WebSphere® WebSphere Process Server for z/OS
バージョン 7.0.0

Business Process Choreographer

IBM®

WebSphere® WebSphere Process Server for z/OS
バージョン 7.0.0

Business Process Choreographer

IBM®

新しい版で明記されるまで、WebSphere Process Server for z/OS バージョン 7、リリース 0、モディフィケーション 0 (製品番号 5655-N53) 以降のすべてのリリースとモディフィケーションが本書の対象となります。

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
Business Process Choreographer
Version 7.0.0

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2010.4

目次

第 1 部 WebSphere Process Server でのビジネス・プロセスおよびヒューマン・タスク

第 1 章 ビジネス・プロセスの概要

ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、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 (例えば、assign または invoke 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 (例えば、sequence または while 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外部パートナーとの対話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。
- プロセスと交換され、アクティビティー間で受け渡されるデータを保管する変数。
- 複数のサービス対話を同じビジネス・プロセス・インスタンスに相互に関連付けるために使用される相関セット。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処理実行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、アクティビティー・グループ、またはスコ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 の仕様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。これらのインライン予定タスクは、フォームの入力、文書の承認などの、人が関わるビジネス・プロセスに参加できます。
- 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、すべての BPEL 変数、相関プロパティー、パートナー・リンク、およびプロセス・コンテキストやアクティビティー・コンテキストにアクセスすることができます。

- WebSphere® Information Server またはリレーショナル・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。
- プロセスのバージョン管理のための有効開始タイム・スタンプ。
- ビジネス・プロセス内のトランザクション境界を手動で設定または制御するための拡張。
- アクティビティーのタイムアウト。

プロセス・テンプレート

プロセス・テンプレートとは、ランタイム環境にデプロイされ、インストールされるプロセス定義です。

プロセス・プロパティーは、プロセスの定義時に指定されます。ランタイム環境では、ランタイム・データベースにプロセス・テンプレートのプロパティーが格納されます。それらのプロパティーにアクセスするには、Business Process Choreographer データベースのビュー (PROCESS_TEMPLATE ビューなど) または照会テーブルを使用します。

さらに、インストール済みのビジネス・プロセスは、以下のいずれかの状態も保持することができます。

開始 プロセス・テンプレートが作成され、開始されると、そのテンプレートの新規インスタンスを開始できます。

停止 プロセス・テンプレートが停止状態にある場合は、このテンプレートの新規インスタンスを作成して開始することはできません。テンプレートの既存のインスタンスは、完了するまで継続して実行されます。

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。
- 複数のトランザクション内で実行されます。
- サービスと同期的および非同期的に対話します。
- プロセスの状態はランタイム・データベースに保管されるため、プロセスの順方向リカバリーが可能です。

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、割り込み不可能なビジネス・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・トランザクションまたはアクティビティー・セッションのいずれかの作業単位に参加します。

Microflow には、次の特性があります。
- 1 つのトランザクションまたはアクティビティー・セッションの中で実行されます。
- 通常は短時間で実行します。
- 状態は一時的であるため、ランタイム・データベースに保管されません。
- 通常はサービスを同期的に呼び出します。
- 割り込み不可能な子プロセスのみを持つことができます。
- 以下を含むことはできません。
  - ヒューマン・タスク
  - wait アクティビティー
  - 開始していない receive アクティビティーまたは pick アクティビティー

関連概念

27 ページの『ビジネス・プロセスの対話に影響を及ぼす要因』
各種の呼び出しシナリオにおけるビジネス・プロセスの振る舞いは、いくつかの要因から影響を受けます。これには、対話スタイル、ビジネス・プロセスのタイプ、操作タイプ、およびサービス・エンドポイントの解決が含まれます。

30 ページの『ビジネス・プロセスのトランザクションの振る舞い』
ビジネス・プロセスは、トランザクションの一部として実行されます。ビジネス・プロセスのナビゲーションは、長期実行プロセスの場合には複数のトランザクションにまたがり、Microflow の場合には 1 つのトランザクションの一部となります。このようなナビゲーション・トランザクションは、外部要求、内部メッセージ、または非同期サービスからの応答によってトリガーされます。トランザクションが開始すると、プロセス定義に従って必要なアクティビティーが実行されます。呼び出されたサービスは、トランザクションに参加できます。

プロセス・インスタンス

プロセス・インスタンスは、プロセス・テンプレートをインスタンス化したものです。

Web Services Business Process Execution Language (WS-BPEL) で定義されているビジネス・プロセスは、ステートフル Web サービスを表しており、そのようなサービスとして長期間にわたって他の Web サービスと対話することができます。BPEL プロセスが開始するときにはいつでも、そのプロセスの新規インスタンスが作成され、他のビジネス・パートナーと通信が可能になります。インスタンスの最後のアクティビティーが完了するか、terminate アクティビティーが実行されるか、プロセスによって処理されない障害が発生すると、インスタンスは完了します。

多くのプロセス・インスタンス・プロパティーは、対応するプロセス・テンプレートから継承されます。プロセス・インスタンスの状態など、それ以外のものは、プロセス・インスタンスの存続時間中に割り当てと変更が行われます。それらのプロパティーはすべてランタイム・データベースに格納されます。それらのプロパティ

ーにアクセスするには、Business Process Choreographer データベースのビュー (PROCESS_INSTANCE ビューや QUERY_PROPERTY ビューなど) または照会テーブルを使用します。

プロセス・バージョン管理の概要

ビジネス・プロセスは、時間と共に進化していきます。ビジネス・プロセスは、変化する環境およびビジネス上のニーズを反映する必要があります。これらの変更は、規制の変更やビジネス・プロセスの最適化などの、ビジネス主導の変更である場合があります。ビジネス・プロセス・アプリケーションには、長期実行インスタンスを含めることができます。これらのインスタンスは、数週間か数カ月にわたって実行されることも、数年にわたって実行されることもあります。この特性のため、新しいバージョンのビジネス・プロセスの導入には、特定の要件が課されます。

プロセスの新しいバージョンを作成するには、WebSphere Integration Developer で、モジュールを作成します。このモジュールには、元のプロセスのコピーをベースとした、プロセスの新しいバージョンが含まれます。プロセスの新しいバージョンに対して必要な変更を行い、新しいバージョンに有効開始日を指定してから、このモジュールをランタイム環境にデプロイします。既存のプロセス・インスタンスを新しいバージョンにマイグレーションできるようにしたい場合は、WebSphere Integration Developer でプロセス・マイグレーション仕様を定義し、プロセスの新しいバージョンと一緒にデプロイする必要もあります。

プロセスへの変更の性質に応じて、既存のプロセス・インスタンスは、作成時および開始時に使用していたバージョンを使用して完了しなければならないことがあります。一方で、新規インスタンスは、新しいバージョンのビジネス・プロセスから作成する必要があります。状況によっては (例えばプロセス内のエラーを訂正したい場合など)、プロセスの実行中のインスタンスを新しいバージョンにマイグレーションして、それらがこのバージョンを使用して完了するようにしたい場合もあります。

関連概念

26 ページの『ビジネス・プロセスの呼び出しシナリオ』
ビジネス・プロセスは、Service Component Architecture (SCA) コンポーネントの実装です。ビジネス・プロセスは、サービスを他のパートナーに公開したり、他のパートナーから提供されるサービスを消費したりすることができます。ビジネス・プロセスは、Business Process Choreographer API で使用可能なサービス・プロバイダー、他の SCA サービス・コンポーネントの SCA サービス・プロバイダー、または他の SCA サービス・コンポーネント (他のビジネス・プロセスを含む) を呼び出す SCA クライアントのいずれかです。

プロセス・バージョン管理: ビジネス・プロセスの異なるバージョンの呼び出し

WebSphere Integration Developer でビジネス・プロセスを定義するときに、有効開始日などのバージョン管理情報を含めることができます。実行時に、現在有効なバージョンのプロセスを動的に呼び出すか、特定のバージョンのプロセスを呼び出すことができます。

プロセスのバージョンは、その名前と有効開始日によって決まります。つまり、同じプロセスの異なるバージョンに同じプロセス名を付けることはできますが、それぞれの有効開始日は異なっています。プロセス・インスタンスを呼び出したときに使用されるプロセスのバージョンは、そのプロセスが*早期*バインディング・シナリオと*実行時*バインディング・シナリオのどちらで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・シナリオでは、呼び出されるプロセスのバージョンは、モデリング中かまたはプロセスのデプロイ時に決定されます。呼び出し元は、静的にバインドされている専用のプロセスを呼び出します。異なるバージョンの有効開始日に照らして有効な、プロセスの別のバージョンが存在していても、静的に結合されている現行のプロセスが呼び出され、その他のバージョンはすべて無視されます。

早期バインディングの例は SCA ワイヤーです。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・シナリオでは、呼び出されるプロセス・テンプレートは、呼び出し側がそのプロセスを呼び出したときに決定されます。この場合は、現行で有効なプロセスのバージョンが使用されます。現在有効なプロセスのバージョンが、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、それらのインスタンスが新しいバージョンにマイグレーションされていなければ、開始時に関連付けられたプロセス・テンプレートを使用して、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。

- 新規プロセス・インスタンスで使用される現在有効なプロセス・テンプレート
- 有効ではなくなったが、既存の長期実行プロセス・インスタンスにはまだ使用できるプロセス・テンプレート
- 有効開始日に従って将来的に有効になるプロセス・テンプレート

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通常、現在有効になっているバージョンのプロセスに基づいており、有効開始日が将来の日付ではありません。

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。

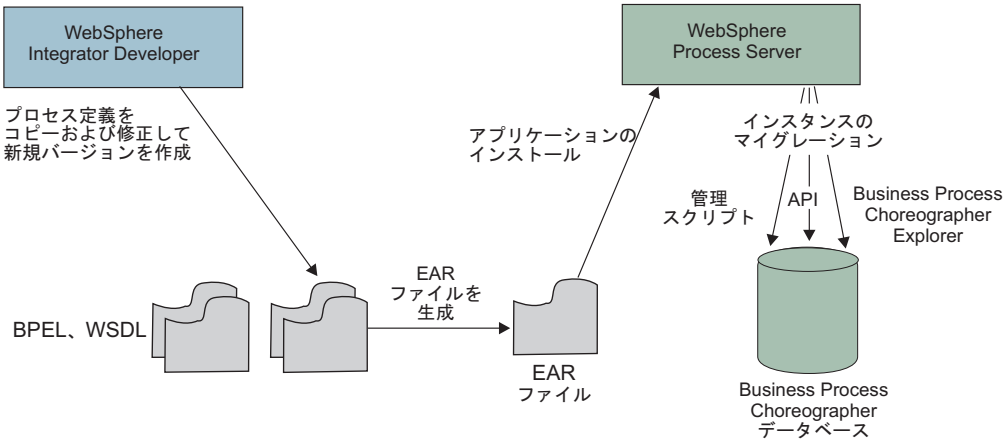
プロセス・バージョン管理: 実行中のプロセス・インスタンスを新しいバージョンのビジネス・プロセスにマイグレーションする

新しいバージョンのプロセスを導入するとき、このバージョンを、新規プロセス・インスタンスと既に開始したインスタンスの両方に適用したい場合があります。こ

のことは、プロセスへの変更が頻繁に必要となるが、個々のプロセス・インスタンスは比較的存続時間が長くなる可能性がある環境で、重要になります。このような場合、実行中のプロセス・インスタンスを、新しいバージョンにマイグレーションする必要があります。

次の図に、WebSphere Integration Developer でプロセスの新バージョンを定義して、WebSphere Process Server で実行中のインスタンスをマイグレーションするまでの手順を示します。

注: プロセスの新バージョンをマイグレーション・ターゲットにするには、そのプロセスの新バージョンとともにマイグレーション指定をデプロイする必要があります。



実行中のプロセス・インスタンスを、新しいバージョンのプロセスへマイグレーションするには、管理スクリプトを使用してインスタンスを一括してマイグレーションするか、Business Process Choreographer Explorer を使用して特定のインスタンスをマイグレーションします。 プロセス・インスタンスのマイグレーションを行うと、プロセス・ナビゲーションの現在位置にあるプロセス、変数、およびアクティビティーは、新しいバージョンのビジネス・プロセスを参照するようになり、後続のナビゲーションは、新しいバージョンのビジネス・プロセスのロジックに依存します。プロセス・インスタンスがマイグレーションされるときに既にナビゲート済みのアクティビティーは、マイグレーションされません。プロセス・インスタンスのマイグレーション中に、このプロセス・インスタンスに属していてまだ終了状態に達していないインライン・ヒューマン・タスクのすべてのインスタンスも、マイグレーションされます。

新しいバージョンのプロセスに含まれるプロセスへの変更が、アクティビティーの表示名または説明などのプロセス・ロジックに影響しない場合は、プロセス・ナビゲーション中に、実行中のプロセス・インスタンスをいつでもマイグレーションできます。ただし、変更によって、新規アクティビティー、変数、または条件式などのプロセス・ロジックに影響する場合、プロセスのロジックに影響するすべての変更が、プロセス・ナビゲーションの現在位置よりも後にある場合のみ、実行中のプロセス・インスタンスを新規バージョンにマイグレーションすることができます。プロセスの新しいバージョンに対して行うことができる変更について詳しくは、この技術情報を参照してください。

インライン・ヒューマン・タスクへの変更はすべて、プロセスのロジックに影響するものと見なされます。変更されたインライン・ヒューマン・タスクがすべてプロセス・ナビゲーションの現在位置よりも後にある場合のみ、実行中のプロセス・インスタンスを新しいバージョンにマイグレーションできます。

ビジネス・プロセスにイベントが定義されている場合、CBE を使用してプロセス・マイグレーションを追跡できます。マイグレーションの開始時にイベントを生成し、マイグレーションの完了時に再度イベントを生成することができます。プロセス・マイグレーションの履歴は、Business Process Choreographer データベースでも保持されます。

例

プロセス・インスタンスのマイグレーション例についてひととおり作業するには、Business Process Management Samples and Tutorials の Web サイトへ移動してください。

相関セット

相関は、複数の長期間実行されるメッセージ交換 (通常 BPEL プロセスとそのパートナー・サービスとの間で行われる) の追跡に使用されます。相関セットにより、メッセージ本文の内容に基づいてメッセージを適切なプロセス・インスタンスに経路指定することができます。これにより、プロセス・インスタンスはパートナー・サービスとの会話を維持できます。

相関セットは、WSDL ファイルで定義された 1 つ以上のプロパティーで構成されています。プロパティー別名は、データをメッセージから相関プロパティーにマップする方法を、Business Process Choreographer に指示するルールです。
invoke、receive、pick、および reply アクティビティーで相関セットを使用することで、送受信されるメッセージでどの相関セットが出現するかを指示することができます。各相関セットの値は、プロセス・インスタンスを一意的に識別します。このことは、プロセス・インスタンスが既に終了状態 (完了状態など) に達した場合にも当てはまります。

プロセスが複数の receive または pick アクティビティーで構成されている場合、相関セットは必須です。新規プロセス・インスタンスを開始する receive または pick アクティビティーでは、必ずしも相関セットを必要としません。ただし、残りの receive または pick アクティビティーでは、メッセージの送付先のプロセス・インスタンスを一意的に識別するために、相関セットが必要です。

プロセス・ライフ・サイクル

プロセスを開始すると、ビジネス・プロセス・インスタンスのナビゲーションが始まり、環境との対話が始まります。つまり、特定の対話は特定のプロセス状態でのみ可能であり、これらの対話が今度はプロセス・インスタンスの状態に影響を与えます。

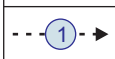
プロセス・インスタンスの状態遷移図

プロセスは、プロセス・インスタンスのライフ・サイクル中に重要なイベントが生じるたびに状態が変化します。例えば、API 要求によって、実行状態のプロセスが

中断状態にされる、などです。状態遷移図には、プロセスのライフ・サイクル中に発生する可能性がある状態遷移が示されます。Microflow と長期実行プロセスでは、状態遷移図が異なります。

以下の図で使用される規則

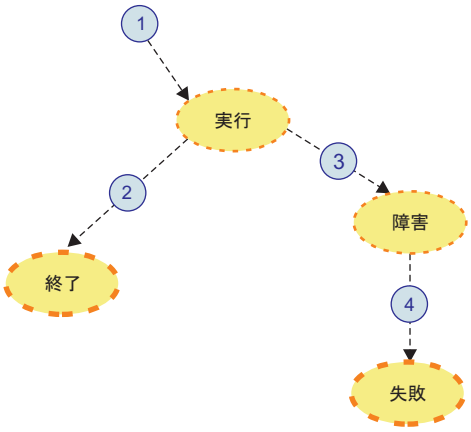
図の中の状態遷移は、数字で示されます。これらの数字は、付随するテキスト内で説明されます。また、図には以下のタイプのシンボルが含まれます。

シンボル	説明
	過渡状態。このような状態は表示されません。
	永続状態。
	過渡の終了状態。
	永続の終了状態。
	Business Flow Manager によって自動的にトリガーされる状態遷移。
	API を使用した外部対話の結果である状態遷移。
	Business Flow Manager によって制御される状態遷移、または API を使用した外部対話の結果である状態遷移。

Microflow インスタンスの状態遷移図

Microflow はステートレスと見なされます。これは、プロセスが常にトランザクション内で実行され、インスタンス情報がプロセス・インスタンスのナビゲート用に保持されないためです。ただし、プロセス定義、および Business Flow Manager の構成方法によっては、Microflow の状態を Common Base Event または監査ログ内に公開することができます。

