローされます。

理由

ビジネス・プロセスのすべての変数は、プロセスが開始されるときに NULL 値を持っており、それらの変数は事前初期化されません。 Java 断片または Java 式の内側で未初期化変数を使用すると、NullPointerException が発生します。

解決方法

変数は、使用する前に初期化する必要があります。これは、assign アクティビティーで実行できます。例えば、変数は assign の to-spec に現れる必要があります。あるいは、Java 断片の内側で変数を初期化することが可能です。

標準障害の例外「missingReply」(メッセージ: CWWBE0071E)

microflow または長期実行プロセスを実行すると、BPEL 標準障害「missingReply」(メッセージ: CWWBE0071E) が発生するか、このエラーがシステム・ログまたは SystemOut.log ファイルで検出されます。

理由

両方向オペレーションでは応答を送信する必要があります。このエラーは、プロセスが reply アクティビティーをナビゲートせずに終了する場合に生成されます。この状態は、以下の事情の下で発生します。

- reply アクティビティーがスキップされた。
- 障害が発生し、対応する障害ハンドラーに reply アクティビティーが含まれていない。
- 障害が発生し、対応する障害ハンドラーが存在しない。

解決方法

モデルを訂正し、プロセスの終了前に reply アクティビティーが必ず実行されるようにします。

並列パスが順次化される

flow アクティビティーの内側に 2 つ以上の並列 invoke アクティビティーがありますが、invoke アクティビティーが順次実行されます。

解決方法

- 真の並列処理を実現するには、それぞれのパスを別々のトランザクションに置く必要があります。すべての並列 invoke アクティビティーの「トランザクションの振る舞い」属性を、「事前コミット (commit before)」または「所有が必要 (requires own)」に設定します。
- データベース・システムとして Cloudscape を使用している場合、プロセス・エンジンは並列パスの実行を直列化します。この振る舞いを変更することはできません。

ネストされたデータ・オブジェクトを別のデータ・オブジェクトにコピーするとソース・オブジェクトの参照が破棄される

データ・オブジェクト Father には、別のデータ・オブジェクト Child が含まれます。 Java 断片またはクライアント・アプリケーションの内側では、Child を含むオブジェクトがフェッチされ、データ・オブジェクト Mother の副構造で設定されます。データ・オブジェクト Father での Child への参照は消失します。

理由

Child への参照は、Father から Mother に移されます。

解決方法

上記のようなデータ形式変更を Java 断片またはクライアント・アプリケーションで実行し、Father に参照を保存する場合は、データ・オブジェクトを、別のオブジェクトに割り当てられる前にコピーします。以下のコード断片はその方法を示しています。

```
BOCopy copyService = (BOCopy)ServiceManager.INSTANCE.locateService
                        ("com/ibm/websphere/bo/BOCopy");
DataObject Child = Father.get("Child");
DataObject BCopy = copyService.copy(Child);
Mother.set("Child", BCopy);
```

CScope が使用不可である

長期実行プロセスでの Microflow の開始またはナビゲーション・ステップの実行が失敗し、「事後条件違反 !(cscope != null) (postcondition violation !(cscope != null))」というアサーションが表示されます。

理由

特定の状態ではプロセス・エンジンは補正サービスを使用しますが、これは使用不可でした。

解決方法

管理についての PDF に記載のとおり、補正サービスを使用可能に設定します。

プロセス関連またはタスク関連メッセージの操作

ディスプレイに表示またはログ・ファイルに書き込まれる Business Process Choreographer メッセージの詳細情報を取得する方法について説明します。

Business Process Choreographer に属するメッセージのプレフィックスは、プロセス関連メッセージの場合は *CWWB*、タスク関連メッセージの場合は *CWTK* です。これらのメッセージのフォーマットは、*PrefixComponentNumberTypeCode* です。タイプ・コードは、以下のとおりです。

I 情報メッセージ

W 警告メッセージ

E エラー・メッセージ

プロセスおよびタスクが実行されると、メッセージは Business Process Choreographer Explorer に表示されるか、`SystemOut.log` ファイルおよびトレースに追加されます。これらのファイルで提供されるメッセージ・テキストを使用しても問題を解決できない場合は、WebSphere Application Server 症状データベースを使用して詳細な情報を検索できます。Business Process Choreographer メッセージを表示するには、WebSphere ログ・アナライザーを使用して `activity.log` ファイルを確認します。

1. WebSphere ログ・アナライザーを開始します。

   次のスクリプト *install_root*`/bin/waslogbr.sh` を実行します。
2. **オプション: 「ファイル」>「データベースの更新」>「WebSphere Application Server 症状データベース」**をクリックして、症状データベースの最新バージョンを確認します。
3. **オプション:** アクティビティー・ログをロードします。
   a. アクティビティー・ログ・ファイルを選択します。
      - *install_root*`/profiles/`*profile_name*`/logs/activity.log` ファイル
   b. **「開く (Open)」**をクリックします。