次の図は、Microflow インスタンスの状態を示しています。

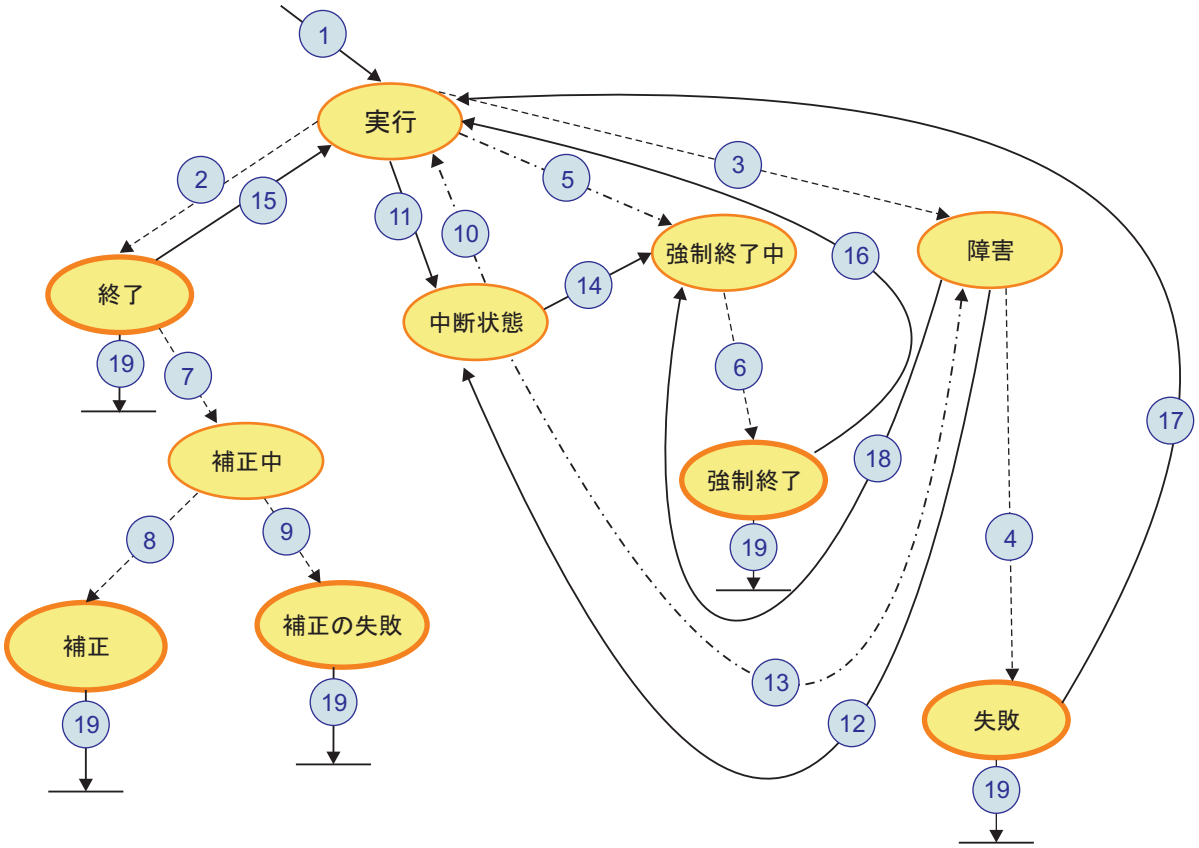


プロセス・インスタンスの正常な開始後、プロセス・インスタンスが到達する最初のプロセス状態は実行状態です (1)。プロセス・インスタンスが通常どおり最後まで実行されて完了すると、プロセス状態は実行から完了に変化します (2)。障害がプロセス境界に到達すると、プロセスは障害状態になります (3)。障害ハンドラーの実行中は、プロセスは障害状態のままです。このあと、プロセス・インスタンスは失敗状態になります (4)。

これらのすべての状態遷移は、Business Flow Manager によってトリガーされます。Microflow が開始すると、これらの自動ステップを操作することはできません。

長期実行プロセス・インスタンスの状態遷移図

長期実行プロセスは、複数のトランザクション内で実行されます。長期実行プロセスの状態は持続するため、目で確認できます。次の図は、長期実行プロセス・インスタンスで生じる可能性がある状態遷移を示しています。



実行状態、終了状態、障害状態、失敗状態、およびそれらの間の状態遷移は、Microflow のものと同じです。

プロセス・インスタンスは、外部要求または強制終了アクティビティーのいずれかによって強制終了されます。プロセス・インスタンスの終了は複数のナビゲーション・ステップで構成されることがあり、例えば長期実行アクティビティーまたはサブプロセスを強制終了するために、複数のチェーニングされたトランザクションになる可能性があります。この強制終了フェーズ時に、プロセス・インスタンスは強制終了中状態になります (5)、(14)、(18)。プロセスの長期実行部分がすべて終了すると、プロセス・インスタンスの状態も強制終了に変わります (6)。

子プロセスが正常に終了し、親プロセスがその後に失敗する場合は、子プロセスを補正できます。補正される間、子プロセスは補正中状態になります (7)。補正が正常に終了すると、子プロセスは補正済み状態になります (8)。補正が正常に終了しない場合は、子プロセスが補正の失敗状態になります (9)。これらの状態トランザクションは、親プロセスによって自動的に開始されます。

プロセス・インスタンスのナビゲーションがまだアクティブである場合、つまり実行状態または障害状態である場合、API 要求を使用して中断状態にすることができます。この状態は、指定した時間の経過後、または再開要求によって、再アクティブ化できます。プロセスの状態は、中断要求によって実行または障害から中断に (11)、(12)、再開要求によって中断から実行または障害に (10)、(13) 変化します。中断状態のプロセスも強制終了される場合があります (14)。最上位プロセス・インスタンスのみを中断および再開できます。ただし、中断または再開状態は子プロセスに伝搬されます。

プロセスが終了状態のいずれか (終了、強制終了、または失敗) に到達した場合は、再始動 API 要求によってもう一度開始できます (15)、(16)、(17)。最上位プロセス・インスタンスのみを再開できる一方で、補正できるのは子プロセス・インスタンスのみです。

プロセス・インスタンスは、終了状態に到達すると削除できます (19)。プロセスは、「**完了時に自動的に削除する (automatically delete on completion)**」属性を設定して自動的に削除することも、明示的な削除要求によって削除をトリガーすることもできます。

関連概念

30 ページの『ビジネス・プロセスのトランザクションの振る舞い』
ビジネス・プロセスは、トランザクションの一部として実行されます。ビジネス・プロセスのナビゲーションは、長期実行プロセスの場合には複数のトランザクションにまたがり、Microflow の場合には 1 つのトランザクションの一部となります。このようなナビゲーション・トランザクションは、外部要求、内部メッセージ、または非同期サービスからの応答によってトリガーされます。トランザクションが開始すると、プロセス定義に従って必要なアクティビティーが実行されます。呼び出されたサービスは、トランザクションに参加できます。

アクティビティーの状態遷移図

アクティビティー・インスタンスの実行中に重要なステップが発生すると、アクティビティー・インスタンスの状態は変化します。状態および状態遷移はアクティビティーのタイプによって異なります。

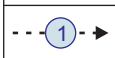
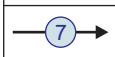
状態および状態遷移は、基本アクティビティーのライフ・サイクルで重要です。基本アクティビティーは以下のアクティビティー・タイプにグループ化されます。状態遷移図はアクティビティー・タイプによって異なります。

- short-lived アクティビティー (assign、empty、reply、rethrow、throw、terminate、および Java 断片アクティビティーなど)
- 外部イベントを待機するアクティビティー (receive および wait アクティビティーなど)
- pick (receive choice) アクティビティー
- invoke アクティビティー
- 単一または順次所有権を持つタスクのヒューマン・タスク・アクティビティー
- 並列所有権を持つタスクのヒューマン・タスク・アクティビティー

プロセス・インスタンスの状態遷移と対照的に、アクティビティーの終了状態は明示的に公開されません。アクティビティーのライフ・サイクルは、エンクロージング・プロセスによって異なります。アクティビティーは常にプロセス・インスタンスと共に削除されます。

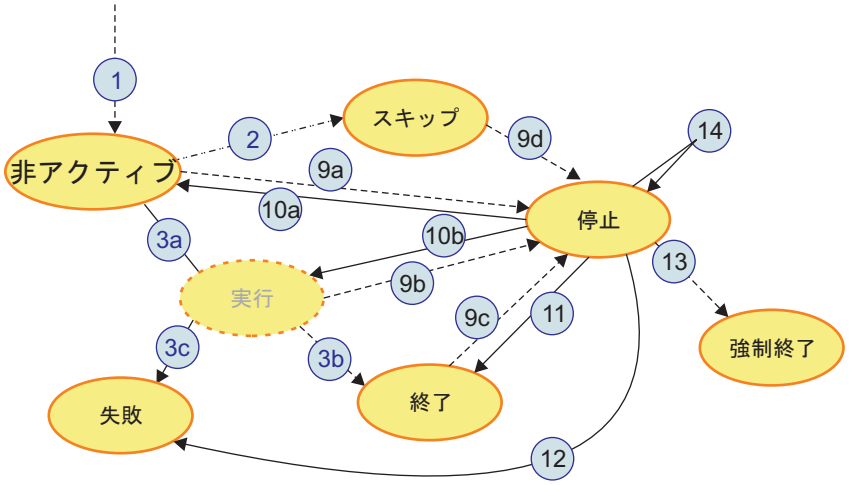
以下の図で使用される規則

図の中の状態遷移は、数字で示されます。これらの数字は、付随するテキスト内で説明されます。また、図には以下のタイプのシンボルが含まれます。

シンボル	説明
	過渡状態。これらの状態は表示されません。
	永続状態。
	Business Flow Manager によって自動的にトリガーされる状態遷移。
	ユーザー対話 (例えば、API 要求によるもの) の結果である状態遷移。
	Business Flow Manager またはユーザー対話によって制御される状態遷移。

short-lived アクティビティー・タイプの状態遷移図

次の状態遷移図は、単純な short-lived アクティビティー・タイプ (assign、 empty、 reply、 rethrow、 throw、 terminate、および Java 断片アクティビティーなど) の状態および状態遷移を示しています。ここでは、非アクティブ、スキップ、終了、失敗、停止および強制終了という状態を紹介します。これらの状態は、すべての基本アクティビティー・タイプに共通しています。



アクティビティーが作成された後は、非アクティブ状態になります (1)。フロー内の囲まれたアクティビティーは、複数の着信リンクおよび結合条件を持つ可能性があります。 そうしたアクティビティーが開始するには、すべての着信リンクをナビゲートする必要があります。アクティビティーの **suppressJoinFailure** 属性および結合条件の評価の結果が、アクティビティーのその後の動作を決定します。

- 結合条件が `false` と評価され、**suppressJoinFailure** 属性が `true` に設定されます。

  アクティビティーの状態はスキップに変わり (2)、アクティビティーを離れるリンクはデッド・パスとしてナビゲートされます。
- 結合条件が `false` と評価され、**suppressJoinFailure** 属性が `false` に設定されます。

アクティビティーはまだ開始されていないため、非アクティブ状態のままです。また、bpws:joinFailure 標準障害が引き起こされます。

- 結合条件が true と評価されます。

  フロー内で囲まれていないアクティビティーの場合、それは予期される動作です。以降のアクティビティーの動作は、アクティビティーの入り口で評価される出口条件があるかどうかによって異なります。

  – 出口条件が true に評価された場合は、アクティビティーの状態がスキップに変わり (2)、アクティビティーを離れるリンクの遷移条件が評価されます。
  – 出口条件が false に評価された場合、または出口条件が指定されていない場合は、アクティビティーがアクティブ化され、状態が実行中に変化します (3a)。アクティビティーの実装が実行されて正常に完了すると、以降のアクティビティーの動作が、アクティビティーの入り口で評価される出口条件があるかどうかによって決定されます。
    - このような出口条件が指定されていて true に評価された場合、または出口条件が指定されていない場合は、アクティビティーの状態が終了に変化し (3b)、アクティビティーを離れるリンクの遷移条件が評価されます。
    - 出口条件が false に評価された場合は、アクティビティーの状態が停止に変化します (9b)。

    「**エラーの継続**」設定が yes に設定されており、実装が失敗した場合 (例えば assign アクティビティーの copy ステートメントの構文が正しくない場合) には、アクティビティーの状態が失敗に変化します (3c)。すべての short-lived アクティビティーは割り込み不可能です。その結果、実行状態は表示されません。

アクティビティー・インスタンスは、いずれの状態でも (非アクティブ状態も含めて) スキップできます。アクティビティーが非アクティブ状態にある場合は、ナビゲーションによってそのアクティビティーに到達したときに、結合条件の結果に関係なく、状態が非アクティブからスキップに変化します (2)。アクティビティーを離れるリンクの遷移条件も評価されます。アクティビティーが自動的にスキップされる場合、条件は評価されません。

プロセスの「エラーの継続」設定が no に設定されている場合の障害処理動作

「**エラーの継続**」設定が no に設定されている場合には、障害リンクまたは直接エンクロージング障害ハンドラーによって障害が catch されないと、アクティビティーが停止状態になります (9a から 9d)。停止状態に到達する可能性があるのは次の状態のときです。

- アクティベーションのアクティブ化が失敗 (結合条件の評価中に例外が発生した場合など)。

  アクティビティーの状態は非アクティブから停止に変化します (9a)。アクティビティーは forceRetry または forceJoinCondition API 要求を使用して、管理者が修復できます。forceRetry API 要求では、アクティビティーの状態は非アクティブに変化し (10a)、アクティビティーのアクティブ化が再試行されます。再試行が成功した場合、状態は実行 (3a) に変化し、最後に終了 (3b) に変化します。再試行が失敗した場合、アクティビティーは停止状態に戻ります (14)。forceJoinCondition API 要求では、アクテ

ィビティーの状態は非アクティブ (10a) に変化し、次に、API パラメーターとして渡された条件値に応じて、状態は実行 (3a) またはスキップ (2) に変化します。

エラー動作の継続は、API 修復要求によって変更できます。この要求が行われてアクティブ化が再度失敗した場合、アクティビティーは非アクティブ状態で終了し (10a)、障害はエンクロージング・スコープの障害ハンドラーに伝搬されます。

- アクティビティーの実装が失敗 (assign ステートメントの XPath 式で例外が発生した場合など)。

  アクティビティーの状態は実行から停止に変化します (9b)。状態変更は単一トランザクションで発生するため、実行状態は表示されません。

  アクティビティーは forceRetry API 要求を使用して、管理者が修復できます。アクティビティーは実行状態に戻ります (10b)。また、アクティビティーは、forceComplete API 要求によって修復できます。この場合、アクティビティーは終了状態になり (11)、プロセスのナビゲーションは継続します。

  アクティビティーが修復される場合、修復ステップ中にアクティビティーの実装が再度失敗すると、再度停止状態に到達します (14)。エラー動作の継続が API 修復要求によって変更され、実装が再度失敗した場合、アクティビティーは失敗状態で終了し、障害はエンクロージング・スコープの障害ハンドラーに伝搬されます。
- アクティビティーを離れるリンクに対する遷移条件の評価が失敗。

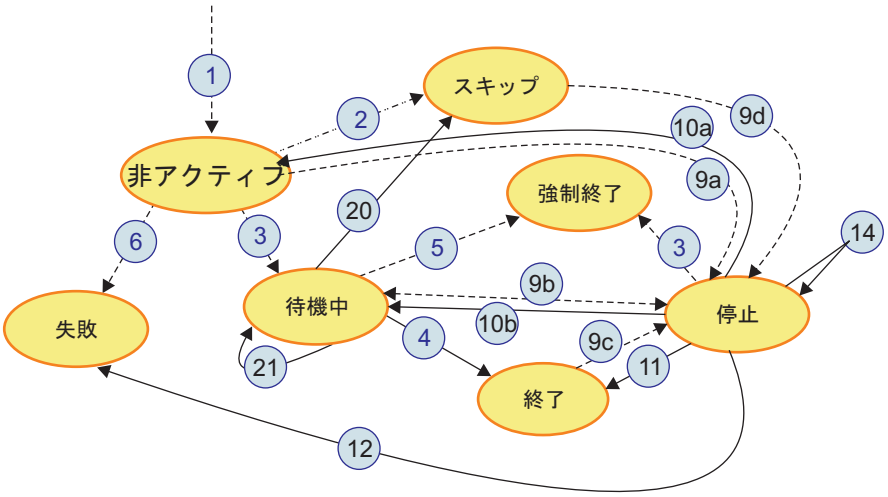
  エラーが発生する前、アクティビティーの状態は終了またはスキップでした (9c または 9d)。アクティビティーは forceComplete API 要求を使用して管理者が修復できます。評価が成功した場合、状態は再度終了になります (11)。評価が失敗した場合は、アクティビティーの状態は停止 (14) または失敗 (12) のいずれかになります。

  代わりに、forceNavigate API 要求を使用してアクティビティーを修復することもできます。この場合、管理者は、たどるべきアクティビティーの発信リンクを決定できます。アクティビティーの状態は終了に戻り (11)、遷移条件は評価されません。ただし、指定したリンクの遷移条件は `true` に評価されたものと見なされます。そのため、アクティビティーが並列フローにある場合、他のリンクはすべてデッド・パスとしてナビゲートされます。

例えば、アクティビティーが停止状態で、エンクロージング・スコープが強制終了された場合、並列ブランチにおける catch されていない障害のために、アクティビティーは強制終了されます。状態は、強制終了状態に変化します (13)。

外部イベントを待機するアクティビティーの状態遷移図

次の図は、wait または receive アクティビティーのライフ・サイクル中に発行される可能性がある状態および状態遷移を示しています。



receive および wait アクティビティーの開始段階、および停止状態からの、あるいは停止状態への状態遷移は、short-lived アクティビティーと同じです。しかし、receive および wait アクティビティーがアクティブ化された後、状態は実行ではなく待機に変化します (3)。 receive または wait アクティビティーは外部要求を受け取るか、または指定されたタイムアウトを待機する準備ができました。その後で完了し、終了状態へと移行します (4)。receive アクティビティーの場合、終了状態への遷移は、受信されたメッセージによってトリガーされます。 wait アクティビティーの場合、この遷移は指定された待機時間が経過した後に自動的に行われます。強制完了 API 要求を使用してこの遷移を強制的に行うこともできます。ただし、receive アクティビティーまたは wait アクティビティーに出口条件が指定されており、その条件評価属性が `on exit` に設定されている場合は、出口条件が `false` に評価されると、アクティビティーの状態が停止に変化し (9b)、終了状態にはなりません。アクティビティーに有効期限が定義されている場合、アクティビティーの管理者あるいはエンクロージング・スコープまたはプロセスの管理者は、その有効期限が切れたときにアクティビティーの状態を変更せずにスケジュールを変更できます (21)。

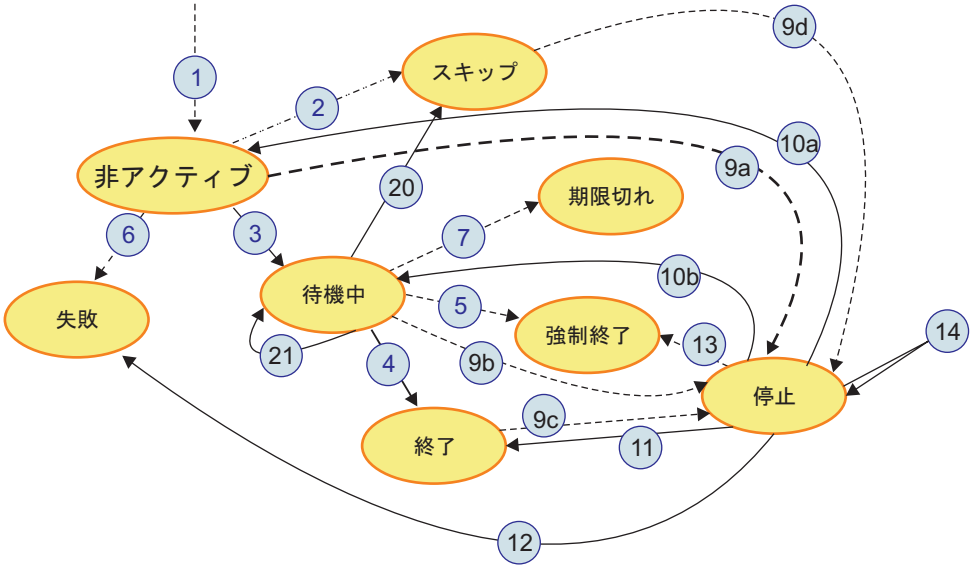
アクティビティーの開始が完了する前に (wait アクティビティーの待機時間の評価が失敗したときなど)、wait または receive アクティビティーが失敗することがあります。「**エラーの継続**」設定が `yes` に設定されている場合、またはエンクロージング・スコープの障害リンクか障害ハンドラーによって障害が処理される場合に障害が発生すると、アクティビティーの状態は、待機状態になる前に失敗に変化します (6)。

アクティビティーが待機状態の場合、エンクロージング・プロセスは終了要求を受け取ることがあります。あるいは、wait または receive アクティビティーに並行する分岐で障害が発生します。これらのイベントのいずれかが発生した場合、wait または receive アクティビティーは終了され、アクティビティーの状態は強制終了に変化します (5)。

wait または receive アクティビティーは待機状態にある場合にスキップできます。アクティビティーの状態は即時にスキップ状態に変化します (20)。この場合、アクティビティーを離れるリンクの遷移条件が評価されます。

pick (receive choice) アクティビティーの状態遷移図

次の図は、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) の状態および状態遷移を示しています。

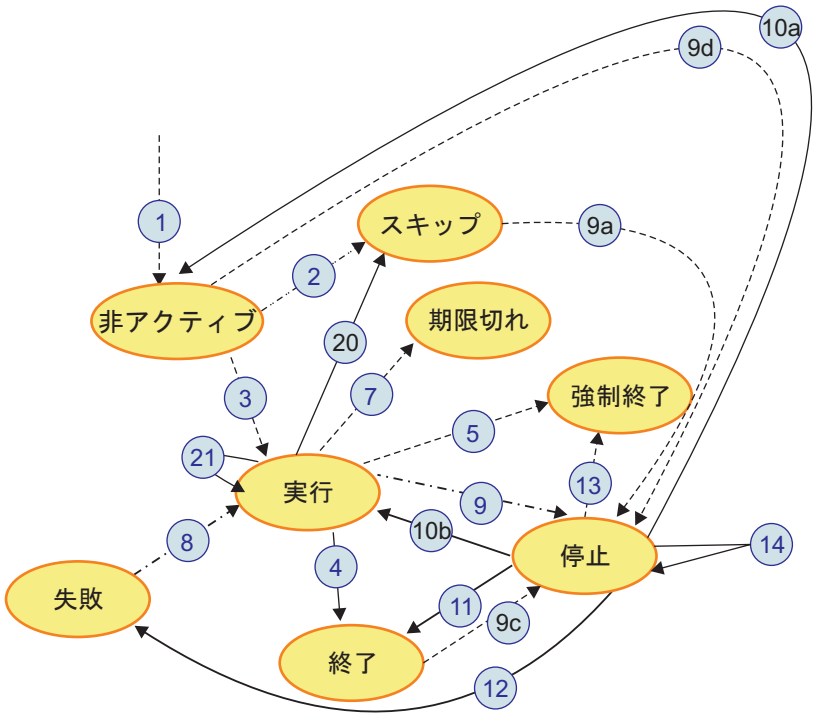


pick アクティビティーの場合、状態および状態遷移 (1) から (6) 、および停止状態とスキップ状態との遷移は、receive アクティビティーと同じです。

さらに、pick アクティビティーの要求が到達する前に待機中の pick アクティビティーのオン・アラーム・ブランチが活動状態にされると、pick アクティビティーの有効期限が切れる可能性があります。その後アクティビティーは期限切れ状態 (7) になります。

invoke アクティビティーの状態遷移図

invoke アクティビティーの場合の状態遷移は、対応するサービスが同期的に呼び出されるか非同期的に呼び出されるかによって異なります。次の図は、非同期実装を伴う invoke アクティビティーのライフ・サイクル中に発行される可能性がある状態および状態遷移を示しています。サービス要求トランザクションに対するサービス応答が後続のトランザクションで起こる場合、実装は非同期です。



invoke アクティビティーのアクティブ化は、その他のすべてのアクティビティー・タイプのアクティブ化と同じです (1)、(2)。

invoke アクティビティーが通常どおり最後まで実行されて完了すると、そのアクティビティーは開始され、状態が実行に変化します (3)。サービス呼び出しが正常に戻されると、そのアクティビティーは終了状態になります (4)。

サービスへの応答がなく、アクティビティーが停止状態にある限り、管理者はアクティビティーを強制再試行または強制完了できます。このことは、システム障害などでサービスに応答できない場合に役立ちます。また、実行から停止 (9)、失敗 (8)、および終了 (4) への状態遷移は、対応する API によって発生することもあります。

他のすべてのアクティビティーと同様に、invoke アクティビティーは停止できます (9)。その後、アクティビティーは管理アクションによって修復されるか、またはエンクロージング・スコープまたはプロセスも強制終了されるため、アクティビティーは強制終了されます (13)。

実行状態のアクティビティーは、そのアクティビティーに有効期限が定義されている場合、有効期限が切れる可能性があります。その後、アクティビティー状態は期限切れ (7) になり、タイムアウト障害がスローされます。この障害は障害ハンドラーが扱うことができます。

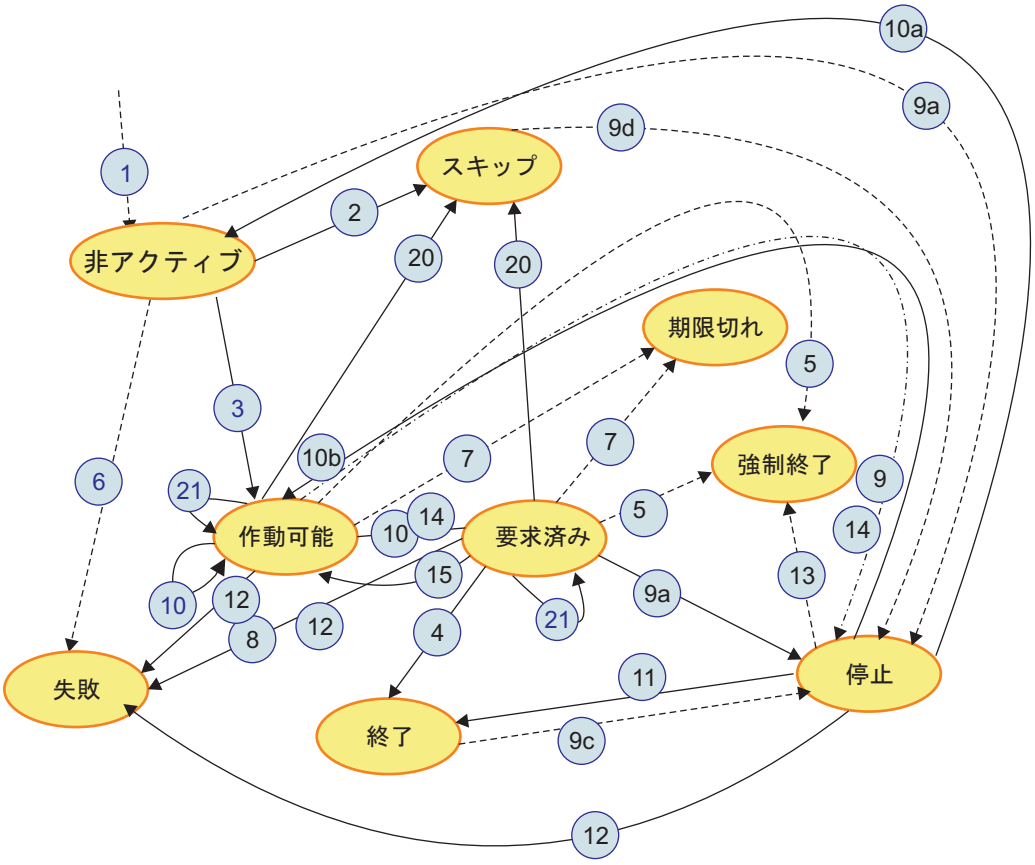
例えば、プロセス内の並列パスでの障害などのためにアクティビティーのエンクロージング・スコープが強制終了されたときに、そのアクティビティーが実行状態の場合は、そのアクティビティーも強制終了され、強制終了状態になります (5)。

同期サービス呼び出しを伴う invoke アクティビティーの状態遷移は、Java 断片の場合と同じです。同期呼び出しと非同期呼び出しの状態および状態遷移の違いは以下のとおりです。

- 同期サービス呼び出しを伴う invoke アクティビティーの実行状態は表示されません。
- 同期サービス呼び出しを伴う invoke アクティビティーには有効期限は適用されません。期限切れ状態に達することはありません。
- 同期サービス呼び出しを使用する invoke アクティビティーは強制終了しません。

単一または順次所有権を持つヒューマン・タスク・アクティビティーの状態遷移図

次の図は、単一または順次所有権を持つタスクのヒューマン・タスク・アクティビティーのライフ・サイクル中に発行される可能性がある状態および状態遷移を示しています。



ヒューマン・タスク・アクティビティーの実行時の動作は、invoke アクティビティーの動作と類似しています。invoke アクティビティーの実行状態は、ヒューマン・タスク・アクティビティーの作動可能状態と要求済み状態に相当します。作動可能状態は、そのアクティビティーではユーザーが作業する準備ができていることを示します。いずれかのユーザーが作業のためにアクティビティーを要求すると、そのアクティビティーは要求済み状態になります (15)。

そのアクティビティーで作業するユーザーは、要求される情報を入力してアクティビティーを完了します。その後、アクティビティーは終了、失敗、または停止状態になります。あるいは、そのアクティビティーを要求したユーザーがアクティビティーを完了できないと判断する場合があります。このように判断したら、ほかの人が作業できるように、そのアクティビティーを解放します。この場合、アクティビティーは作動可能状態に戻ります (16)。

ヒューマン・タスク・アクティビティーは作動可能状態または要求済みの状態にある場合にスキップできます。どちらの場合も、状態変更がスキップされ、インライン・ヒューマン・タスクは強制終了します。次のナビゲーション・ステップで、アクティビティーを離れるリンクの遷移条件が評価されます。

その他の状態遷移は、非同期サービス呼び出しを使用する invoke アクティビティーの場合と同じです。

並列所有権を持つヒューマン・タスク・アクティビティーの状態遷移図

並列所有権を持つタスクの状態遷移図は、非同期 invoke アクティビティーの状態遷移図と類似しています。ヒューマン・タスク・アクティビティーは、アクティビティーがアクティブ化されると、実行状態になります。必要な潜在的所有者のすべてが作業を完了すると、アクティビティーは終了します。作動可能状態および要求済み状態は、並列所有権を持つタスクに関連付けられたアクティビティーには使用されません。

関連概念

38 ページの『ビジネス・プロセスでの障害処理および補正処理』
障害とは、ビジネス・プロセスの正常な処理を変更する可能性がある例外的な状態です。障害は、サービス呼び出しから返されたりプロセスによって明示的に発生させたりする可能性があり、ランタイム環境によって発生したシステム障害の可能性もあります。うまく設計されたプロセスでは、障害を考慮し、それらをどんな場合でも処理します。補正とは、障害を処理するための方法の 1 つです。

43 ページの『アクティビティーおよびビジネス・プロセスのエラー動作の継続』
ビジネス・プロセスを定義するときには、予期しない障害が発生し、その障害に対して障害ハンドラーが定義されていない場合の動作を指定できます。プロセスを定義するときに「**エラーの継続**」設定を使用すると、障害が発生した場合にプロセスを停止するように指定できます。

サブプロセスのライフ・サイクル管理

別のプロセスによって作成され、開始されるプロセスを、*サブプロセス* といいます。サブプロセスのライフ・サイクルの管理方法は、これらのプロセスをどのようにモデル化するか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、ビジネス・ロジックの 1 つ以上のステップを独立したプロセスとして実装し、このプロセスをメイン・プロセスから呼び出すことには意味があります。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、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。

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・サイクルの操作は、中断、再開、終了、削除、および補正で構成されます。親子の関係では、プロセスのライフ・サイクルを管理する操作は、トップレベルのプロセス・インスタンスでのみ実行することができます。

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。この属性は、以下の値のうちのいずれかをとることができます。

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ない場合と呼び出される場合があるプロセス・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・サイクル操作は、サブプロセス・インスタンスに伝搬されません。

片方向インターフェースを使用して作成され、開始される長期実行プロセスは、ピア・プロセスと見なされます。その autonomy 属性は、実行時には無視されます。

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・サイクル操作は、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。子プロセスは、その親プロセスに戻るときに完了する必要があります。すなわち、子プロセスの最後の操作は、呼び出し元の親プロセスに対する応答である必要があります。プロセス・ロジックで考えられるすべてのパスが、そのパスでの最後の操作となる応答アクティビティーで終了することを確認してください。

親が終了する場合、親はすべての子プロセスが終了するまで待つ必要があります。子プロセスを呼び出した invoke アクティビティーの有効期限が切れるか、スキップされるか、強制的に再試行されるか、強制的に完了とされる場合、子プロセスが先に強制終了されます。子プロセスの強制終了中、invoke アクティビティーは、子プロセスが強制終了状態に達するまでは実行状態のままです。

Microflow は常に子プロセスとして実行されます。すなわち、その autonomy 属性は無視されます。

親子関係は、モジュール内で、または SCA インポートまたはエクスポートを使用してモジュール境界を越えて、直接対話するプロセス間でのみ確立できます。別の SCA コンポーネント (例えば、2 つのプロセス・コンポーネント間で結合されているインターフェース・マップ・コンポーネント) がこの対話に割り込むと、親子関係が確立されないことがあります。

関連概念

『ビジネス・プロセスによって呼び出されるスタンドアロン・ヒューマン・タスクのライフ・サイクル』
インライン・タスクのライフ・サイクルは、常に、それに関連付けられているビジネス・プロセスによって管理されます。スタンドアロン予定タスクのライフ・サイクルは、そのタスクの定義に応じた呼び出しビジネス・プロセスで管理できます。

ビジネス・プロセスによって呼び出されるスタンドアロン・ヒューマン・タスクのライフ・サイクル

インライン・タスクのライフ・サイクルは、常に、それに関連付けられているビジネス・プロセスによって管理されます。スタンドアロン予定タスクのライフ・サイクルは、そのタスクの定義に応じた呼び出しビジネス・プロセスで管理できます。

再利用のためには、多くの場合、ビジネス・ロジックの 1 つのステップを独立したスタンドアロン・タスクとして実装し、そのタスクをメイン・プロセスのさまざまな場所から呼び出すのが妥当です。そのようなアプリケーションをデプロイするときには、スタンドアロン・タスクを同じ Business Process Choreographer データベースにデプロイする必要があります。

スタンドアロン予定タスクには、呼び出しプロセスとの対等リレーションシップまたは親子リレーションシップがあります。そのリレーションシップによって、呼び出したタスクのライフ・サイクルを管理する方法が決まります。

プロセス対タスクのリレーションシップは、タスクの autonomy 属性によって決まります。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。

peer タスクとビジネス・プロセスのリレーションシップが対等であれば、タスクのライフ・サイクルはビジネス・プロセスから独立しています。

child タスクとビジネス・プロセスのリレーションシップが親子である場合、プロセス・インスタンスのライフ・サイクル操作は、その一部がタスク・インスタンスにも適用されます。適用される操作は、削除と終了です。

また、呼び出し invoke アクティビティーの以下のライフ・サイクル操作も、タスク・インスタンスに適用されます。

- invoke アクティビティーを再始動すると、現在のタスク・インスタンスが削除され、新しいタスク・インスタンスが作成され、開始されます。
- invoke アクティビティーを強制的に完了すると、タスク・インスタンスが終了します。
- 実行状態の invoke アクティビティーをスキップすると、タスク・インスタンスは強制終了されます。
- invoke アクティビティーを削除または強制終了すると、タスク・インスタンスは削除されます。

タスクの autonomy 属性が child に設定されていても、タスク・インスタンスをビジネス・プロセスとは別に中断および再開することはできます。

親子リレーションシップは、直接対話するプロセスとタスクの間でのみ確立できます。別の SCA コンポーネント (例えば、プロセスとタスクの間で結

合されているインターフェース・マップ・コンポーネント) がこの対話をインターセプトすると、親子リレーションシップが確立されないことがあります。

関連概念

21 ページの『サブプロセスのライフ・サイクル管理』
別のプロセスによって作成され、開始されるプロセスを、サブプロセス といいます。サブプロセスのライフ・サイクルの管理方法は、これらのプロセスをどのようにモデル化するかによって異なります。