Business Process Choreographer Explorer のトラブルシューティング

このトピックを参照して、Business Process Choreographer Explorer に関連した問題を解決します。

以下の情報を使用して、Business Process Choreographer Explorer に関連した問題を解決します。

- ブラウザーを使って Business Process Choreographer Explorer にアクセスしようとすると、エラー・メッセージ「HTTP 404 - File not found」が表示される場合は、次の操作を試してください。
  - 管理コンソールを使用して、Web クライアント・アプリケーション BPCExplorer_*node_name_server_name* がサーバー上に実際にデプロイされ、実行されていることを確認します。
  - 管理コンソールのアプリケーションのページにある「デプロイメント記述子の表示」の下で、コンテキスト・ルートが Business Process Choreographer Explorer をセットアップしたときに使用したコンテキスト・ルートであることを確認します。
- Business Process Choreographer Explorer の使用中にエラー・メッセージが表示される場合は、エラー・ページで「**詳しくは、検索してください**」リンクをクリックします。 これにより、IBM 技術サポート・サイトでエラー・コードの検索を開始します。このサイトで提供される情報は英語のみです。Business Process Choreographer Explorer エラー・ページに表示されるエラー・メッセージ・コードをクリップボードにコピーします。エラー・コードの形式は CWWBcnnnnc で、c は文字を、nnnn は 4 桁の数値を示します。「WebSphere Process Server technical support」ページに進みます。エラー・コードを「**追加検索項目 (Additional search terms)**」フィールドに貼り付けて、「**実行 (Go)**」をクリックします。
- StandardFaultException を標準障害 missingReply (メッセージ CWWBE0071E) とともに取得した場合、これはプロセス・モデルに関する問題の症状です。この問題の解決方法について詳しくは、395 ページの『ビジネス・プロセスおよびヒューマン・タスクの管理のトラブルシューティング』を参照してください。
- Business Process Choreographer Explorer にログオンできるものの、一部の項目が表示されない場合、または特定のボタンが使用できない場合、これはユーザーの権限に問題があることを示しています。

  この問題の解決方法として、次の方法が考えられます。
  - 管理コンソールを使用して、セキュリティーをオンにする。
  - 正しい ID を使用して Business Process Choreographer Explorer にログオンしているか確認する。プロセス管理者またはタスク管理者ではないユーザー ID でログオンすると、すべての管理ビューおよびオプションは非表示または使用不可になります。
  - WebSphere Integration Developer を使用して、ビジネス・プロセスに定義されている権限設定を確認または変更する。
- エラー・メッセージ CWWBU0001E: ″BFMConnection 関数の呼び出し時に通信エラーが発生しました。″ または ″HTMConnection 関数の呼び出し時に通信エラーが発生しました。″ このエラーは、ビジネス・プロセス・コンテナーまたはヒューマン・タスク・コンテナーがそれぞれ停止され、クライアントがサーバーに接

続できなくなったことを示します。ビジネス・プロセス・コンテナーおよびヒューマン・タスク・コンテナーが実行中でアクセス可能であることを確認してください。ネストされた例外に、問題に関する詳細情報が含まれる場合があります。

- エラー・メッセージ WWBU0024E: 理由「命名例外」で、ローカルのビジネス・プロセス EJB への接続を確立できませんでした。(WWBU0024E: ″Could not establish a connection to local business process EJB with a reason ″Naming Exception″.) ビジネス・プロセス・コンテナーが実行していないときにユーザーがログオンしようとすると、このエラーがスローされます。アプリケーション BPEContainer_*InstallScope* が実行中であることを確認してください。ここで、*InstallScope* は cluster_name または hostname_servername です。

関連タスク

387 ページの『ビジネス・プロセスの実行のトラブルシューティング』
ここでは、ビジネス・プロセスの実行に関する共通の問題の解決方法について説明します。

ビジネス・プロセスおよびヒューマン・タスクの管理のトラブルシューティング

ここでは、ビジネス・プロセスおよびヒューマン・タスクの共通問題を解決する方法について説明します。

次の情報は、ビジネス・プロセスおよびヒューマン・タスクの問題をデバッグするのに役立ちます。

- ビジネス・プロセス・アプリケーションにまだプロセス・インスタンスがある間にそのアプリケーションを停止しようとすると、管理コンソールは、応答を停止します。 アプリケーションを停止する前に、ビジネス・プロセスを停止して新規インスタンスが作成されないようにし、次のいずれかの操作を実行する必要があります。
  - すべての既存のプロセス・インスタンスが、正常な方法で終了するのを待つ。
  - すべてのプロセス・インスタンスを強制終了して削除する。

  これらの操作を行った後でのみ、プロセス・アプリケーションを停止できます。この問題を回避する方法について詳しくは、「technote 1166009」を参照してください。
- ヒューマン・タスク・アプリケーションにまだタスク・インスタンスがある間にそのアプリケーションを停止しようとすると、管理コンソールは、応答を停止します。 アプリケーションを停止するには、次のようにする必要があります。
  1. 新規インスタンスが作成されないようにするためにヒューマン・タスクを停止します。
  2. 以下のいずれかを実行します。
     - すべての既存のタスク・インスタンスが、正常な方法で終了するのを待つ。
     - すべてのタスク・インスタンスを強制終了して削除する。
  3. タスク・アプリケーションを停止します。

スタッフ・サービス、スタッフ・プラグイン、およびスタッフ解決のトラブルシューティング

以下の情報を使用して、許可ロールへのスタッフ割り当てに関連した問題を解決します。

以下のトピックを取り上げます。
- スタッフ動詞のデプロイメント時のエラー
- スタッフ・リポジトリー内の項目が作業項目割り当てに反映されていない
- タスクまたはプロセスの予期しないスタッフ割り当て
- スタッフ・アクティビティーの停止
- 作業項目割り当てに即時に反映されないスタッフ・リポジトリーへの変更
- スタッフ解決に関連したエラー・メッセージと警告メッセージ
- グループ作業項目および「グループ」動詞の問題

『テクニカル・サポート検索』ページでも、追加情報を検索できます。

スタッフ動詞のデプロイメント時のエラー

LDAP スタッフ・プラグインを使用する場合、プラグイン構成パラメーターの値が誤っているためにデプロイメントが失敗することがあります。すべての必須パラメーターが設定されていることを確認します。BaseDN パラメーターを LDAP ディレクトリー・ツリーのルートに設定するには、空ストリングを指定します。つまり、BaseDN パラメーターに 2 つのアポストロフィ (') 文字 ('') を設定します。二重引用符 (") は使用しないでください。BaseDN パラメーターの設定を失敗すると、デプロイメント時に NullPointerException 例外になります。