実行時のプロセス・インスタンスの動的変更

通常、ビジネス・プロセスは、プロセス・モデルで定義されているとおりにナビゲートされます。ところが、状況によっては、実行時にプロセス・インスタンスのナビゲーションをオーバーライドしなければならない場合もあります。例えば、プロセス・インスタンスを修復できるようにしたい場合や、現行コンテキストに適したアクティビティーだけを実行するようにしたい場合などです。

プロセス・インスタンス内で前後にジャンプしたり、アクティビティーをスキップしたりすることによって、プロセスのナビゲーションを動的に変更できます。そのような状況では、プロセス・インスタンスのナビゲーションを続行するために、プロセスの変数に含まれているプロセス・データを変更しなければならない場合もあります。

Business Process Choreographer Explorer と Business Process Choreographer API では、実行時にプロセス・インスタンスを動的に変更する操作がサポートされています。さらに、WebSphere が提供する Business Space でも、プロセス・インスタンスの各部分の再実行やアクティビティーのスキップがサポートされています。

プロセス・インスタンス内での前後へのジャンプ

プロセス・インスタンス内部でジャンプ操作を使用して、実行時にプロセス・インスタンスを動的に変更できます。1 つのアクティビティー (ソース・アクティビティー) から別のアクティビティー (ターゲット・アクティビティー) にジャンプできます。ソース・アクティビティーは、いずれかのアクティブ状態 (実行中、待機中、作動可能、要求済み、停止) になっている基本アクティビティーでなければなりません。ターゲット・アクティビティーは、基本アクティビティーであっても構造化アクティビティーであってもかまいません。

以下のジャンプ・アクションを使用できます。

完了してジャンプ
要求済みの状態になっているヒューマン・タスク・アクティビティーを完了して、ターゲット・アクティビティーにジャンプします。

強制完了してジャンプ
アクティビティーを強制的に完了して、ターゲット・アクティビティーからプロセスのナビゲーションを続行します。

スキップしてジャンプ
ソース・アクティビティーをスキップして、ターゲット・アクティビティーからナビゲーションを続行します。

ジャンプ
ソース・アクティビティーからターゲット・アクティビティーにジャンプします。

ジャンプ API の場合のみ: ソース・アクティビティーは、スキップ、終了、失敗、または期限切れのいずれかの実行状態である基本アクティビティーでなければなりません。また、アクティビティーを含むパスで最後にナビゲートされるアクティビティーでなければなりません。関連付けられたプロセス・インスタンスは実行状態が「中断」である必要があります。プロセス・インスタンスが再開されると、ナビゲーションは指定されたターゲット・アクティビティーで続行されます。

ジャンプ・アクションでは、ソース・アクティビティーを完了するか、強制完了するか、スキップすることになります。ジャンプの後は、ターゲット・アクティビティーからプロセスのナビゲーションを続行します。プロセスの先の方にジャンプすることができます。この場合ターゲット・アクティビティーはプロセス・インスタンスの後の方にあります。プロセス内の前の方のアクティビティーにジャンプして戻ることもできます。

ジャンプ操作は、シーケンス・アクティビティーに含まれているアクティビティー間で実行できます。汎用フロー・アクティビティーや並列アクティビティー・アクティビティー (フロー・アクティビティーともいう) 内の、フォークや結合を含まないパスでも、ジャンプ操作は可能です。このようなジャンプ・アクションのすべてにおいて、ソース・アクティビティーとターゲット・アクティビティーは、収容側のアクティビティー内の同じネスト・レベルに存在していなければなりません。

ソース・アクティビティーの出口条件やターゲット・アクティビティーの入り口は、ジャンプ・アクションでは無視されます。

ジャンプ・アクションを実行するには、エンクロージング・スコープのスコープ管理者、プロセス管理者、システム管理者のいずれかでなければなりません。

アクティビティーのスキップ

アクティビティーをスキップすることによっても、プロセス・インスタンスを動的に変更できます。スキップできるのは、いずれかのアクティブ状態になっている基本アクティビティーか、またはプロセスの後のほうでアクティブになる可能性がある基本アクティビティーです。

アクティブなアクティビティーをスキップすると、そのアクティビティーの実装が終了し、そのアクティビティーの後からプロセスのナビゲーションが続行します。例えば、アクティビティーに発信リンクがあれば、プロセスのナビゲーションはそれらのリンクの遷移条件の評価から続行することになります。

プロセス・フローの後の方にあるアクティビティーをスキップすると、そのアクティビティーにスキップのマークが付きます。ナビゲーションがそのアクティビティーに達すると、そのアクティビティーはスキップされ、そのアクティビティーの後からナビゲーションが続行します。ナビゲーションが対象のアクティビティーに達するまでは、スキップ要求を取り消すことも可能です。

アクティビティーをスキップするには、エンクロージング・スコープのスコープ管理者、プロセス管理者、システム管理者のいずれかでなければなりません。さら

に、アクティビティー管理者であれば、現時点でアクティブになっているアクティビティーをスキップできます。

変数の変更

実行時にプロセス・インスタンスのフローを変更するときには、多くの場合、ジャンプしたアクティビティーまたはスキップしたアクティビティーの後でプロセスが正しく流れるように、変数を更新する必要があります。例えば、修復のシナリオにおいて、ジャンプ・アクションの前に有効なデータを用意することで、そのデータに基づいて後続のアクティビティーが正常に実行されるようにすることができます。

以下のアクションがサポートされています。

- 特定のアクティビティーのすべての変数の名前を取得する
- グローバル変数またはローカル変数の実際の値または初期値を取得する
- グローバル変数またはローカル変数の値を設定する

変数の値を表示するには、少なくともプロセスまたはエンクロージング・スコープの読み取り権限が必要です。変数を更新するには、スコープ管理者、プロセス管理者、システム管理者のいずれかでなければなりません。

ビジネス・プロセスの呼び出しシナリオ

ビジネス・プロセスは、Service Component Architecture (SCA) コンポーネントの実装です。ビジネス・プロセスは、サービスを他のパートナーに公開したり、他のパートナーから提供されるサービスを消費したりすることができます。ビジネス・プロセスは、Business Process Choreographer API で使用可能なサービス・プロバイダー、他の SCA サービス・コンポーネントの SCA サービス・プロバイダー、または他の SCA サービス・コンポーネント (他のビジネス・プロセスを含む) を呼び出す SCA クライアントのいずれかです。

Business Process Choreographer API で使用可能なサービス・プロバイダーとしてのビジネス・プロセス

Business Flow Manager API を使用すると、ビジネス・プロセスをインスタンス化できます。このクライアント・アプリケーションは、ビジネス・プロセス・インスタンスを作成および開始したり、既存のプロセス・インスタンスを照会および処理したりすることができます。Business Flow Manager API は、EJB クライアント、Web サービス・クライアント、JMS クライアント、および REST クライアントの設計に使用できる、EJB、Web サービス、JMS メッセージ・インターフェース、および REST インターフェースとして提供されます。

他の SCA サービス・コンポーネントの SCA サービス・プロバイダーとしてのビジネス・プロセス

この呼び出しシナリオでは、ビジネス・プロセスは、クライアントとして動作する他の SCA コンポーネントによって呼び出すことができる SCA コンポーネントを表します。ビジネス・プロセスによって提供されるサービスは、SCA コンポーネントの実装として、SCA クライアントから呼び出すことができます。これらの機構には、以下が含まれます。

- SCA クライアント (参照) と、ビジネス・プロセスを表すコンポーネントのインターフェースとを接続するためのワイヤー
- 対話スタイル、トランザクションの振る舞い、対話の信頼性などの面を決定する、コンポーネント参照およびコンポーネント・インターフェースの SCA 修飾子設定

他の SCA サービス・コンポーネントを呼び出す SCA クライアントとしてのビジネス・プロセス

ビジネス・プロセスは別のビジネス・プロセスを呼び出すことができます。これを行うには、同じモジュール内またはモジュール間での SCA ワイヤリングを使用します。SCA ワイヤリングの場合は、呼び出し元が別のサービスに関連付けられます (*早期バインディング* と呼びます)。別のプロセスによって提供されるサービスを呼び出す場合は、実行時バインディングを使用して、現在有効なプロセスのバージョンを選択できます。これを行うには、呼び出し側プロセスのパートナー・リンクでの指定を使用します。

他のビジネス・プロセスを呼び出す SCA クライアントとしてのビジネス・プロセス

SCA クライアントと SCA サービスの両方がビジネス・プロセスで表される場合、その両方を SCA レベルおよびビジネス・プロセス・レベルで選択できます。SCA レベルでは、SCA ワイヤーを使用して、SCA クライアントを SCA サービスに接続できます。ビジネス・プロセス・レベルでは、サービス・プロバイダーとして動作するビジネス・プロセスの名前を使用して、パートナー・リンクを関連付けることができます。

ビジネス・プロセスの対話に影響を及ぼす要因

各種の呼び出しシナリオにおけるビジネス・プロセスの振る舞いは、いくつかの要因から影響を受けます。これには、対話スタイル、ビジネス・プロセスのタイプ、操作タイプ、およびサービス・エンドポイントの解決が含まれます。

対話スタイル

ビジネス・プロセスで提供される操作は、同期的にも非同期的にも呼び出すことができます。

重要: 同期対話の妥当な応答時間は、数秒を超えてはなりません。ビジネス・プロセスにより実装されている要求応答操作で、短時間以内に結果が戻らない場合、非同期対話スタイルを使用してパフォーマンスを改善することを検討してください。このような操作を同期的に呼び出すと、リソースがブロックされます。また、システム・ワークロードに依存するシチュエーションがタイムアウトになる傾向もあるため、非決定論的であるといえます。

ビジネス・プロセスのタイプ

ビジネス・プロセスは、Microflow プロセスまたは長期実行プロセスのいずれかです。それぞれのプロセス・タイプの特性は、呼び出しシナリオに影響を及ぼします。

WSDL 操作タイプ

Service Component Architecture (SCA) 参照および SCA インターフェースは、1 つ以上の操作を含む WSDL ポート・タイプに関連付けられています。操作は、片方向操作の場合も要求応答操作の場合もあります。

- 片方向操作では、サービスの完了が呼び出し側クライアントに通知されません。サービス実行は、関連するサービスを正常に呼び出して終了します。
- 要求応答操作では、サービスの完了が呼び出し側クライアントに通知されます。サービスは、サービス実行の結果が呼び出し側クライアントで使用可能になると終了します。

サービス・エンドポイントの解決

ビジネス・プロセスのコンテキストでは、以下のいずれかの方法で、呼び出し側のクライアントを呼び出されるサービスに関連付けることができます。

- SCA ワイヤーは、呼び出されるサービスのインターフェースに SCA 参照を静的に関連付けます。これは SCA レベルの機構であり、クライアント、サービス、またはその両方がビジネス・プロセスとして実装されている場合に適用できます。
- エンドポイント参照 (EPR) は BPEL パートナー・リンクに割り当てることができます。EPR は、パートナー・リンクを使用して呼び出すサービスのエンドポイント・アドレスを判別します。したがって、動的サービス呼び出しの場合に SCA で可能な呼び出しに準拠する任意のサービスを動的に呼び出すことができます (Web サービス・バインディング、MQ JMS バインディング、MQ バインディング、SCA バインディングなど)。
- ビジネス・プロセス・テンプレート名は、SCA クライアントとして動作するビジネス・プロセスの一部であるパートナー・リンクに対して設定できます。テンプレート名は、同一のサーバーまたはクラスターにデプロイされている別のビジネス・プロセスの名前を一意的に決定します。

関連概念

4 ページの『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。

『ビジネス・プロセスとサービスの間の動的バインディング』
このシナリオでは、ビジネス・プロセスがクライアントとして使用されること、およびプロセスの実行時にプロセス・モデルが BPEL パートナー・リンクを割り当て可能であることを前提としています。サービスの動的バインディングにより、ビジネス・プロセスは、実行時にアドレスが決定されるサービスを呼び出すことができます。これは、サービスのエンドポイントがプロセスのモデル化時に判明していない場合に特に有用です。

ビジネス・プロセスとサービスの間の動的バインディング

このシナリオでは、ビジネス・プロセスがクライアントとして使用されること、およびプロセスの実行時にプロセス・モデルが BPEL パートナー・リンクを割り当て可能であることを前提としています。サービスの動的バインディングにより、ビジネス・プロセスは、実行時にアドレスが決定されるサービスを呼び出すことができます。これは、サービスのエンドポイントがプロセスのモデル化時に判明していない場合に特に有用です。

プロセス・モデルでは、ビジネス・プロセスが対話するサービスがパートナー・リンクとしてモデル化されています。パートナー・リンクを使用してパートナーのサービスに対する操作を呼び出すには、パートナー・サービスのバインディングおよ

び通信データが必要です。パートナー・サービスに関連する情報は、通常、ビジネス・プロセスをデプロイするときに設定します。

BPEL パートナー・リンクに対応する Service Component Architecture (SCA) 参照は、未接続のままにすることができます。この場合はデフォルトとして、呼び出しに使用されるバインディングがエンドポイント・アドレス URL に応じて SCA または Web サービス・バインディングになります。代わりに、SCA 参照を SCA インポートに事前に接続しておくこともできます。この場合、バインディングおよびサービス品質の指定は SCA インポートから取得され、サービス・エンドポイント・アドレスのみがエンドポイント参照によって上書きされます。

プロセス・モデルには、パートナー・リンクのエンドポイント参照値を割り当てる assign アクティビティーまたは Java 断片アクティビティーのいずれかを組み込んでください。パートナー・リンクを接続しない場合、サービスは以下のいずれかの方法で呼び出されます。

- microflow の場合、サービスは同期的に呼び出されます。
- 長期実行プロセスの場合、サービスは非同期的に呼び出されます

関連概念

27 ページの『ビジネス・プロセスの対話に影響を及ぼす要因』
各種の呼び出しシナリオにおけるビジネス・プロセスの振る舞いは、いくつかの要因から影響を受けます。これには、対話スタイル、ビジネス・プロセスのタイプ、操作タイプ、およびサービス・エンドポイントの解決が含まれます。

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。 SCA サービスとプロセスの間でのデータ交換方法は、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実装されます。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。
- データの交換には、データ型の変数が使用される。

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、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。

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。
- データの交換には、データ型の変数が使用される。

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。

ビジネス・プロセスのトランザクションの振る舞い

ビジネス・プロセスは、トランザクションの一部として実行されます。ビジネス・プロセスのナビゲーションは、長期実行プロセスの場合には複数のトランザクションにまたがり、Microflow の場合には 1 つのトランザクションの一部となります。このようなナビゲーション・トランザクションは、外部要求、内部メッセージ、または非同期サービスからの応答によってトリガーされます。トランザクションが開始すると、プロセス定義に従って必要なアクティビティーが実行されます。呼び出されたサービスは、トランザクションに参加できます。

関連概念

4 ページの『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。

9 ページの『プロセス・インスタンスの状態遷移図』
プロセスは、プロセス・インスタンスのライフ・サイクル中に重要なイベントが生じるたびに状態が変化します。例えば、API 要求によって、実行状態のプロセスが中断状態にされる、などです。状態遷移図には、プロセスのライフ・サイクル中に発生する可能性がある状態遷移が示されます。Microflow と長期実行プロセスでは、状態遷移図が異なります。

26 ページの『ビジネス・プロセスの呼び出しシナリオ』
ビジネス・プロセスは、Service Component Architecture (SCA) コンポーネントの実装です。ビジネス・プロセスは、サービスを他のパートナーに公開したり、他のパートナーから提供されるサービスを消費したりすることができます。ビジネス・プロセスは、Business Process Choreographer API で使用可能なサービス・プロバイダー、他の SCA サービス・コンポーネントの SCA サービス・プロバイダー、または他の SCA サービス・コンポーネント (他のビジネス・プロセスを含む) を呼び出す SCA クライアントのいずれかです。

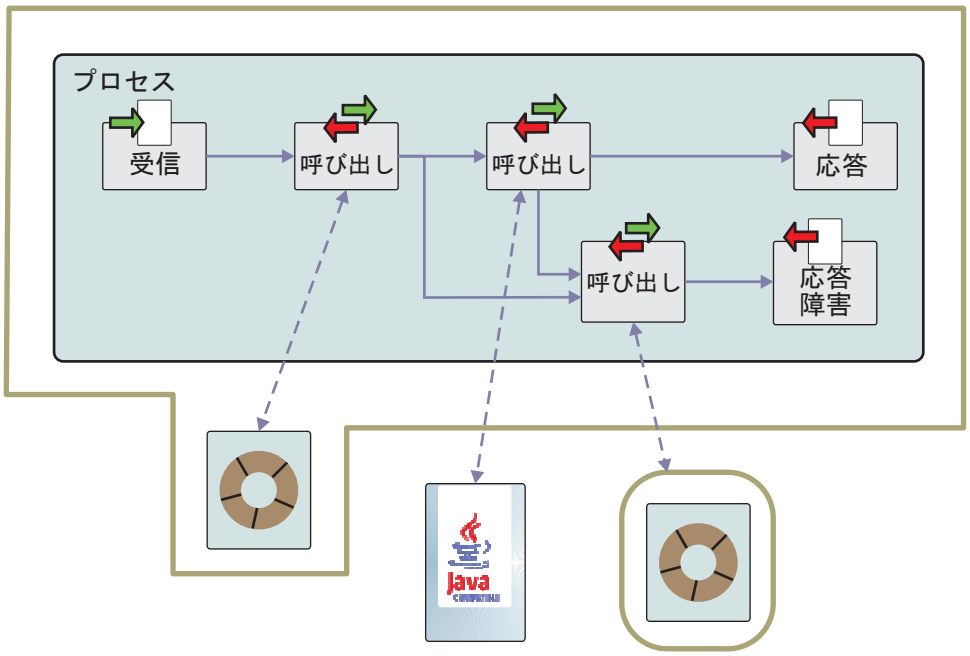
Microflow のトランザクションの振る舞い

Microflow は短期存続プロセスです。トランザクション内で実行される場合と、Microflow の SCA コンポーネントに指定されているようにアクティビティー・セッション内で実行される場合があります。ここでは、トランザクションの一部として実行される Microflow について説明します。

Microflow は割り込み不可です。したがって Microflow には、外部イベントやユーザー対話 (例えばヒューマン・タスク・アクティビティー) を待機するアクティビティーを含めることはできません。

Microflow は一時的なプロセスです。Microflow のプロセス・インスタンス状態はメモリーに保持されますが、ランタイム・データベースには保管されません。ただし、Microflow インスタンスの状態は、監査ログまたは Common Base Event 内に保持することができます。

以下の図は、Microflow のトランザクションおよび Microflow が対話するサービスを示しています。トランザクション境界内のサービスは、Microflow トランザクションに参加しています。境界の外側にあるサービスは、トランザクションに参加していません。



呼び出されるサービスおよび Microflow トランザクション

Microflow は 1 つのトランザクションで実行されます。ただし、Microflow が呼び出すサービスには、複数のトランザクションが関わる可能性があります。これは、invoke アクティビティーを使用して呼び出されるサービスは、Microflow のトランザクションに参加することも、独自のトランザクション内で実行することもできるためです。

以下の設定では、サービスが Microflow のトランザクションに参加するか、独自のトランザクション内で実行されるかが決まります。

- サービスの呼び出しに使用する対話スタイル。

  対話スタイルは、同期または非同期です。スタイルは、ターゲット Service Component Architecture (SCA) コンポーネントまたは SCA インポートの優先される対話スタイルによって、および以下の表に示すように操作が片方向操作であるか要求応答操作であるかによって決まります。

表 1.