スタッフ・リポジトリー内の項目が作業項目割り当てに反映されていない

スタッフ照会によって検索されるユーザー ID の最大数は、使用中の XSL 変換ファイルで定義される Threshold 変数によって指定します。 LDAP スタッフ・プラグインに使用される XSL 変換ファイルの例は、`LDAPTransformation.xsl` であり、 Linux および UNIX の場合は *install-root*/ProcessChoreographer/Staff に置かれ、 Windows プラットフォームの場合は *install-root*¥ProcessChoreographer¥Staff に置かれます。デフォルトの Threshold 値は 20 です。この値を変更するには、次のようにします。

1. 新規スタッフ・プラグイン・プロバイダー構成を作成し、独自バージョンの XSL ファイルを指定します。
2. 必要に応じて、XSL ファイルの以下の項目を変更します。

```
<xsl:variable name="Threshold">20</xsl:variable>
```

注: 大きな Threshold 値を指定すると、パフォーマンスが低下する場合があります。このため、100 より大きな値を指定しないでください。

タスクまたはプロセス・インスタンスの予期しないスタッフ割り当て

タスクの特定のロールにスタッフ動詞を定義しない場合、またはスタッフ解決が失敗するか、結果を戻さない場合、デフォルトのスタッフ割り当てが実行されます。これらのデフォルトによって、予期しない許可がユーザーに与えられる可能性があります。例えば、プロセス・スターターがプロセス管理者権限を入手します。また、多くの許可が従属成果物によって継承されます。例えば、プロセス管理者が、すべてのインライン・タスクの管理者になる場合もあります。

次の表は、どのデフォルトがどの状態に該当するかを示しています。

表 41. ビジネス・プロセスのロール

ビジネス・プロセスのロール	**プロセス・モデルでロールが定義されていない場合 ...**	**プロセス・モデルでロールは定義されているが、スタッフ解決が失敗するか、適切な結果を戻さない場合...**
プロセス管理者	プロセス・スターターがプロセス管理者になる	次の例外が発生し、プロセスは開始されない。 EngineAdministratorCannotBeResolvedException
プロセス・リーダー	リーダーなし	リーダーなし

表 42. インライン・ヒューマン・タスクのロールおよびそれらのエスカレーション

インライン・ヒューマン・タスクのロールおよびそれらのエスカレーション	**タスク・モデルでロールが定義されていない場合 ...**	**タスク・モデルでロールは定義されているが、スタッフ解決が失敗するか、適切な結果を戻さない場合...**
タスク管理者	継承のみが適用される	継承のみが適用される
タスクの潜在的インスタンス作成者	だれでも潜在的インスタンス作成者になる	だれでも潜在的インスタンス作成者になる
タスクの潜在的スターター	だれでも潜在的スターターになる	だれでも潜在的スターターになる
タスクの潜在的所有者	だれでも潜在的所有者になる	管理者が潜在的所有者になる
タスク・エディター	エディターなし	エディターなし
タスク・リーダー	継承のみが適用される	継承のみが適用される
エスカレーション受信者	管理者がエスカレーション受信側になる	管理者がエスカレーション受信側になる

次の継承規則がインライン・タスクに適用されます。

- プロセス管理者が、すべてのインライン・タスク、それらのサブタスク、後続タスク、およびエスカレーションの管理者になります。
- プロセス・リーダーが、すべてのインライン・タスク、それらのサブタスク、後続タスク、およびエスカレーションのリーダーになります。
- タスク管理者が、これらすべてのタスクのすべてのサブタスク、後続タスク、およびエスカレーションの管理者になります。
- タスク・リーダーが、これらすべてのタスクのすべてのサブタスク、後続タスク、およびエスカレーションのリーダーになります。
- 任意のタスク・ロールのメンバーが、このタスクのエスカレーション、サブタスク、および後続のタスクのリーダーになります。
- エスカレーション受信側が、エスカレートしたタスクのリーダーになります。

表 43. スタンドアロン・ヒューマン・タスクのロールおよびそれらのエスカレーション

スタンドアロン・ヒューマン・タスクのロールおよびそれらのエスカレーション	タスク・モデルでロールが定義されていない場合 ...	タスク・モデルでロールは定義されているが、スタッフ解決が失敗するか、正しい結果を戻さない場合...
タスク管理者	オリジネーターが管理者になる	例外 AdministratorCannotBeResolvedException がスローされ、タスクは開始されない
タスクの潜在的インスタンス作成者	だれでも潜在的インスタンス作成者になる	だれでも潜在的インスタンス作成者になる
タスクの潜在的スターター	オリジネーターが潜在的スターターになる	例外 CannotCreateWorkItemException がスローされ、タスクは開始されない
潜在的所有者	だれでも潜在的所有者になる	管理者が潜在的所有者になる
編集者	エディターなし	エディターなし
リーダー	継承のみが適用される	継承のみが適用される
エスカレーション受信者	管理者がエスカレーション受信側になる	管理者がエスカレーション受信側になる

次の継承規則がスタンドアロン・タスクに適用されます。

- タスク管理者が、すべてのサブタスク、後続タスク、およびこれらすべてのタスクのエスカレーションの管理者になります。
- タスク・リーダーが、すべてのサブタスク、後続タスク、およびこれらすべてのタスクのエスカレーションのリーダーになります。
- 任意のタスク・ロールのメンバーが、このタスクのエスカレーション、サブタスク、および後続のタスクのリーダーになります。
- エスカレーション受信側が、エスカレートしたタスクのリーダーになります。

注: メソッドが Business Flow Manager API によって呼び出されると、J2EE ロール BPESystemAdministrator のメンバーは管理者権限を持ち、J2EE ロール BPESystemMonitor のメンバーはリーダー権限を持ちます。