ターゲット・コンポーネントまたはインポートの優先される対話スタイル	片方向操作	要求応答操作
任意	非同期呼び出し	同期呼び出し
同期	同期呼び出し	同期呼び出し
非同期	非同期呼び出し	同期呼び出し

注: サービス呼び出しの不適切なパターンの例として、優先される対話スタイルが「非同期」である要求応答操作の microflow からの呼び出しが挙げられます。

呼び出されたサービスが長期実行プロセスである場合は、その長期実行プロセスが完了する前に microflow トランザクションがタイムアウトになり、ランタイム・エラーが発生することがあります。

- プロセスおよびサービスに対して指定する SCA トランザクション修飾子。以下の修飾子があります。
  - プロセス・コンポーネントの参照に対する **suspendTransaction** 修飾子は、プロセスのトランザクション・コンテキストを、呼び出されるサービスに伝搬するかどうかを指定します。
  - サービス・インターフェースに対する **joinTransaction** 修飾子は、トランザクションが伝搬される場合に、サービスが呼び出し元のトランザクションに参加するかどうかを指定します。

以上の設定に基づいて、呼び出されるサービスには以下の規則が適用されます。

同期呼び出し

joinTransaction	**suspendTransaction = true**	**suspendTransaction = false**
joinTransaction = true	サービスは Microflow トランザクションに参加しない	サービスは Microflow トランザクションに参加する
joinTransaction = false	サービスは Microflow トランザクションに参加しない	サービスは Microflow トランザクションに参加しない

サービスが Microflow トランザクションに参加する場合、サービスによるトランザクション・リソースへの変更が保持されるのは、Microflow トランザクションがコミットされる場合のみです。サービスが Microflow トランザクションに参加しない場合、サービスによるトランザクション・リソースへの変更は、トランザクションがロールバックされる場合でも保持されることがあります。サービスによる変更を元に戻すには、補正を使用できます。

非同期呼び出し

サービスは常に独自のトランザクション内で実行されます。非同期 SCA メッセージの送信を現在のナビゲーション・トランザクションに参加させるには、Microflow の **asynchronousInvocation** 修飾子を `commit` に設定する必要があります。

関連概念

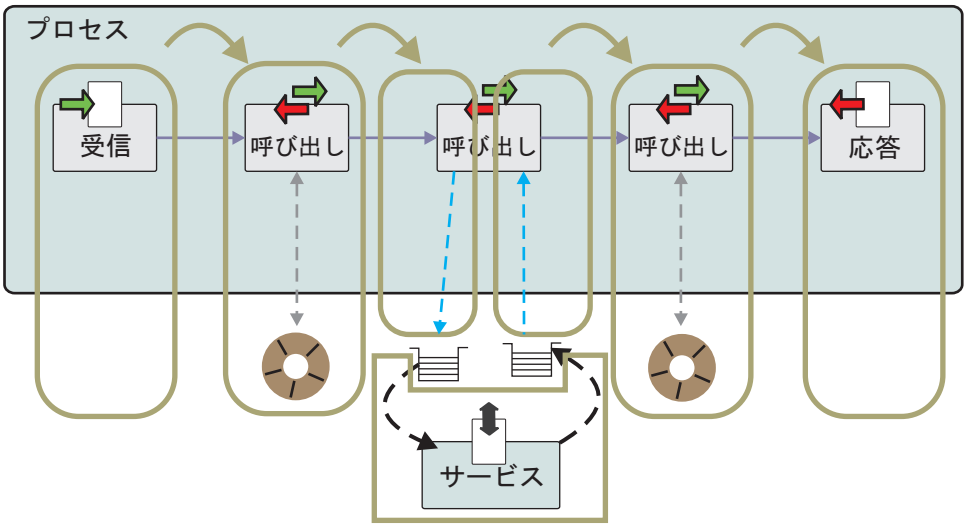
46 ページの『ビジネス・プロセスでの補正処理』
補正処理は、プロセス・モデルで補正が定義された実行中のプロセス・インスタンスにおける、障害処理の手段の 1 つです。補正は、障害が発生した時点までにコミットされた操作の影響をリバースし、整合した状態に戻します。

長期実行プロセスのトランザクションの振る舞い

長期実行プロセスは、複数のトランザクションにわたっています。各トランザクションは、Java Message Service (JMS) メッセージまたは作業マネージャー・ベースの実装によってトリガーされます。

トランザクション境界を超えるナビゲーションを可能にするため、プロセス・インスタンスおよびそのアクティビティー・インスタンスの状態は、データベースに保持されます。

以下の図は、長期実行プロセスの各ナビゲーション・ステップが、それ自体のトランザクション内でどのように振る舞うかを示しています。示されているように、ナビゲーション・ステップは、サービスを呼び出す invoke アクティビティーによって複数のアクティビティーにまたがる場合があります。また、複数のアクティビティーが 1 つのトランザクション内で実行される場合もあります。



以下に、長期実行プロセスのトランザクション境界について説明します。トランザクション境界を操作するには、トランザクションの振る舞い属性を使用します。ただし、Business Flow Manager はいつでもトランザクション境界を追加したり除去したりできます。

一般に、トランザクション境界が必要なのは以下の状況です。

- 外部要求を待機している場合。つまり、対応する要求がまだ届かないプロセス・ナビゲーションで、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) に達したとき。
- wait アクティビティー用にタイマーをスケジューリングする場合。
- invoke アクティビティーを使用してサービスを非同期に呼び出す場合。
- ヒューマン・タスク・アクティビティーを呼び出す場合。

また、Business Flow Manager は以下の状況でトランザクション境界を導入します。ただし、プロセス設計はこれらの境界に依存してはいけません。これらの境界は、プロセス・ナビゲーション時にオーバーライドされたり、将来変更される可能性があるためです。

- プロセス・ナビゲーション時に障害が発生する場合。
- サービスを同期的に呼び出す invoke アクティビティーの開始前および開始後。このサービスはプロセスのトランザクションに参加しません。
- ライフ・サイクル操作を子プロセスに伝搬する場合。例えば、親プロセスが中断されると、その子プロセスは後続のトランザクションで中断されます。
- プロセスの完了時にプロセス・インスタンスを自動的に削除する場合。
- 一連のアクティビティーにまたがるトランザクションのロールバックを引き起こす障害からのリカバリーを試行する場合。
- トランザクションの振る舞い属性を使用して指定した場合。

保証されたトランザクション境界が必要な場合は、単一トランザクション内で実行する必要があるビジネス・ロジックを Microflow に分解し、その Microflow をサブプロセスとして呼び出します。Microflow のロジックは、単一トランザクション内で常に実行されています。

トランザクション境界への影響

ビジネス・プロセスをモデル化するときは、対応する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変更して、invoke、断片、およびヒューマン・タスク・アクティビティーのトランザクション境界を示すことができます。invoke アクティビティーが、現在のトランザクションに参加していない同期サービスを呼び出す場合、トランザクションの振る舞い属性は無視されます。この場合、invoke アクティビティーが開始する前と、invoke アクティビティーが完了したあとには、常にトランザクション境界が存在します。

この属性は、次の値のいずれかをとることができます。

事前コミット (Commit before)
現在のトランザクションがコミットされ、新規トランザクションが開始します。この属性値をもつアクティビティーは、新規トランザクションの最初のアクティビティーになります。

事後コミット (Commit after)
アクティビティーが現在のトランザクションに参加します。アクティビティーが正常に完了した後、トランザクションがコミットされ、新規トランザクションが開始します。すぐ後に続くアクティビティーごとに新規トランザクションが開始するため、各後続アクティビティーがこれらの新規トランザクションの最初のアクティビティーになります。

参加 (Participates)
アクティビティーが現在のトランザクションに参加します。アクティビティーの前にも後にも、追加のトランザクション境界は設定されません。

以下の状況では、この設定の値によって、トランザクションがトランザクションの振る舞い属性の設定に応じて以下のアクティビティーのナビゲーションを続行することができます。

- invoke アクティビティーがサービスを非同期的に呼び出す場合、応答メッセージの到着によって新規トランザクションがトリガーされます。トランザクションは、invoke アクティビティーの状況が更新されるとすぐにコミットされるため、非常に短いものになります。
- 一連のヒューマン・タスク・アクティビティーでは、ヒューマン・タスク・アクティビティーごとに 2 つのトランザクションが必要です。1 つはヒューマン・タスク・アクティビティーをアクティブ化するためのもの、もう一つはヒューマン・タスク・アクティビティーを完了するためのものです。設定を「参加 (Participates)」に変更すると、トランザクションの数をヒューマン・タスク・アクティビティーごとに 1 つに削減できます。これは、直前のヒューマン・タスク・アクティビティーの完了と次のアクティビティーのアクティブ化が同じトランザクション内にあるためです。
- completeAndClaimSuccessor API を使用するサーバー制御のページ・フローを使用可能にする場合。

所有が必要 (Requires own)

アクティビティーは独自のトランザクション内で実行されます。これは、アクティビティーが開始する前に現在のトランザクションがコミットされ、このアクティビティーの完了後に新規トランザクションが開始することを意味します。

プロセスを開始するトランザクションを receive アクティビティー後にコミットするか、あるいは、対応するアクティビティーのトランザクション振る舞い属性を変更して receive-choice (pick) アクティビティーの receive アクションを完了するかを決定することもできます。receive アクティビティーおよび receive-choice アクティビティーを開始する場合、属性は次の値のいずれかをとることができます。

事後コミット (Commit after)

アクティビティーが正常に完了した後に、プロセスを開始するトランザクションがコミットされ、新しいトランザクションが開始します。この設定は、同期 API 呼び出しを使用してプロセス・インスタンスを呼び出す場合に便利です。

参加 (Participates)

アクティビティーの完了後も、プロセスを開始するトランザクションは続行されます。initiateAndClaimFirst API を使用してプロセス・インスタンスを呼び出す場合は、この設定が必要です。その API を使用すれば、新しいプロセス・インスタンスを作成してから、すぐに最初のヒューマン・タスクを要求できます。

プロセスから別の BPEL プロセスを呼び出す場合は、プロセスを開始するトランザクションの中に、対応する invoke アクティビティーを組み込まないようにする必要があります。そのためには、トランザクション振る舞い属性を以下のいずれかの方法で設定します。

- 開始側の receive アクティビティーまたは receive choice アクティビティーの属性を「事後コミット (Commit after)」に設定します。
- invoke アクティビティーの属性を「事前コミット (Commit before)」または「所有が必要 (Requires own)」に設定します。

flow アクティビティー内の並列分岐の並行ナビゲーション

flow アクティビティー内の並列分岐のナビゲーションを並行して行うには、各並列アクティビティーが個々のトランザクション内で処理されるように、各分岐の最初に新規トランザクション境界が必要です。これは、フローの最初から並行が実現されるように、各並列分岐の最初のアクティビティーに対する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)」に設定する必要があることを意味します。

並列 forEach アクティビティーの分岐の並行ナビゲーション

forEach アクティビティーの各分岐の処理は、それぞれ別個の専用トランザクションで開始されます。したがって、各分岐の並列実行が可能です。

長期実行プロセスで呼び出されるサービスおよびトランザクション

invoke アクティビティーを使用して長期実行プロセス内で呼び出されるサービスは、長期実行プロセスの現在のトランザクションに参加することも、独自のトランザクション内で実行することもできます。

以下の設定では、サービスが長期実行プロセスのトランザクションに参加するか、独自のトランザクション内で実行されるかが決まります。

- サービスの呼び出しに使用する対話スタイル。

  対話スタイルは、同期または非同期です。そのスタイルは、ターゲット SCA コンポーネントまたは SCA インポートの優先される対話スタイルによって決まります。以下の表を参照してください。

表 2.

ターゲット・コンポーネントまたはインポートの優先される対話スタイル	**片方向操作**	**要求応答操作**
任意	非同期呼び出し	非同期呼び出し
同期	同期呼び出し	同期呼び出し
非同期	非同期呼び出し	非同期呼び出し

- プロセスおよびサービスに対して指定する Service Component Architecture (SCA) トランザクション修飾子。以下の修飾子があります。
  - プロセス・コンポーネントの参照に対する **suspendTransaction** 修飾子は、プロセスのトランザクション・コンテキストを、呼び出されるサービスに伝搬するかどうかを指定します。
  - サービス・インターフェースに対する **joinTransaction** 修飾子は、トランザクションが伝搬される場合に、サービスが呼び出し元のトランザクションに参加するかどうかを指定します。

    対話スタイルと SCA 修飾子の設定に応じて、呼び出されるサービスに以下の規則が適用されます。

    同期呼び出し

joinTransaction	**suspendTransaction = true**	**suspendTransaction = false**
joinTransaction = true	サービスは長期実行プロセスのトランザクションに参加しない	サービスは長期実行プロセスのトランザクションに参加する
joinTransaction = false	サービスは長期実行プロセスのトランザクションに参加しない	サービスは長期実行プロセスのトランザクションに参加しない

    サービスが長期実行プロセスの現在のトランザクションに参加する場合、サービスによるトランザクション・リソースへの変更が保持されるのは、現在のトランザクションがコミットされる場合のみです。

    非同期呼び出し

    サービスは常にそのサービスのトランザクション内で実行されます。

トランザクションのロールバック時の正常なサービス呼び出しのリカバリー

リカバリー動作は、呼び出されたサービスが現在のトランザクションに参加するかどうかによって異なります。

invoke アクティビティーは、現在のトランザクションに参加しているサービスを呼び出します。サービスの実行が完了します。サービスの完了後にエラーが発生し、トランザクションの開始前のプロセスの状態にトランザクションがロールバックする場合、呼び出されたサービスの結果もロールバックされます。トランザクションの再試行時に、サービスはもう一度呼び出されます。

対照的に、呼び出されたサービスが現在のトランザクションに参加せず、呼び出されたサービスが応答を返す場合、その応答は別個のトランザクションに保管されます。応答の保管後にエラーが発生した場合、現在のトランザクションがロールバックされ、トランザクションが再試行されます。ただし、再試行時にサービスは再度呼び出されず、保管された応答が復元されてナビゲーションが続行します。

ビジネス・プロセスでの障害処理および補正処理

障害とは、ビジネス・プロセスの正常な処理を変更する可能性がある例外的な状態です。障害は、サービス呼び出しから返されたりプロセスによって明示的に発生させたりする可能性があり、ランタイム環境によって発生したシステム障害の可能性もあります。うまく設計されたプロセスでは、障害を考慮し、それらをどんな場合でも処理します。補正とは、障害を処理するための方法の 1 つです。

関連概念

13 ページの『アクティビティーの状態遷移図』
アクティビティー・インスタンスの実行中に重要なステップが発生すると、アクティビティー・インスタンスの状態は変化します。状態および状態遷移はアクティビティーのタイプによって異なります。

ビジネス・プロセスでの障害発生

障害を発生させるには、throw または rethrow アクティビティーを使用することも、Java 断片アクティビティーをプログラマチックに使用することもできます。サービスの呼び出しでも障害を発生させることができます。

プロセスの呼び出し元に障害を伝搬するには、障害指定とともに reply アクティビティーを使用します。

障害発生での throw アクティビティーと rethrow アクティビティー

ビジネス・プロセスでの throw アクティビティーは、標準障害を含むあらゆるタイプの障害をスローできますが、意図された使用パターンはビジネス障害をスローするためのものです。throw アクティビティーによってスローされた障害は、ビジネス・プロセス内で catch して処理する必要があります。要求応答インターフェースを使用したプロセスで、プロセス内の障害が処理されない場合、そのプロセスは bpws:missingReply 標準障害で終了します。クライアント・アプリケーションの場合、この障害は StandardFaultException オブジェクト内に返されます。

throw アクティビティーでは、ビジネス障害を返すことができません。ビジネス障害をプロセス・クライアントに返すには、reply アクティビティーを使用する必要があります。reply アクティビティーは、プロセスによって実装されたインターフェースに定義されているビジネス障害のみを返すことができます。

rethrow アクティビティーは、障害ハンドラー内で使用して、障害を次にそのエンクロージング・スコープに再スローすることができます。この方法は、現在のスコープで何らかの障害処理 (特定の補正ハンドラーをトリガーするなど) を行うが、障害が含まれる他のスコープにもこの問題を認識させる場合に有用です。また、現在の障害ハンドラーが障害を処理できないため、エンクロージング・スコープのいずれか、またはプロセスに定義されている障害ハンドラーにその障害を伝搬する場合にも、rethrow アクティビティーを使用できます。

既存の障害は障害ハンドラーからしか再スローできないため、rethrow アクティビティーは障害ハンドラー内でのみ使用できます。

Java 断片での障害発生

raiseFault メソッドを使用して、ビジネス・プロセス内の Java 断片の障害をプログラマチックに発生させることができます。以下のいずれかの方法で、ビジネス障害を発生させることができます。

- `raiseFault(QName fault, String variableName);`
- `raiseFault(QName fault);`

以下の例では、`http://process/UpdateCustomerRecordProcess/Interface0/` 名前空間に IncompleteData という障害が作成され、この障害が Java 断片からスローされます。

```
javax.xml.namespace.QName fault = new javax.xml.namespace.QName
   ("http://process/UpdateCustomerRecordProcess/Interface0/", "IncompleteData");
raiseFault(fault);
```

スローされた障害が、いずれの WSDL インターフェースで宣言されたものでもない場合は、障害の名前空間としてプロセスのターゲット名前空間を指定します。これで、catch アクティビティーを使用して、ビジネス・プロセス内のこの障害を catch できます。

障害を catch するには、ServiceBusinessException オブジェクトを直接スローするのではなく、raiseFault メッセージを使用してください。

呼び出し元への障害伝搬

障害指定とともに reply アクティビティーを行うことにより、指定した障害が要求応答操作の呼び出し元に伝搬されます。reply アクティビティーは、プロセスによって実装されたインターフェースに定義されている障害のみを返すことができます。このメソッドが有用なのは、ビジネス・プロセスは catch された障害に正しく応答できないが、プロセスの起動側はそれに応答できる場合です。例えば、ビジネス・プロセスによって見つからないアカウント番号を呼び出し元が受け渡す場合、プロセスは AccountNotFound 障害を使用してこのサービス呼び出しに応答します。

障害指定を伴う reply アクティビティーはプロセスを完了しません。プロセスのナビゲーションは、終了状態に達するまで続行します。

ビジネス・プロセスでの障害処理

プロセスで障害が発生すると、ナビゲーションが障害ハンドラーまたは障害リンクで続行されます。

障害ハンドラーは invoke アクティビティー、スコープ、およびプロセスに対して指定できます。障害リンクは汎用 flow アクティビティーに指定できます。スコープおよび throw アクティビティーと rethrow アクティビティーを除くすべての基本アクティビティーは、障害リンクのソース・アクティビティーにすることができます。

障害ハンドラーまたは障害リンクでは、特定の障害名、障害タイプ、またはそれら両方を catch できます。障害が発生すると、Business Flow Manager は以下のルールを使用して、そのエンクロージング・スコープまたは障害が発生したアクティビティーの障害ハンドラーまたは障害リンクと、発生した障害を突き合わせます。

- 障害ハンドラーがない invoke アクティビティーまたはその他の基本アクティビティーが 1 つ以上の障害リンクのソースになっている場合、Business Flow Manager は一致する障害リンクを探します。障害リンクが使用可能でない場合は、エンクロージング・スコープで一致する障害ハンドラーを探します。
- 1 つ以上の障害ハンドラーを持つ invoke アクティビティーまたはスコープが 1 つ以上の障害リンクのソースになっている場合、Business Flow Manager は一致する障害ハンドラーを探します。障害ハンドラーが使用可能でない場合は、デフォルトの障害ハンドラーを実行した後、一致する障害リンクを探します。一致する障害リンクが使用可能でない場合は、エンクロージング・スコープで一致する障害ハンドラーを探します。
- その障害に関連する障害データがない場合、Business Flow Manager は、障害名が一致する障害ハンドラーまたは障害リンクを使用します。障害ハンドラーも障害リンクも見つからない場合は、catch all 障害ハンドラーまたは障害リンクを使用します (使用可能な場合)。障害変数が定義された障害ハンドラーおよび障害リンクでは、データを持たない障害を catch することはできません。
- その障害に関連する障害データがある場合、Business Flow Manager は、障害名が一致し、障害データのタイプに一致するタイプの障害変数を持つ障害ハンドラーまたは障害リンクを使用します。障害名と障害データ・タイプに一致する障害ハンドラーまたは障害リンクが見つからない場合は、障害データのタイプに一致するタイプの障害変数と障害名を持たない障害ハンドラーまたは障害リンクを使用します。適切な障害ハンドラーも障害リンクも見つからない場合は、catch all 障害ハンドラーまたは障害リンクを使用します (使用可能な場合)。障害変数が定義されていない障害ハンドラーおよび障害リンクでは、データを持つ障害を catch することはできません。

いずれの障害ハンドラーまたは障害リンクの定義にも一致しない障害が発生した場合は、デフォルトの障害ハンドラーが開始されます。デフォルトの障害ハンドラーは、明示的には指定されません。デフォルトの障害ハンドラーは、直接エンクロージング・スコープに対して使用可能なすべての補正ハンドラーを、対応するスコープの完了順序と逆の順序で実行します。スコープが 1 つ以上の障害リンクのソースになっている場合、Business Flow Manager は一致する障害リンクを探します。一致する障害リンクが使用可能でない場合、またはスコープがいずれの障害リンクのソースでもない場合は、デフォルトの障害ハンドラーが障害を次のレベル (エンクロ

ージング・スコープまたはプロセス) に再スローします。この次のレベルで、Business Flow Manager は、使用可能な障害ハンドラーまたは障害リンクに対して障害の突き合せを再試行します。

障害ハンドラー、障害リンク、catch all 障害ハンドラー、catch all 障害リンクのいずれによっても障害が catch されない場合、障害はプロセス・スコープに到達し、プロセスが失敗状態で終了します。障害ハンドラーがプロセス・スコープで障害を catch し、それを処理する場合でも、プロセスは失敗状態で終了します。

障害ハンドラーの考慮事項

障害ハンドラーを定義するときには、以下のオプションを考慮してください。

- 障害を catch して問題を修正し、ビジネス・プロセスを正常に完了するまで続行させる。
- 障害を catch し、このスコープでは解決不可能であると判断する。この場合、以下の追加オプションがあります。
  - 新規の障害をスローする。
  - 元の障害を再スローして、別のスコープでそれを処理できるようにする。
  - 要求応答操作である場合は、障害応答で応答する。
  - ヒューマン・タスクを起動して問題を修正する。
  - microflow の場合、障害ハンドラーで問題を解決できなければ、プロセスをロールバックして補正する必要があることがあります。
  - 長期実行プロセスの場合は、プロセスに対して「**エラーの継続**」設定を使用して、障害を管理的に処理することも考慮してください。

障害リンクの考慮事項

ビジネス・プロセスで障害リンクを使用する場合は、以下の点を考慮してください。

- ソース・アクティビティー内で障害が発生した場合に限って障害リンクをアクティブ化してください。通常のリンクの条件の評価は、アクティビティー実行の一部ではありません。
- 障害リンクのソース・アクティビティーが scope アクティビティーである場合、スコープ内で障害が発生すると、最初に scope アクティビティーの障害ハンドラーが評価されます。しかし、障害ハンドラーは障害を再スローすることができます。この場合は、スコープの障害リンクがその障害を catch し、ナビゲートできます。
- 1 つのアクティビティーが複数の障害リンクのソースになっている場合、障害が発生したときにナビゲートできる障害リンクはいずれか 1 つだけです。
- 障害リンクのターゲット・アクティビティーは通常どおりに実行されます。障害ハンドラーの compensate アクティビティーおよび rethrow アクティビティーを障害リンクのターゲットにすることはできません。
- 障害が障害データを含む場合は、囲んでいるスコープで障害データ・タイプの変数を宣言する必要があります。障害リンクがこの変数を参照していなければ、障害リンクのターゲット・アクティビティーが障害データにアクセスすることはできません。

トランザクションの考慮事項

一般に、現行トランザクションに参加しいている呼び出されたサービスによって実行時例外が発生した場合、現行トランザクションは、トランザクション開始前のプロセスの状態にロールバックします。トランザクションが Business Flow Manager によってトリガーされた場合、トランザクションは自動的に再試行され、サービスが再度呼び出されます。トランザクションが再試行されると、Business Flow Manager は、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却下して、実行時例外の原因となっているアクティビティーを判別するために追加のトランザクション境界を導入することができます。呼び出されたサービスで繰り返し実行時例外が発生する場合、その実行時例外は別のトランザクションに保管されます。次に再試行したときに、サービスが再度呼び出されることはありませんが、保管された例外が復元され、ナビゲーションでは障害処理が続行されます。

ビジネス・プロセスの障害データの取得

プロセスは、ランタイム障害および BPEL 標準の障害を処理できます。これらの障害を処理するために、障害に関する情報にアクセスする必要がある場合があります。

この情報を取得するには、以下のいずれかの構成体を使用できます。

- getCurrentFaultAsException メソッド

  障害ハンドラーで getCurrentFaultAsException メソッドを使用すると、ランタイム障害、BPEL 標準の障害、およびビジネス障害のデータを取得できます。catch all 障害ハンドラーには障害データを取り込むための変数が関連付けられていないため、このメカニズムは、catch all 障害ハンドラーとの組み合わせで役立ちます。また、runtimeFailure 障害を catch する場合にも役立ちます。

  getCurrentFaultAsException メソッドは、Java 断片アクティビティーで呼び出すことができます。このメソッドでは、com.ibm.bpe.api.BpelException タイプの例外オブジェクトとして障害が返されます。 BpelException オブジェクトでは、障害名などの障害に関する詳細情報を取得するための操作が提供されます。
  BpelException オブジェクトは例外インスタンスをラップします。したがって、以下の例に示すようにして障害メッセージとルート例外にアクセスできます。

  ```
  com.ibm.bpe.api.BpelException bpelexception =
  getCurrentFaultAsException();
  System.out.println("Fault  Name" +
  bpelexception.getFaultName())
  bpelexception.printStackTrace( System.out);
  Throwable rootCause = bpelexception.getRootCause()
  ```

- 障害ハンドラーまたは障害リンクの型付き障害変数

  ランタイム障害および BPEL 標準の障害の場合は、障害ハンドラーまたは障害リンクに型付きの障害変数を定義して障害データを取り込むことができます。障害変数の型は StandardFaultType 複合タイプでなければなりません。

関連資料

788 ページの『障害ハンドラーが障害を catch しない』
invoke アクティビティーには、呼び出されたサービスによってスローされる特定の障害を catch するために障害ハンドラーが関連付けられています。しかし、呼び出されたサービスが予期される障害を返したのに、障害ハンドラーが実行されません。

アクティビティーおよびビジネス・プロセスのエラー動作の継続

ビジネス・プロセスを定義するときには、予期しない障害が発生し、その障害に対して障害ハンドラーが定義されていない場合の動作を指定できます。プロセスを定義するときに「**エラーの継続**」設定を使用すると、障害が発生した場合にプロセスを停止するように指定できます。

ほとんどのアクティビティーでは、「エラーの継続」動作はプロセスの場合と同じです。「エラーの継続」動作は、invoke、Java 断片、カスタム、ヒューマン・タスクのアクティビティーに対して明示的に指定できます。デフォルトでは、これらのアクティビティーの「エラーの継続」動作は、プロセスの場合と同じです。予期しない障害が検出された場合、アクティビティーの障害処理が開始されます。「**エラーの継続**」設定が `Yes` に設定されている場合、標準の障害処理が適用されます。アクティビティーまたはプロセスの「**エラーの継続**」設定が `No` に設定されている場合に、直接エンクロージング・スコープの障害ハンドラーまたはこのアクティビティーから離れる障害リンクによって障害が処理されないと、そのアクティビティーは停止されます。障害ハンドラーまたは補正ハンドラーがアクティビティーに関連付けられている場合、直接エンクロージング・スコープはそのアクティビティー自体です。それ以外の場合、直接エンクロージング・スコープは、そのアクティビティーが含まれている 1 つ外側のスコープです。直接エンクロージング・スコープで catch all 障害リンクまたは障害ハンドラーが定義されている場合は、常に障害が処理され、アクティビティーが停止することがないので、「**エラーの継続**」設定の値は効力を持ちません。

予期しない障害が原因でアクティビティーが停止した場合は、アクティビティーの **stopReason** プロパティーを使用して障害の原因を判別し、修復に使用するアクションを決定できます。障害の場合に **stopReason** プロパティーが取る可能性がある値を次の表に示します。

stopReason プロパティーの値	**原因**	**許可される アクション**
STOP_REASON_ ACTIVATION_ FAILED	アクティビティーの結合条件の評価が失敗しました。強制結合条件を使用して、条件を true または false に設定するか、再評価することができます。	• 強制再試行 • 強制結合条件

stopReason プロパティーの値	**原因**	**許可される アクション**
STOP_REASON_ IMPLEMENTATION_ FAILED	アクティビティーの実装で障害がスローされました。	
	この値は、アクティビティーの実装が失敗した場合に設定されます。以下に例を示します。 • invoke アクティビティーによって呼び出されたサービスが障害ハンドラーによって処理できない障害を戻した。 • wait アクティビティーがアクティブ化されたときに timeout 式が失敗した。 • アクティビティーの出口条件の評価が失敗した。	• 強制再試行 • 強制完了
	while または repeatUntil 条件の評価が失敗した場合、kind プロパティーの値は または です。	• 強制再試行 • 強制完了 • 強制ループ条件
	forEach カウンターまたは条件の評価が失敗した場合、kind プロパティーの値は、KIND_FOR_EACH_SERIAL または KIND_FOR_ EACH_PARALLEL のいずれかです。	• 強制再試行 • 強制完了 • 強制 forEach カウンター値
	choice アクティビティー (switch アクティビティーとも呼ばれる) の分岐式の評価が失敗しました。	強制分岐ナビゲーション
STOP_REASON_ FOLLOW_ON_ NAVIGATION_ FAILED	発信リンクの遷移条件の評価が失敗しました。この値は以下の状況のいずれかのときに設定されます。 • 並列 flow (並列アクティビティー・アクティビティーとも呼ばれます) で、アクティビティーが完了した後、発信リンクの遷移条件が評価され、いずれか 1 つの条件で障害が発生した場合。 • 循環 flow で、後続のナビゲーションに対して適正な発信リンクが存在しない場合。	• 強制発信リンク・ナビゲーション • 強制完了

stopReason プロパティーの値	**原因**	**許可される アクション**
STOP_REASON_ EXIT_CONDITION_FALSE	出口条件の基準が満たされませんでした。この値は、アクティビティーの出口で出口条件を評価したときに、条件が `false` と評価された場合にのみ設定されます。	• 強制再試行 • 強制完了

停止したアクティビティーは、Business Process Choreographer API の force または skip メソッドのいずれかにより修復できます。あるいは、Business Process Choreographer Explorer の該当アクションによっても修復できます。また、修復アクションを実行する前にアクティビティー変数を変更できます。API メソッドおよび Business Process Choreographer Explorer アクションを次の表にまとめます。

表 3.

修復アクション	**API メソッド**	**Business Process Choreographer Explorer の アクション**
強制再試行	forceRetry	**再始動**
強制完了	forceComplete	**強制完了**
強制結合条件	forceJoinCondition	**結合の修復**
強制ループ条件	forceLoopCondition	**次の反復**または**ループの終了**
強制 forEach カウンター値	forceForEachCounterValues	**forEach の修復**
choice アクティビティーでの強制分岐ナビゲーション	`forceNavigate(..., int positionOfBranch)`	**強制ケース・ナビゲーション**または**強制ケース実行**
強制発信リンク・ナビゲーション	`forceNavigate(..., ... linksToBeFollowed)`	**強制ナビゲーション**
強制完了してジャンプ	forceCompleteAndJump	**ソース・アクティビティーを強制完了してジャンプ**
スキップ	skip	**スキップ**または**アクティビティーのスキップ**
スキップしてジャンプ	skipAndJump	**ソース・アクティビティーをスキップしてジャンプ**

関連概念

13 ページの『アクティビティーの状態遷移図』
アクティビティー・インスタンスの実行中に重要なステップが発生すると、アクティビティー・インスタンスの状態は変化します。状態および状態遷移はアクティビティーのタイプによって異なります。

関連タスク

566 ページの『ヒューマン・タスク・アクティビティーの処理』
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。

573 ページの『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。

575 ページの『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。

ビジネス・プロセスでの補正処理

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・インスタンスにおける、障害処理の手段の 1 つです。補正は、障害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。

関連概念

31 ページの『Microflow のトランザクションの振る舞い』
Microflow は短期存続プロセスです。トランザクション内で実行される場合と、Microflow の SCA コンポーネントに指定されているようにアクティビティー・セッション内で実行される場合があります。ここでは、トランザクションの一部として実行される Microflow について説明します。

Microflow での補正処理

Microflow の補正は、*テクニカル補正* とも呼ばれます。このタイプの補正は、Microflow が含まれているトランザクション、つまりアクティビティー・セッションがロールバックされた場合に起動されます。

通常、取り消しアクションは、トランザクション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、プロセス・モデルで指定されます。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してください。

補正処理中に障害が発生した場合、補正アクションでは、障害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。

長期実行プロセスでの補正処理

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。補正ロジックは、スコープ・レベルで定義するか、invoke アクティビティーまたはヒューマン・タスク・アクティビティーに対して定義することができます。すなわち、単一の invoke アクティビティーまたはヒューマン・タスク・アクティビティー、あるいはスコープ内の一連のアクティビティーを補正できます。

BPEL プロセスでは、補正は、スコープまたはプロセスの障害ハンドラーによってトリガーされるか、スコープまたは invoke アクティビティーの補正ハンドラーによってトリガーされます。

BPMN プロセスでは、補正は、補正アクティビティーを介して、外部の補正ハンドラーまたは障害ハンドラーによってトリガーすることができます。このサポートは、汎用 flow アクティビティーおよびコラボレーション・スコープで使用することができ、補正はフローまたはスコープ内からトリガーできます。この場合、補正アクティビティーは、補正アクティビティーが開始する前に正常に完了した、現行スコープ内のすべてのアクティビティーについて、補正をトリガーします。これらのアクティビティーがすべて補正されると、補正アクティビティーが完了し、プロセスのナビゲーションが続行されます。完了したアクティビティーの補正が失敗した場合、補正アクティビティーは、引き続き残りのアクティビティーの補正をトリガーします。これらのアクティビティーの補正が終了すると、補正アクティビティーは、失敗した補正の障害をエンクロージング・スコープへ渡します。