注: メソッドが Human Task Manager API によって呼び出されると、J2EE ロール TaskSystemAdministrator のメンバーは管理者権限を持ち、J2EE ロール TaskSystemMonitor のメンバーはリーダー権限を持ちます。

スタッフ・アクティビティーの停止

次の問題が 1 つ以上発生する場合:

- ビジネス・プロセスは正常にナビゲートを開始しましたが、ヒューマン・タスクを要求できません。
- SystemOut.log ファイルに次のメッセージが含まれています。`CWWB0057I: プロセス 'MyProcess' のアクティビティー 'MyStaffActivity' は処理不能の障害のため停止しました...`

このメッセージは、WebSphere Application Server セキュリティーが使用可能になっていない可能性があることを示しています。要員の許可を使用する

ヒューマン・タスクやプロセスでは、セキュリティーを使用可能にすることとユーザー・レジストリーを構成することが必要です。次のステップを実行します。

1. WebSphere セキュリティーが使用可能になっていることを確認します。管理コンソールで、「**セキュリティー**」 → 「**グローバル・セキュリティー**」に移動して、「**グローバル・セキュリティーを使用可能に設定 (Enable global security)**」 のチェック・ボックスを選択してあることを確認します。
2. ユーザー・レジストリーが構成されていることを確認します。管理コンソールで、「**セキュリティー**」 → 「**ユーザー・レジストリー**」に移動して、「**アクティブ・ユーザー・レジストリー (Active user registry)**」属性にチェック・マークを付けます。
3. 停止している場合は、アクティビティーを再始動します。

スタッフ・リポジトリーへの変更が作業項目割り当てにすぐに反映されない。

Business Process Choreographer は、ランタイム・データベースで、Lightweight Directory Access Protocol (LDAP) サーバーなど、スタッフ・ディレクトリーに対して評価されたスタッフ割り当ての結果をキャッシュに入れます。スタッフ・ディレクトリーで変更が発生しても、変更はすぐにはデータベース・キャッシュに反映されません。

「*管理者ガイド* 」に、このキャッシュを最新表示する次の 3 通りの方法が説明されています。

- **管理コンソールを使用したスタッフ照会結果の最新表示**。大幅な変更があり、ほとんどすべてのスタッフ照会の結果を最新表示する必要がある場合にこの方式を使用します。
- **管理コマンドを使用したスタッフ照会結果の最新表示**。 wsadmin ツールを使用して管理スクリプトを作成するか、スタッフ照会結果のサブセットのみをすぐに最新表示する場合に、この方式を使用します。
- **refresh デーモンを使用したスタッフ照会結果の最新表示**。期限切れのすべてのスタッフ照会結果を定期的に自動で更新するようにする場合に、この方式を使用します。

注: これらのどの方式も、グループ動詞のユーザーのグループ・メンバーシップ関連を最新表示しません。このグループ・メンバーシップは、デフォルトで 2 時間後に期限が切れるユーザー・ログイン・セッション (WebSphere セキュリティー LTPA トークン) にキャッシュされます。プロセス・ナビゲーションに使用されるプロセス・スターター ID のグループ・メンバーシップ・リストは最新表示されないことにも注意してください。

スタッフ解決に関連したエラー・メッセージと警告メッセージ

スタッフ解決中にスタッフ・リポジトリーにアクセスすると、一般的エラーが発生する場合があります。これらのエラーの詳細を確認するには、次のトレース設定を使用してトレースを使用可能にします。 `com.ibm.bpe.*=all:com.ibm.task.*=all:com.ibm.ws.staffsupport.ws.*=all`

警告またはエラー・メッセージによって、次の一般的エラー状態が示されます。

- trace.log ファイルの `Could not connect to LDAP server` は、 LDAP サーバーに接続できないことを示します。ネットワーク設定、使用するスタッフ・プラグイン・プロバイダーの構成 (特にプロバイダー URL) を検査し、LDAP サーバーで SSL 接続が必要かどうかを検査します。
- System.out または System.err ファイルの `javax.xml.transform.TransformerException: org.xml.sax.SAXParseException: Element type "xsl:template" must be followed by either attribute specifications, ">" or "/>"` は、LDAPTransformation.xsl ファイルを読み取れないことを示します。スタッフ・プラグイン・プロバイダー構成、および構成済み XSLT ファイルに誤りがないかどうかを確認します。
- trace.log ファイルの `LDAP object not found. dn: uid=unknown,cn=users,dc=ibm,dc=com [LDAP: error code 32 - No Such Object]` は、LDAP 項目が見つからないことを示します。タスク・モデルのスタッフ動詞パラメーターおよび LDAP ディレクトリーの内容を確認し、タスク・モデルに不一致がないかどうかを検査します。
- trace.log ファイルの `Requested attribute "uid" not found in: uid=test222,cn=users,dc=ibm,dc=com` は、照会された LDAP オブジェクトで属性が見つからないことを示します。タスク・モデルのスタッフ動詞パラメーターおよび LDAP ディレクトリーの内容を確認し、タスク・モデルに不一致がないかどうかを検査します。スタッフ・プロバイダー構成の XSLT ファイルにエラーがないかどうかも検査します。

グループ作業項目および「グループ」動詞の問題

グループ動詞を使用すると、次のような特別な状態が発生する場合があります。

- グループ名が指定されていても、グループ・メンバーが許可されません。
  - WebSphere セキュリティーのローカル OS レジストリーを使用するときにグループの短い名前を指定し、 LDAP レジストリーを使用するときにグループ dn を指定します。
  - グループ名の大/小文字を区別します。