子プロセスの補正

長期実行プロセスは、正常終了した子プロセスも補正できます。 子プロセスは、invoke アクティビティーによって呼び出されます。この invoke アクティビティーには、補正ハンドラーを関連付けることができます。補正ハンドラーが定義されている場合、そのロジックにより、子プロセスの補正が補正アクティビティーによってトリガーされるかどうかが決まります。補正ハンドラーが補正アクティビティーを使用しない場合、子プロセスの補正は行われません。invoke アクティビティーに補正ハンドラーがない場合は、子プロセスの補正は自動的にトリガーされます。

長期間実行される子プロセスの場合、プロセス・レベルの直接ネストされたスコープのうち、正常に完了したものすべてに対して、完了した順序と逆の順序で補正が実行されます。microflow 子プロセスの場合、補正によって、呼び出しの転送実行の逆順で、すべての呼び出しのやり直しアクションが実行されます。

一般に、ヒューマン・タスク・アクティビティー、invoke アクティビティー (子プロセスを呼び出したアクティビティーを含む)、および scope アクティビティーで、正常に完了したもののみが補正されます。これらのアクティビティーの補正は、以下のいずれかの方法で実行できます。

- スコープ内に囲まれたすべてのものを補正する。

  子プロセスを開始したアクティビティーの補正は、スコープ内の他のアクティビティーの補正のシーケンスに統合されています。スコープ内のすべてのものの補正は、現行スコープのデフォルト補正ハンドラーによって、またはターゲットを指定しない補正アクティビティーによって、アクティブ化されます。

  例えば、アクティビティー A、B、および C がスコープ内に囲まれています。アクティビティー B は子プロセスの invoke アクティビティーであり、アクティビティー A および C は scope アクティビティーです。すべてのアクティビティーは順番に正常に完了します。親プロセスが補正されると、これらのアクティビティーは、完了した順序とは逆に、アクティビティー C、アクティビティー B、アクティビティー A の順序で補正されます。
- スコープ内の特定のアクティビティーの補正

  補正は、スコープまたはアクティビティーをターゲットとする補正アクティビティーによってアクティブ化されます。子プロセスを開始した invoke アクティビティーは、その invoke アクティビティーに補正ハンドラーが定義されていなくても、補正アクティビティーのターゲットにすることができます。

インフラストラクチャー障害からの回復

長期実行プロセスは、複数のトランザクションにわたっています。Business Flow Manager には、インフラストラクチャー障害が原因でトランザクションが失敗した場合に、それらの障害から自動的に回復するための機能が用意されています。

長期実行プロセスの場合、Business Flow Manager は、後続のナビゲーションを起動する要求メッセージを Business Flow Manager 自身に送信します。要求メッセージが着信するごとに、新しいトランザクションが開始され、要求メッセージが Business Flow Manager に渡されて処理されます。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。

- 要求メッセージを受信します。
- 要求に基づいてナビゲートします。
- 状態をデータベースに格納します。
- 後続のトランザクションを起動する要求メッセージを送信します。

Business Flow Manager は、インフラストラクチャー障害に対処するために以下のキューを使用します。

- 保存キューには、失敗したが自動的に再試行するメッセージを格納します。
- 保留キューには、失敗回数が再試行限度を超えたメッセージを格納します。これらのメッセージは、重大なインフラストラクチャー障害があることや、メッセージが壊れているために処理できないことを示している可能性があります。

メッセージが正常に処理された場合は、インフラストラクチャーが使用可能であると推測されます。しかし、Business Flow Manager は、以下の場合にもメッセージの処理に失敗することがあります。

原因	応答
インフラストラクチャーが使用不可	通常の処理モードでは、インフラストラクチャーが再度作動可能になるまで、指定した期間にわたってすべてのメッセージが有効な状態が維持されます。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。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。

インフラストラクチャーが使用不可であり、かつ保存キューがいっぱいである場合は、メッセージ処理が通常の処理から*静止モード* に切り替わります。静止モードでは、インフラストラクチャーが再度使用可能になるまで、メッセージの処理速度が低下します。インフラストラクチャーが使用可能になると、メッセージ処理が通常モードに戻ります。

通常のメッセージ処理

通常の処理では、以下のようにメッセージが処理されます。

- メッセージが 3 回失敗した場合は、保存キューに格納されます。
- 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された場合、保存キューにあるすべてのメッセージが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のカウントが保持されます。このカウントを*保存キューの巡回数* と呼びます。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。
  - 次のメッセージが失敗した場合、そのメッセージも保存キューに格納されます。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、すべてのメッセージが保存キューから入力キューに移動され、メッセージ処理が静止モードに切り替わります。

静止モードでのメッセージ処理

静止モードでは、メッセージの処理が定期的に試行されます。処理に失敗したメッセージは、配信回数や保存キューの巡回数を増加させることなく、入力キューに書き戻されます。メッセージを正常に処理できるようになると、ただちにメッセージ処理が通常のモードに戻ります。

再試行限度

再試行限度は、保留キューに書き込まれる前に保存キューへ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) 最後の再試行が開始されます。最後の再試行でさらに 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。再試行限度をゼロに設定すると、失敗を繰り返しているメッセージが 3 回まで再試行され、その後ただちに保留キューに移動します。

この Business Flow Manager のプロパティーを変更するには、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Business Flow Manager**」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。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。 Business Flow Manager 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。

この Business Flow Manager のプロパティーを変更するには、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Business Flow Manager**」をクリックします。

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、管理スクリプト、または Failed Event Manager を使用して実行できます。

ビジネス・プロセスのための許可

許可を使用して、特定の特権を特定ユーザーまたは特定のユーザー・グループに割り当てます。これにより、プロセスおよびアクティビティーに対してユーザーが実行を許可されるアクションが決まります。ビジネス・プロセスの許可は、ヒューマン・タスクを使用して行います。

許可のロールを使用して、特定のロールで使用可能な一連のアクションを定義します。Business Flow Manager は、ナビゲーションおよび許可を行う際にアクティビティーのロールを使用します。各アクティビティーのロールは、必ず 1 つのヒューマン・タスクのロールに一致します。ヒューマン・タスクに指定されているロールは、関連するビジネス・プロセスおよびアクティビティーによって継承されます。

したがって、例えば、インライン・ヒューマン・タスクをビジネス・プロセス内にモデル化する場合、タスクの所有者は自動的にアクティビティーの所有者になります。

ビジネス・プロセスのための許可のロール

ロールとは、同じ許可レベルを共有する担当者の集合です。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、Java EE 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。

ビジネス・プロセスのための Java EE ロール

Business Process Choreographer の構成時に、Java EE ロールが設定されます。Java EE ロール・ベースの許可を設定するには、ユーザー・レジストリーが構成されており、アプリケーション・セキュリティーが有効になっている必要があります。

プロセスでサポートされている Java 2 Platform, Enterprise Edition (Java EE) ロールを次に示します。

- 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。
- 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。
- JMSAPIUser。Business Flow Manager JMS API 要求は、呼び出し元には関係なく、このロールが割り当てられているユーザー ID のために実行されます。

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 Platform、Enterprise Edition (Java EE) 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 インフォメーション・センターの『ロール・ベースの許可の System Authorization Facility』を参照してください。

ビジネス・プロセスおよびアクティビティーのインスタンス・ベースの許可のロール

ビジネス・プロセスとアクティビティーには、事前定義の許可のロールのセットが用意されています。プロセスをモデル化するときに、これらのロールを割り当てることができます。インスタンス・ベースのロールとユーザーの関連付けは、実行時に担当者解決を使用して決定されます。

プロセスに対するアクションのための許可のロール

プロセスのロールに割り当てられた担当者には、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
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、このロールのメンバーは自動的に、アクティビティーと、ヒューマン・タスク・アクティビティーに関連付けられているインライン予定タスク (サブタスク、後続のタスク、エスカレーションなど) の読者になります。
プロセス管理者	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、作業項目を作成、削除、および転送し、アクティビティーのスキップなどによって、実行時のプロセスのナビゲーションを変更することができます。また、このロールのメンバーは自動的に、アクティビティーと、ヒューマン・タスク・アクティビティーに関連付けられているインライン予定タスク (サブタスク、後続のタスク、エスカレーションなど) の管理者になります。
プロセス・アクティビティー管理者	プロセスにおけるアクティビティーを修復します。
スコープ・リーダー	スコープにおけるアクティビティーおよび変数のプロパティーを表示します。また、このロールのメンバーは自動的に、スコープ内のアクティビティーと、ヒューマン・タスク・アクティビティーに関連付けられているインライン予定タスク (サブタスク、後続のタスク、エスカレーションなど) のプロパティーの読者になります。
スコープ管理者	アクティビティーの変数更新、アクティビティーのスキップ、スキップ要求の取り消しなど、スコープ内のアクティビティーを管理します。また、このロールのメンバーは自動的に、スコープ内のアクティビティーと、ヒューマン・タスク・アクティビティーに関連付けられているインライン予定タスク (サブタスク、後続のタスク、エスカレーションなど) の管理者になります。

プロセス・スターターは、Business Flow Manager でプロセス・ナビゲーションと外部サービスの呼び出しに使用されるロールです。プロセス・インスタンスがデータベースにまだ存在している場合は、プロセスのナビゲーションを続行できるようにするため、ユーザー・レジストリーからプロセス・スターターのユーザー ID を削除しないでください (ただし、プロセスの所有権を別のユーザーに移転した場合はこの限りではありません)。

ヒューマン・タスクを使用して、これらのロールにユーザーが割り当てられます。

ロール	担当者割り当て
プロセス・スターター	プロセス・スターターを指定するには、プロセスの開始 receive アクティビティーまたは開始 pick (receive choice) アクティビティーにインライン・ヒューマン・タスクを割り当てます。
プロセス・リーダー	プロセス・リーダーを指定するには、プロセスに関連付けられている管理タスクで読者のロールを設定します。このロールは、プロセス内のすべてのアクティビティーに継承されます。
プロセス管理者	プロセス管理者は、プロセスに割り当てられている管理タスクにより定義されます。このロールは、プロセス内のすべてのアクティビティーに継承されます。
プロセス・アクティビティー管理者	プロセス・アクティビティー管理者は、プロセスに関連付けられている管理タスクにより定義されます。このタスクで定義される管理者のロールは、プロセス・アクティビティー管理者としても使用されます。 **注:** 管理タスクは、プロセス管理者決定のためのタスクとは異なります。 プロセス・レベルで定義されているアクティビティー管理タスクは、管理タスクが定義されていないアクティビティーのデフォルト管理タスクです。
スコープ・リーダー	スコープ・リーダーを指定するには、スコープに関連付けられている管理タスクで読者のロールを設定します。このロールは、スコープ内のすべてのアクティビティーに継承されます。
スコープ管理者	スコープ管理者は、スコープに割り当てられている管理タスクにより定義されます。このロールは、スコープ内のすべてのアクティビティーに継承されます。

注: プロセス管理がシステム管理者に制限される場合、インスタンス・ベースの管理は使用不可になります。つまり、プロセス、スコープ、およびアクティビティーに対する管理アクションは BPESystemAdministrator ロールのユーザーに制限されます。さらに、プロセス・インスタンスまたはその一部の読み取り、表示、およびモニターを実行できるのは、BPESystemAdministrator ロールまたは BPESystemMonitor ロールのユーザーのみになります。この管理モードについて詳しくは、58 ページの『代替プロセス管理許可モード』を参照してください。

アクティビティーに対するアクションのための許可のロール

ヒューマン・タスクをモデル化し、ビジネス・プロセスにヒューマン・タスク・アクティビティーとして組み込むと、そのタスクの所有者が自動的にアクティビティー所有者になります。ヒューマン・タスクに対して定義されているロールのメンバーは、対応するヒューマン・タスク・アクティビティーで同じロールを継承します。Business Flow Manager は、ナビゲーションおよび許可を行う際にアクティビティーのロールを使用します。インライン呼び出しタスクの可能なスターターは、関連付けられている receive または pick (receive choice) アクティビティー、またはイベント・ハンドラーの可能なスターターです。

アクティビティーのインスタンス・ベース・ロールには、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復し、長期実行アクティビティーを強制的に完了します。

プロセスのロールのデフォルトの担当者割り当て

特定のロールに担当者割り当て基準を定義しない場合、または担当者解決が失敗するか、結果を戻さない場合は、デフォルトの担当者割り当てが実行されます。以下の表に、適用されるデフォルトのスタッフ割り当てを示します。

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 ...
プロセス管理者	プロセス・スターターがプロセス管理者になる
プロセス・リーダー	リーダーなし

また、ビジネス・プロセスを作成および開始する呼び出しタスクを定義しないと、プロセスの可能なスターターに対し、デフォルトの担当者割り当て基準である **Everybody** が使用されます。

ビジネス・プロセスの作成および開始の許可

プロセスの作成および開始を許可される一連のユーザーの決定は、新規プロセス・インスタンスの作成および開始に使用する receive または pick (receive choice) アクティビティーに関連付けられている呼び出しタスクによるほか、プロセスに関連付けられている管理タスクによります。ビジネス・プロセスは、これらのタスクに割り当てられているロールを継承します。

さらに、プロセスのインストール時に特定の Service Component Architecture (SCA) セキュリティー修飾子を設定することにより、SCA クライアントによってプロセスが呼び出されたときにプロセスを開始できるユーザーのセットを制限できます。

ビジネス・プロセスを作成および開始するには、以下のようにヒューマン・タスクを使用します。

- インライン呼び出しタスクを、プロセスの receive または pick (receive choice) アクティビティーの開始に割り当てます。

ビジネス・プロセスによっては、重要なビジネス・データを変更する場合があるため、許可されたユーザーのみがこれらのプロセスを作成および開始する権限を持つようにします。このようなタイプのビジネス・プロセスの場合、プロセス・テンプレートに対してインライン呼び出しタスクを指定することによって、プロセスの receive アクティビティーの開始にヒューマン・タスクを割り当てることができます。インライン呼び出しタスクに定義された潜在的スターターが、プロセスの潜在的スターターになります。

プロセスは、Human Task Manager API を使用して呼び出しタスクを作成および開始することによって開始することも、Business Flow Manager API を使用することによって開始することもできます。どちらのメソッドでも、同様に許可検査が行われます。インライン・タスクを指定していない場合、誰でもプロセスを開始できます。

- スタンドアロン呼び出しタスクを、プロセスの receive または pick (receive choice) アクティビティーの開始に割り当てます。

  ビジネス・プロセスに結合されたスタンドアロン呼び出しタスクを使用して、プロセスの開始時に許可検査を実行することもできます。ただし、スタンドアロン呼び出しタスクを使用する場合は、以下の点を考慮してください。

  – 許可検査が実行されるのは、プロセスが呼び出しタスクによって開始される場合のみです。つまり、Business Flow Manager API、またはプロセス・コンポーネントに直接接続された SCA クライアントを使用してプロセスを開始する場合には、検査が省略されます。
  – SCA インフラストラクチャーを使用してプロセスを呼び出します。これに対し、インライン・タスクは Business Flow Manager によって直接呼び出されます。
  – 担当者割り当て基準定義内のプロセス・コンテキストにアクセスできません。つまり、スタンドアロン・タスクでは、プロセス・コンテキストに基づく動的な担当者割り当てはサポートされません。

プロセスに管理タスクが割り当てられている場合は、管理タスクの管理者ロールがプロセスによって継承されます。プロセス管理者は、プロセス・インスタンスの作成や開始などのさまざまなアクションを、プロセスに対して実行できます。

関連概念

『ビジネス・プロセスと対話するための許可』

長期実行プロセスは、複数の receive アクティビティー、pick (receive choice) アクティビティー、およびイベント・ハンドラーを持つことができます。これらは、対応するプロセス・インスタンスの該当する操作に要求をサブミットすることによって提供されます。プロセス・インスタンスは、プロセス・モデルで定義された相関セットに従って、固有の相関セット・インスタンスを要求時に提供することによって、暗黙のうちに識別されます。

ビジネス・プロセスと対話するための許可

長期実行プロセスは、複数の receive アクティビティー、pick (receive choice) アクティビティー、およびイベント・ハンドラーを持つことができます。これらは、対応するプロセス・インスタンスの該当する操作に要求をサブミットすることによっ

て提供されます。プロセス・インスタンスは、プロセス・モデルで定義された相関セットに従って、固有の相関セット・インスタンスを要求時に提供することによって、暗黙のうちに識別されます。

receive アクティビティーまたは pick アクティビティーを使用すると、プロセス・インスタンスを作成できます。そのため、要求をプロセスにサブミットすることで既存のプロセス・インスタンスと対話するのは、新しいプロセス・インスタンスを開始するのと似ています。

プロセス・インスタンスに要求をサブミットできる権限を持つユーザーは、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・ハンドラーに関連付けられた呼び出しタスクと、プロセスに関連付けられた管理タスクによって決定されます。

プロセス・インスタンスと対話するために、以下の方法でヒューマン・タスクを使用できます。

- インライン呼び出しタスクを 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・ハンドラーに割り当てます。

  インライン呼び出しタスクに定義された潜在的スターターが、プロセスの対応する操作に要求をサブミットします。呼び出しタスクはオプションです。呼び出しタスクが定義されていない場合は、要求をサブミットする権限がすべてのユーザーに付与されます。
- スタンドアロン・ヒューマン・タスクを使用して、ビジネス・プロセスのインバウンド操作を保護することもできます。プロセス作成の操作に対するスタンドアロン呼び出しタスクの場合と同じルールおよび制約事項が適用されます。
- 管理タスクをプロセスに割り当てます。

  管理タスクの管理者ロールがプロセスによって継承されます。プロセス管理者は、その操作を使用してプロセスと対話できます。

  プロセスに管理タスクが指定されていない場合は、プロセスのスターターがプロセス管理者になります。この場合、プロセスのスターターがプロセス・インスタンスの操作に対する要求をサブミットできます。