  この状態が発生する理由の 1 つとして、 WebSphere セキュリティーに LDAP ユーザー・レジストリーを構成し、「**許可において大/小文字を無視 (Ignore case for authorization)**」オプションを選択したことが考えられます。この場合、オプションの選択を解除するか、LDAP グループ dn をすべて大文字で指定します。
- グループ・メンバーシップの変更がすぐに許可に反映されません。影響を受けたユーザーがまだログオンしていると、この状態が発生する場合があります。ユーザーのグループ・メンバーシップは、そのユーザーのログイン・セッションにキャッシュされ、デフォルトで 2 時間後に有効期限が切れます。ログイン・セッションの有効期限が切れるのを待つか (デフォルトは 2 時間)、アプリケーション・サーバーを再始動します。Human Task Manager によって提供される最新表示の方式は、この動詞には適用できません。プロセス・スターターのグループ・メンバーシップ・リストは最新表示されないことに注意してください。

プロセス関連およびタスク関連の監査証跡情報の使用

ビジネス・プロセスおよびヒューマン・タスクのイベント・タイプとデータベース構造について説明します。

ビジネス・プロセス・コンテナーとタスク・コンテナーの一方または両方で、ロギングを使用可能にする必要があります。

ロギングが使用可能に設定されている場合には、ビジネス・プロセスまたはヒューマン・タスクの実行時に重要なステップが発生すると常に、情報が監査ログまたは Common Event Infrastructure (CEI) ログに書き込まれます。CEI について詳しくは、「*WebSphere Process Server* モニター」PDF を参照してください。 以降のトピックでは、ビジネス・プロセスおよびヒューマン・タスクのイベント・タイプとデータベース構造について説明します。

ビジネス・プロセスの監査イベント・タイプ

ここでは、ビジネス・プロセスの処理中に監査ログに書き込むことができるイベントのタイプについて説明します。

イベントをログに記録するには、次の条件を満たす必要があります。

- ビジネス・プロセス・コンテナー用の対応する監査ロギング・タイプが使用可能になっている
- プロセス・モデルの対応するエンティティー用にイベントが必ず使用可能になっている

次の表に、ビジネス・プロセスの実行中に発生する可能性がある監査イベントのコード一覧を示します。

表 44. プロセス・インスタンス・イベント

監査イベント	イベント・コード
PROCESS_STARTED	21000
PROCESS_SUSPENDED	21001
PROCESS_RESUMED	21002
PROCESS_COMPLETED	21004
PROCESS_TERMINATED	21005
PROCESS_RESTARTED	21019
PROCESS_DELETED	21020
PROCESS_FAILED	42001
PROCESS_COMPENSATING	42003
PROCESS_COMPENSATED	42004
PROCESS_TERMINATING	42009
PROCESS_FAILING	42010
PROCESS_CORRELATION_SET_INITIALIZED	42027
PROCESS_COMPENSATION_INDOUBT	42030
PROCESS_WORKITEM_DELETED	42041
PROCESS_WORKITEM_CREATED	42042
PROCESS_COMPENSATION_FAILED	42046

表 44. プロセス・インスタンス・イベント (続き)

監査イベント	イベント・コード
PROCESS_EVENT_RECEIVED	42047
PROCESS_EVENT_ESCALATED	42049
PROCESS_WORKITEM_TRANSFERRED	42056

表 45. アクティビティー・イベント

監査イベント	イベント・コード
ACTIVITY_READY	21006
ACTIVITY_STARTED	21007
ACTIVITY_COMPLETED	21011
ACTIVITY_CLAIM_CANCELED	21021
ACTIVITY_CLAIMED	21022
ACTIVITY_TERMINATED	21027
ACTIVITY_FAILED	21080
ACTIVITY_EXPIRED	21081
ACTIVITY_LOOPED	42002
ACTIVITY_SKIPPED	42005
ACTIVITY_TERMINATING	42008
ACTIVITY_FAILING	42011
ACTIVITY_OUTPUT_MESSAGE_SET	42012
ACTIVITY_FAULT_MESSAGE_SET	42013
ACTIVITY_STOPPED	42015
ACTIVITY_FORCE_RETRIED	42031
ACTIVITY_FORCE_COMPLETED	42032
ACTIVITY_UNDO_STARTED	42033
ACTIVITY_UNDO_SKIPPED	42034
ACTIVITY_UNDO_COMPLETED	42035
ACTIVITY_MESSAGE_RECEIVED	42036
ACTIVITY_LOOP_CONDITION_TRUE	42037
ACTIVITY_LOOP_CONDITION_FALSE	42038
ACTIVITY_WORKITEM_DELETED	42039
ACTIVITY_WORKITEM_CREATED	42040
ACTIVITY_ESCALATED	42050
ACTIVITY_WORKITEM_REFRESHED	42054
ACTIVITY_WORKITEM_TRANSFERRED	42055
ACTIVITY_PARALLEL_BRANCHES_STARTED	42057

表 46. 変数関連イベント

監査イベント	イベント・コード
VARIABLE_UPDATED	21090

表 47. 制御リンク・イベント

監査イベント	イベント・コード
LINK_EVALUATED_TO_TRUE	21034
LINK_EVALUATED_TO_FALSE	42000

表 48. プロセス・テンプレート・イベント

監査イベント	イベント・コード
PROCESS_INSTALLED	42006
PROCESS_UNINSTALLED	42007

表 49. スコープ・インスタンス・イベント

監査イベント	イベント・コード
SCOPE_STARTED	42020
SCOPE_SKIPPED	42021
SCOPE_FAILED	42022
SCOPE_FAILING	42023
SCOPE_TERMINATED	42024
SCOPE_COMPLETED	42026
SCOPE_COMPENSATING	42043
SCOPE_COMPENSATED	42044
SCOPE_COMPENSATION_FAILED	42045
SCOPE_EVENT_RECEIVED	42048
SCOPE_EVENT_ESCALATED	42051

ヒューマン・タスクの監査イベント・タイプ

ここでは、ヒューマン・タスクの処理中に監査ログに書き込むことができるイベントのタイプについて説明します。

イベントをログに記録するには、次の条件を満たす必要があります。

- ヒューマン・タスク・コンテナー用の対応する監査ロギング・タイプが使用可能になっている
- タスク・モデルの対応するエンティティー用にイベントが必ず使用可能になっている

次の表に、ヒューマン・タスクの実行中に発生する可能性がある監査イベントのコード一覧を示します。

表 50. タスク・インスタンス・イベント

監査イベント	イベント・コード
TASK_CREATED	51001
TASK_DELETED	51002
TASK_STARTED	51003
TASK_COMPLETED	51004
TASK_CLAIM_CANCELLED	51005

表 50. タスク・インスタンス・イベント (続き)

監査イベント	イベント・コード
TASK_CLAIMED	51006
TASK_TERMINATED	51007
TASK_FAILED	51008
TASK_EXPIRED	51009
TASK_WAITING_FOR_SUBTASK	51010
TASK_SUBTASKS_COMPLETED	51011
TASK_RESTARTED	51012
TASK_SUSPENDED	51013
TASK_RESUMED	51014
TASK_COMPLETED_WITH_FOLLOW_ON	51015
TASK_UPDATED	51101
TASK_OUTPUT_MESSAGE_UPDATED	51103
TASK_FAULT_MESSAGE_UPDATED	51104
TASK_WORKITEM_DELETED	51201
TASK_WORKITEM_CREATED	51202
TASK_WORKITEM_TRANSFERRED	51204
TASK_WORKITEM_REFRESHED	51205

表 51. タスク・テンプレート・イベント

監査イベント	イベント・コード
TASK_TEMPLATE_INSTALLED	52001
TASK_TEMPLATE_UNINSTALLED	52002

表 52. エスカレーション・インスタンス・イベント

監査イベント	イベント・コード
ESCALATION_FIRED	53001
ESCALATION_WORKITEM_DELETED	53201
ESCALATION_WORKITEM_CREATED	53202
ESCALATION_WORKITEM_TRANSFERRED	53204
ESCALATION_WORKITEM_REFRESHED	53205

ビジネス・プロセス用監査証跡データベース・ビューの構造

AUDIT_LOG_B データベース・ビューは、ビジネス・プロセスについての監査ログ情報を提供します。

監査証跡の内容を読み取るには、データベース表とビューの読み取りをサポートする SQL または他の管理ツールを使用します。

監査イベントは、プロセス・エンティティーと関連があります。監査イベントのタイプは、そのイベントが参照するエンティティーによって異なります。監査イベントには、次のタイプがあります。

- プロセス・テンプレート・イベント (PTE)
- プロセス・インスタンス・イベント (PIE)
- アクティビティー・インスタンス・イベント (AIE)
- 変数関連イベント (VAR)
- 制御リンク・イベント (CLE)
- スコープ関連イベント (SIE)

監査イベント・タイプのコード一覧については、401 ページの『ビジネス・プロセスの監査イベント・タイプ』を参照してください。

以下の表に、AUDIT_LOG_B 監査証跡ビューの構造を示します。表では、列名とイベント・タイプをリストし、列について簡単に説明しています。

インライン・タスクは、AUDIT_LOG_B 監査証跡ビューに記録され、TASK_LOG 監査証跡ビューには記録されません。例えば、インライン参加タスクを要求すると ACTIVITY_CLAIMED イベントが生成されますが、タスク関連イベントは生成されません。

表 53. AUDIT_LOG_B 監査証跡ビューの構造

名前	PTE	PIE	AIE	VAR	CLE	SIE	説明
AIID			x				現行イベントに関連したアクティビティー・インスタンスの ID。
ALID	x	x	x	x	x	x	監査ログ・エントリーの ID。
EVENT_TIME	x	x	x	x	x	x	イベントが発生したときのタイム・スタンプ (協定世界時 (UTC) 形式)。
EVENT_TIME_UTC	x	x	x	x	x	x	イベントが発生したときのタイム・スタンプ (協定世界時 (UTC) 形式)。
AUDIT_EVENT	x	x	x	x	x	x	発生したイベントのタイプ。
PTID	x	x	x	x	x	x	現行イベントに関連したプロセスのプロセス・テンプレート ID。
PIID		x	x	x	x	x	現行イベントに関連したプロセス・インスタンスのプロセス・インスタンス ID。
VARIABLE_NAME				x			現行イベントに関連した変数の名前。
SIID						x	イベントに関連したスコープ・インスタンスの ID。
PROCESS_TEMPL _NAME	x	x	x	x	x	x	現行イベントに関連したプロセス・テンプレートのプロセス・テンプレート名。
TOP_LEVEL_PIID		x	x	x	x	x	現行イベントに関連したトップレベル・プロセスの ID。
PARENT_PIID		x	x	x	x	x	親プロセスのプロセス・インスタンス ID。親が存在しない場合は NULL。
VALID_FROM	x	x	x	x	x	x	現行イベントに関連したプロセス・テンプレートの有効開始日付。
VALID_FROM_ UTC	x	x	x	x	x	x	現行イベントに関連したプロセス・テンプレートの有効開始日付 (協定世界時 (UTC) 形式)。

表 53. AUDIT_LOG_B 監査証跡ビューの構造 (続き)

名前	PTE	PIE	AIE	VAR	CLE	SIE	説明
ATID			x				現行イベントに関連したアクティビティー・テンプレートの ID。
ACTIVITY_NAME			x			x	イベントが発生したアクティビティー名。
ACTIVITY_KIND			x				イベントが発生したときのアクティビティーの種類。考えられる値は次のとおりです。 KIND_EMPTY 3 KIND_INVOKE 21 KIND_RECEIVE 23 KIND_REPLY 24 KIND_THROW 25 KIND_TERMINATE 26 KIND_WAIT 27 KIND_COMPENSATE 29 KIND_SEQUENCE 30 KIND_SWITCH 32 KIND_WHILE 34 KIND_PICK 36 KIND_FLOW 38 KIND_SCRIPT 42 KIND_STAFF 43 KIND_ASSIGN 44 KIND_CUSTOM 45 KIND_RETHROW 46 KIND_FOR_EACH_SERIAL 47 KIND_FOR_EACH_PARALLEL 49 これらは、ActivityInstanceData.KIND_* で定義される定数です。

表 53. AUDIT_LOG_B 監査証跡ビューの構造 (続き)

名前	PTE	PIE	AIE	VAR	CLE	SIE	説明
ACTIVITY_STATE			x				イベントに関連したアクティビティーの状態。考えられる値は次のとおりです。 STATE_INACTIVE 1 STATE_READY 2 STATE_RUNNING 3 STATE_SKIPPED 4 STATE_FINISHED 5 STATE_FAILED 6 STATE_TERMINATED 7 STATE_CLAIMED 8 STATE_TERMINATING 9 STATE_FAILING 10 STATE_WAITING 11 STATE_EXPIRED 12 STATE_STOPPED 13 これらは、ActivityInstanceData.STATE_* で定義される定数です。
CONTROL_LINK_NAME					x		現行リンク・イベントに関連したリンクの名前。
PRINCIPAL		x	x	x	x	x	プリンシパルの名前。PROCESS_DELETED イベントの場合には設定されません。
VARIABLE_DATA				x			variable updatedイベント用の変数のデータ。
EXCEPTION_TEXT		x	x			x	アクティビティーまたはプロセスが失敗する原因となった例外メッセージ。次の場合に適用されます。 PROCESS_FAILED ACTIVITY_FAILED SCOPE_FAILED
DESCRIPTION		x	x	x	x	x	潜在的に解決される可能性のある置換変数を含むアクティビティーまたはプロセスの説明。
CORR_SET_INFO		x					プロセスの開始時に初期化された相関セットのストリング表現。processCorrelationSetInitialized イベント (42027) により出力されます。

表 53. AUDIT_LOG_B 監査証跡ビューの構造 (続き)

名前	PTE	PIE	AIE	VAR	CLE	SIE	説明
USER_NAME		x	x				作業項目が変更されたユーザーの名前。次のイベントの場合に適用されます。 • プロセス・インスタンスの作業項目が削除された • アクティビティー・インスタンスの作業項目が削除された • プロセス・インスタンスの作業項目が作成された • アクティビティー・インスタンスの作業項目が作成された
ADDITIONAL_ INFO		x	x			x	このフィールドの内容は、以下のイベントのタイプによって決まります。 **ACTIVITY_WORKITEM_ TRANSFERRED、 PROCESS_WORK_ITEM_ TRANSFERRED** 作業項目を受け取ったユーザーの名前。 **ACTIVITY_WORKITEM_ CREATED、 ACTIVITY_WORKITEM_ REFRESHED、 ACTIVITY_ESCALATED** 作業項目を作成または更新する対象となったユーザーすべてのリスト (「,」区切り)。リストのユーザーが 1 人のみの場合は、USER_NAME フィールドにこのユーザーのユーザー名が入力され、ADDITIONAL_INFO フィールドは空 (NULL) になります。 **PROCESS_EVENT_RECEIVED、 SCOPE_EVENT_RECEIVED** 選択可能な場合は、イベント・ハンドラーが受信したオペレーションのタイプ。使用形式は次のとおりです。'{' ポート・タイプ・ネーム・スペース '}' ポート・タイプ名 ':' オペレーション名。このフィールドは、「onAlarm」イベントの場合には設定されません。

ヒューマン・タスク用監査証跡データベース・ビューの構造

TASK_AUDIT_LOG データベース・ビューは、ヒューマン・タスクについての監査ログ情報を提供します。

インライン・タスクは AUDIT_LOG_B ビューに記録されます。他のすべてのタスク・タイプは TASK_AUDIT_LOG ビューに記録されます。

監査証跡の内容を読み取るには、データベース表とビューの読み取りをサポートする SQL または他の管理ツールを使用します。

監査イベントは、タスク・エンティティーと関連があります。監査イベントのタイプは、そのイベントが参照するエンティティーによって異なります。監査イベントには、次のタイプがあります。

- タスク・インスタンス・イベント (TIE)
- タスク・テンプレート・イベント (TTE)
- エスカレーション・インスタンス・イベント (EIE)

以下の表に、TASK_AUDIT_LOG 監査証跡ビューの構造を示します。表では、列名とイベント・タイプをリストし、列について簡単に説明しています。

インライン・タスクは、AUDIT_LOG_B 監査証跡ビューに記録され、TASK_AUDIT_LOG 監査証跡ビューには記録されません。例えば、インライン参加タスクを要求すると ACTIVITY_CLAIMED イベントが生成されますが、タスク関連イベントは生成されません。

表 54. TASK_AUDIT_LOG 監査証跡ビューの構造

名前	TIE	TTE	EIE	説明
ALID	x	x	x	監査ログ・エントリーの ID。
AUDIT_EVENT	x	x	x	発生したイベントのタイプ。監査イベント・コードの一覧については、403 ページの『ヒューマン・タスクの監査イベント・タイプ』を参照してください。
CONTAINMENT_ CTX_ID	x	x		収容コンテキストの ID (ACOID、PTID、または PIID など)。
DESCRIPTION	x		x	解決された記述ストリング。記述のプレースホルダーは現行値によって置き換えられます。影響を受けたすべての言語は、XML 文書としてフォーマット設定されて、この列に一緒に記録されます。 create-like イベントのプレースホルダーを含む記述を持つか、update-like イベント用に明示的に更新された言語のみが記録されます。
ESIID			x	現行イベントに関連したエスカレーション・インスタンスの ID。
ESTID			x	現行イベントに関連したエスカレーション・テンプレートの ID。
EVENT_TIME	x	x	x	イベントが発生したときの時刻 (協定世界時 (UTC) 形式)。

表 54. TASK_AUDIT_LOG 監査証跡ビューの構造 (続き)

名前	TIE	TTE	EIE	説明
FAULT_NAME	x			障害メッセージの名前。この属性は、次のイベントの場合に適用されます。 TASK_FAILED TASK_FAULT_MESSAGE_UPDATED
FAULT_NAME_SPACE	x			障害メッセージ・タイプのネーム・スペース。この属性は、次のイベントの場合に適用されます。 TASK_FAILED TASK_FAULT_MESSAGE_UPDATED
FOLLOW_ON_TKIID	x			後続のタスク・インスタンスの ID。
MESSAGE_DATA	x			新規に作成または更新された入力、出力、または障害メッセージの内容。
NAME	x	x	x	イベントに関連付けられたタスク・インスタンス、タスク・テンプレート、またはエスカレーション・インスタンスの名前。
NAMESPACE	x	x		イベントに関連付けられたタスク・インスタンス、タスク・テンプレート、またはエスカレーション・インスタンスのネーム・スペース。
NEW_USER				転送または作成された作業項目の新規所有者。USERS フィールドによってこの値が使用可能になる場合、この値は `null` になることがあります。フィールド USERS も参照してください。この属性は次のイベントに適用されます。
	x			TASK_WORKITEM_CREATED
	x			TASK_WORKITEM_TRANSFERRED
			x	ESCALATION_WORKITEM_CREATED
			x	ESCALATION_WORKITEM_TRANSFERRED
OLD_USER				転送済み作業項目の前の所有者。この属性は、次のイベントの場合に適用されます。
	x			TASK_WORKITEM_TRANSFERRED
	x			TASK_WORKITEM_DELETED
			x	ESCALATION_WORKITEM_TRANSFERRED
			x	ESCALATION_WORKITEM_DELETED
PARENT_CONTEXT_ID	x			タスクの親コンテキスト (例えば、アクティビティー・テンプレートやタスク・インスタンス) の ID。サブタスクおよび後続タスクの場合にのみ設定されます。
PARENT_TASK_NAME	x			親タスク・インスタンスまたはテンプレートの名前。サブタスクおよび後続タスクの場合にのみ設定されます。
PARENT_TASK_NAMESP	x			親タスク・インスタンスまたはテンプレートのネーム・スペース。サブタスクおよび後続タスクの場合にのみ設定されます。
PARENT_TKIID	x			親タスク・インスタンスの ID。

表 54. TASK_AUDIT_LOG 監査証跡ビューの構造 (続き)

名前	TIE	TTE	EIE	説明
PRINCIPAL	x	x	x	出したリクエストによってイベントがトリガーされたプリンシパルの名前。
TASK_KIND	x	x		タスクの種類。考えられる値は次のとおりです。 KIND_HUMAN 101 KIND_ORIGINATING 103 KIND_PARTICIPATING 105 KIND_ADMINISTRATIVE 106
TASK_STATE	x			タスクまたはタスク・テンプレートの状態。タスク・テンプレートの場合に考えられる値は次のとおりです。 STATE_STARTED 1 STATE_STOPPED 2 タスク・インスタンスの場合に考えられる値は次のとおりです。 '1' :'STATE_INACTIVE' '2' :'STATE_READY' '3' :'STATE_RUNNING' '5' :'STATE_FINISHED' '6' :'STATE_FAILED' '7' :'STATE_TERMINATED' '8' :'STATE_CLAIMED' '12' :'STATE_EXPIRED' '101':'FORWARDED'
TKIID	x		x	タスク・インスタンスの ID。
TKTID	x	x		タスク・テンプレートの ID。
TOP_TKIID	x			トップ・タスク・インスタンスの ID。
USERS	x		x	タスクまたはエスカレーション作業項目に割り当てられた新規ユーザー ID。NEW_USER フィールドによってこの値が使用可能になる場合、この値は null になることがあります。この属性が適用されるイベントのリストについては、フィールド NEW_USER を参照してください。
VALID_FROM		x		現行イベントに関連したタスク・テンプレートの有効開始日付。

表 54. TASK_AUDIT_LOG 監査証跡ビューの構造 (続き)

名前	TIE	TTE	EIE	説明
WORK_ITEM_REASON	x		x	作業項目の割り当て理由。考えられる値は次のとおりです。 POTENTIAL_OWNER 1 EDITOR 2 READER 3 OWNER 4 POTENTIAL_STARTER 5 STARTER 6 ADMINISTRATOR 7 POTENTIAL_SENDER 8 ORIGINATOR 9 ESCALATION_RECEIVER 10 POTENTIAL_INSTANCE_CREATOR 11 理由は、作業項目に関連したすべてのイベントの場合に設定されます。例えば、ESCALATION_RECEIVER はエスカレーション作業項目に関連したイベントの場合に設定される一方で、他の理由はタスク作業項目関連のイベントに適用されます。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-8711
東京都港区六本木 3-2-12
IBM World Trade Asia Corporation
Intellectual Property Law & Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_. All rights reserved.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

プログラミング・インターフェース情報は、プログラムを使用してアプリケーション・ソフトウェアを作成する際に役立ちます。

一般使用プログラミング・インターフェースにより、お客様はこのプログラム・ツール・サービスを含むアプリケーション・ソフトウェアを書くことができます。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

警告: 診断、修正、調整情報は、変更される場合がありますので、プログラミング・インターフェースとしては使用しないでください。

商標

IBM および関連の商標については、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

この製品には、Eclipse Project (http://www.eclipse.org/) により開発されたソフトウェアが含まれています。

WebSphere Process Server for z/OS バージョン 6.0.2

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12