別の receive アクティビティー、pick (receive choice) アクティビティー、またはイベント・ハンドラーと同じ操作をプロセスが使用しているときに、対応する receive アクティビティーまたは pick (receive choice) アクティビティーがまだ待機中であるかイベント・ハンドラーがまだアクティブでないために受信側プロセス・インスタンスが要求を受信しようとしていない場合は、要求を送信する側のユーザーが、これらすべてのアクティビティーおよびイベント・ハンドラーに要求を送信する権限を持っている必要があります。さもなければ要求は拒否されます。

関連概念

54 ページの『ビジネス・プロセスの作成および開始の許可』
プロセスの作成および開始を許可される一連のユーザーの決定は、新規プロセス・インスタンスの作成および開始に使用する receive または pick (receive choice) アクティビティーに関連付けられている呼び出しタスクによるほか、プロセスに関連付けられている管理タスクによります。ビジネス・プロセスは、これらのタスクに割り当てられているロールを継承します。

ビジネス・プロセスを管理するための許可

管理タスクを使用して、ビジネス・プロセスおよびそれに関連するアクティビティーに対して管理アクションを実行する権限を、ユーザーまたはユーザーのグループに与えることができます。

注: プロセス管理がシステム管理者に制限される場合、インスタンス・ベースの管理は使用不可になり、プロセス、スコープ、およびアクティビティーに対するすべての管理アクションは BPESystemAdministrator ロールのユーザーに制限されます。この管理モードについて詳しくは、 58 ページの『代替プロセス管理許可モード』を参照してください。

プロセス管理

管理アクションの実行を許可するユーザーを定義し、プロセス・データを読み取るために、長期実行ビジネス・プロセスの一部として管理タスクを指定できます。管理タスクに対する管理者および読者のロールは、プロセス管理者とプロセス・リーダーとするユーザーを決定します。プロセス管理者は、例えばプロセス・インスタンスを強制終了させることができます。

管理タスクはすべてのビジネス・プロセスに関連付けられます。プロセスに対して管理タスクがモデル化されていない場合は、実行時にデフォルトの管理タスクが作成されます。このデフォルトのタスクは、プロセス・スターターをプロセス管理者として定義し、そのプロセスに対してどの読者も割り当てません。

スコープ管理

スコープ・リーダーおよびスコープ管理者を定義するスコープの管理タスクは、モデル化できます。スコープ・リーダーは、ローカル変数を表示できます。スコープ管理者は、スコープ内のアクティビティー・インスタンスを修復し、ローカル変数を表示および更新できます。スコープが別のスコープに囲まれている場合、スコープ・リーダーおよび管理権限はエンクロージング・スコープに継承されます。また、スコープ・リーダーおよび管理者はスコープ内のアクティビティーの読者および管理者になります。

アクティビティー管理

アクティビティー管理タスクの管理者ロールにより、対応するアクティビティーの管理を許可される担当者が決まります。アクティビティー管理者は、例えば、アクティビティーを再始動できます。アクティビティー・インスタンスに対して管理アクション (つまり再始動や完了) を実行できるようになると即時に、管理タスクが作成されます。プロセスの読者および管理者ロール、エンクロージング・スコープの読者および管理者ロールは、自動的にアクティビティーに伝搬されます。

また、次のように、アクティビティーに対する管理タスクをモデル化できます。

- invoke アクティビティーまたは断片アクティビティーごとに管理タスクをモデル化。この管理タスクにより、プロセス管理者以外にアクティビティーの管理を許可される担当者が決まります。
- アクティビティーに対するデフォルトの管理タスクをプロセス・レベルでモデル化。これを、管理タスクが割り当てられていないすべてのアクティビティーに適用します。

代替プロセス管理許可モード

管理タスクを組み込むようにプロセス・テンプレートをモデル化した場合は、この管理モードによって、それらの管理タスクおよび関連アクション (エスカレーションやモニターなど) が非アクティブ化されます。このモードで実行すると、インスタンス・ベースの管理が使用不可になると共に、プロセス管理およびモニターがシステム管理者またはシステム・モニターであるユーザーに制限されるので、パフォーマンスが向上します。

Business Flow Manager のカスタム・プロパティー `ProcessAdministration` が値 `useSystemAdminAuthorizationOnly` に設定されると、新規プロセス・インスタンス開始時の処理や、プロセス・インスタンス、スコープ・インスタンス、およびアクティビティー・インスタンスに対する管理アクションの表示、モニター、または実行を許可されるユーザー ID について、別のルール・セットが適用されます。

この代替管理モードの目的は、Business Process Choreographer データベースに作成されるオブジェクトの数を減らしてパフォーマンスを向上させることであり、これによってタスク・リスト照会およびプロセス・リスト照会の応答時間を短縮できます。

この管理モードの影響を受けるのは、新規に作成されるオブジェクトのみであり、既存のインスタンスは、インスタンス作成時に適用された管理ルールに従います。管理アクションを実行する担当者や自動化プロセスが、適切なロールではないユーザー ID を使用したために管理アクションの実行を拒否されることがないように、注意してください。

代替管理許可モードによって、以下の変更が実施されます。

プロセス・インスタンス

新規プロセス・インスタンスの開始時には、それを開始したのがユーザーであっても別のコンポーネントであっても、そのプロセス・インスタンスの管理タスク・インスタンスは作成されません。プロセス・インスタンスに対して作成される唯一の作業項目は、プロセス・スターター作業項目です。

管理 BPESystemAdministrator ロールのユーザーのみが、プロセス・インスタンスに対して管理アクションを実行できます。例えば、障害が発生したプロセス・インスタンスを強制終了または再始動したり、グローバル変数またはローカル変数の内容を更新したりできます。

表示およびモニター

BPESystemAdministrator ロールまたは BPESystemMonitor ロールのユーザーのみが、プロセス・インスタンスまたはその一部を表示あるいはモニターできます。例えば、Business Process Choreographer Explorer でプロセス・インスタンスの進行状況を表示したり、

Business Flow Manager API を使用してプロセス・インスタンスに属する変数の内容を読み取ったりできます。

スコープ・インスタンス

管理タスクが関連付けられているスコープがアクティブ化されるときには、管理タスク・インスタンスも作業項目も作成されません。

管理 BPESystemAdministrator ロールのユーザーのみが、スコープ・インスタンスに対して管理アクションを実行できます。例えば、スコープ内のあるアクティビティーから別のアクティビティーにジャンプしたり、スコープ内でアクティビティーのスキップ、強制再試行、または強制完了を実行したりできます。

表示およびモニター

BPESystemAdministrator ロールまたは BPESystemMonitor ロールのユーザーのみが、スコープ・インスタンスまたはその一部を表示あるいはモニターできます。

アクティビティー・インスタンス

例えば、実装における不具合によってアクティビティー・インスタンスが停止したなどの理由で、そのアクティビティー・インスタンスに対して管理アクションを実行する必要がある場合は、管理タスクも作業項目も作成されません。

管理 BPESystemAdministrator ロールのユーザーのみが、アクティビティー・インスタンスに対して管理アクションを実行できます。例えば、停止したアクティビティー・インスタンスに対して、強制完了、強制ナビゲート、または強制再試行を実行できます。

表示およびモニター

BPESystemAdministrator ロールまたは BPESystemMonitor ロールのユーザーのみが、アクティビティー・インスタンスまたはその一部を表示あるいはモニターできます。例えば、アクティビティー・インスタンスに属する変数の内容を読み取ることができます。

次のいずれかの条件がご使用のシステムに当てはまる場合、この管理モードはニーズに合わない可能性があります。

- 必要な管理タスクを組み込むようにプロセス・テンプレートをモデル化した場合は、この管理モードを有効にすると、管理タスクおよびそれに関連するすべての機能 (例えば、エスカレーション) が非アクティブ化されてしまいます。
- 上記の管理タスクを実行する必要があるすべてのユーザーを BPESystemAdministrator ロールに追加すると、それらのユーザーに必要以上の権限が付与される可能性があります。

関連タスク

プロセス管理をシステム管理者に制限する

デフォルトのインスタンス・ベース管理をオフにすると、プロセス・インスタンス、スコープ・インスタンス、およびアクティビティー・インスタンスを管理できるのは BPESystemAdministrator ロールのユーザーのみになります。さらに、プロセス・インスタンスを表示またはモニターできるのは、BPESystemMonitor ロールのユーザーのみになります。これにより、データベースの負荷および成長率が低下するので、パフォーマンスを向上させることができます。

第 2 章 ヒューマン・タスクの概要

ヒューマン・タスクは、人とサービスの対話を可能にするコンポーネントです。

いくつかのヒューマン・タスクは、人の予定を表します。これらのタスクは、人または自動化されたサービスのいずれかによって開始することができます。ヒューマン・タスクを使用すると、例外の手動処理や承認など、人による対話が必要なアクティビティーをビジネス・プロセスに実装することができます。他のヒューマン・タスクを使用して、サービスを呼び出したり、人と人の間のコラボレーションを調整したりできます。ただし、タスクの開始方法に関係なく、ユーザーのグループに属し、タスクが割り当てられている個人がタスクに関連付けられている作業を実行します。

ヒューマン・タスクへのユーザーの割り当ては、静的に行われるか、または実行時に担当者ディレクトリーから解決される、ロールやグループなどの基準を指定することにより行われます。あるいは、ヒューマン・タスクの入力データ、またはビジネス・プロセスのデータを使用して、タスクでの作業に適切なユーザーを見つけます。

タスク・テンプレート

ヒューマン・タスク・テンプレートには、WebSphere Integration Developer を使用して作成されたか、または Business Process Choreographer API を使用して実行時に作成されたデプロイ済みタスク・モデルの定義が含まれています。

このテンプレートには、タスク名や優先順位などのプロパティーが含まれており、エスカレーション・テンプレート、カスタム・プロパティー、担当者照会テンプレートなどの成果物が集約されています。タスク・テンプレートのモデル化時に指定されるプロパティーの他に、インストール済みのタスク・テンプレートも、以下のいずれかの状態になります。

開始 タスク・テンプレートが開始されると、そのテンプレートの新規インスタンスを開始できます。

停止 ヒューマン・タスク・アプリケーションをアンインストールするには、その前にタスク・テンプレートを停止する必要があります。タスク・テンプレートが停止状態にある場合は、このテンプレートの新規インスタンスを開始することはできません。

com.ibm.task.api.TaskModel クラスのインスタンスを作成し、実行時に予定タスクまたはコラボレーション・タスクをモデル化することができます。次に、これらのインスタンスを使用して再使用可能なタスク・テンプレートを作成するか、1 回実行のタスク・インスタンスを直接作成できます。実行時のヒューマン・タスクのモデル化は、Eclipse Modeling Framework (EMF) に基づいています。

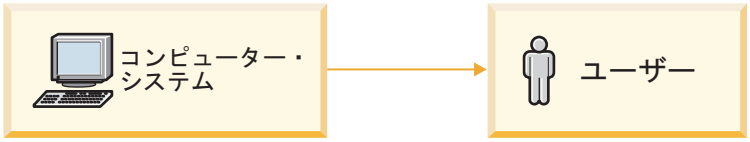
ヒューマン・タスクの種類

タスクの種類は、モデル化中に割り当てられるタスク・テンプレートの種類から派生します。

ヒューマン・タスクの種類には以下のものがあります。

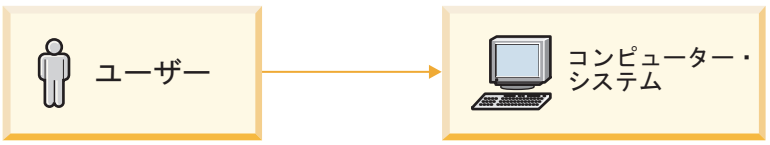
予定タスク

サービス・コンポーネント (ビジネス・プロセスなど) で、これから実行するべきタスクを 1 人以上の担当者に割り当てる場合は、この種のタスクを使用します。予定タスクは、スタンドアロン・タスクとしてもインライン・タスクとしても実装できます。



呼び出しタスク

担当者がサービス・コンポーネントにタスクを「割り当てる」場合は、この種のタスクを使用します。この場合、担当者は自動化サービス (ビジネス・プロセスなど) を呼び出します。



呼び出しタスクは、スタンドアロン・タスクとしてもインライン・タスクとしても実装できます。インラインの呼び出しタスクの場合、担当者は、アクティビティー (receive アクティビティーや pick アクティビティーなど) またはイベント・ハンドラーを通じてビジネス・プロセスが公開している操作を呼び出すことができます。

コラボレーション・タスク

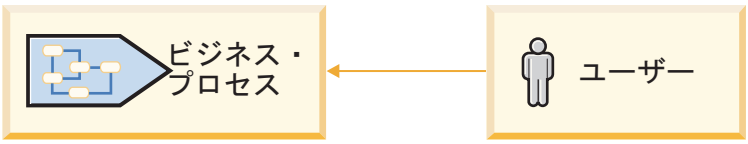
担当者が 1 人以上の担当者にタスクを割り当てる場合は、この種のタスクを使用します。この種のタスクを使用すれば、構造化されて制御された方法で、担当者間で作業を分担できます。

コラボレーション・タスクは、他のコンポーネントとの対話がないためので、スタンドアロン・タスクになります。この自己完結型のタスクによって、他のサービスに対する参照やインターフェースのないスタンドアロンの人的対話が実装されます。

管理タスク

担当者に管理能力 (ビジネス・プロセスの中断、終了、再始動、強制再実

行、強制完了などの権限) を与える場合は、この種のタスクを使用します。管理タスクは、invoke アクティビティーでセットアップすることも、プロセス全体でセットアップすることもできます。



この種のタスクは、ビジネス・プロセスの中でのみ使用できます (つまり、インライン・タスクになります)。

関連概念

97 ページの『予定タスクの状態遷移図』
予定タスクは、クライアント・アプリケーションによって、またはコンポーネントを呼び出すことによって自動的に作成されます。予定タスクは、作業をビジネス・プロセスの一部として実行するとき (インライン・タスク)、または公開して提供する Web サービスを実装するとき (スタンドアロン・タスク) に、ユーザーをサポートします。予定タスクのライフ・サイクル内では、特定の対話は特定のタスク状態でのみ可能であり、これらの対話が今度はタスクの状態に影響を与えます。

102 ページの『コラボレーション・タスクの状態遷移図』
コラボレーション・タスクは、担当者が他の担当者の作業を実行するときに使用できるタスクです。コラボレーション・タスクのライフ・サイクル内では、特定の対話は特定のタスク状態でのみ可能であり、これらの対話が今度はタスクの状態に影響を与えます。

106 ページの『呼び出しタスクの状態遷移図』
呼び出しタスクは、担当者がサービスを呼び出すときに使用できるタスクです。呼び出しタスクのライフ・サイクル内では、特定の対話は特定のタスク状態でのみ可能であり、これらの対話が今度はタスクの状態に影響を与えます。

108 ページの『管理タスクの状態遷移図』
管理タスクは、ビジネス・プロセスとそのアクティビティーを管理するユーザーをサポートします。管理タスクのライフ・サイクル内では、特定の対話は特定のタスク状態でのみ可能であり、これらの対話が今度はタスクの状態に影響を与えます。

77 ページの『スタンドアロン・タスクおよびインライン・タスク』
サービス指向アーキテクチャー (SOA) パターンでは、疎結合コンポーネントのセットを使用してソフトウェア・ソリューションを実現することが推奨されています。SOA パターンに従うヒューマン・タスクがスタンドアロン・タスク と呼ばれるのに対し、ビジネス・プロセスの一部として定義されているヒューマン・タスクはインライン・タスク と呼ばれます。

ヒューマン・タスクのバージョン管理

同じタスクの複数のバージョンが 1 つのランタイム環境に共存するように、ヒューマン・タスクの新規バージョンを作成することができます。

スタンドアロン・ヒューマン・タスクを WebSphere Integration Developer でモデル化するときに、バージョン管理情報を組み込むことができます。タスクのバージョンは、その有効開始日によって決まります。つまり、あるタスクの異なるバージョンに同じタスク名を付けることはできますが、それらの有効開始日はそれぞれ異な

っています。実行時に使用されるタスクのバージョンは、そのタスクが*早期バインディング・シナリオ*と*実行時バインディング・シナリオ*のどちらで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・シナリオでは、使用されるタスクのバージョンは、モデリング中またはタスク・モデルのデプロイ時に決定されます。呼び出し側コンポーネントは、Service Component Architecture (SCA) 結合に従って、静的にバインドされた専用のタスクを呼び出します。有効開始日に照らして有効な、タスクの別のバージョンが存在している場合でも、現行の静的に結合されているタスクが使用され、その他のすべてのバージョンは無視されます。

早期バインディングの例は SCA ワイヤーです。スタンドアロン参照をヒューマン・タスク・コンポーネントに結合すると、この参照を使用するタスクの呼び出しはすべて、ヒューマン・タスク・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・シナリオでは、使用されるヒューマン・タスクは、タスク・インスタンスが作成されるときに決定されます。この場合は、現在有効なタスクのバージョンが使用されます。タスクの新しいバージョンが、以前のどのテンプレート・バージョンよりも優先されます。既存のタスク・インスタンスは、開始時に関連付けられたタスクを使用して、継続して実行されます。そのため、タスクは以下のようなカテゴリーに分けられます。

- 新規タスク・インスタンスで使用される現在有効なタスク
- 有効ではなくなったが、実行中のタスク・インスタンスに対してはまだ有効になっているタスク
- 有効開始日に従って将来的に有効になるタスク

実行時バインディングの例としては、Business Process Choreographer Explorer で新規タスクが呼び出される場合があります。作成されるインスタンスは常に、有効開始日付が将来の日付ではないタスクの最新バージョンに基づいています。後続タスクおよびサブタスクは常に、実行時バインディングを使用して呼び出されます。

タスク・インスタンス

タスク・インスタンスとは、実行時に発生するタスク・テンプレートのインスタンスです。

一般にタスク・インスタンスは、以下の例外を除いて、すべてのプロパティーを対応するタスク・テンプレートから継承します。

TASK_TEMPL ビューでの列名	タスク・インスタンスによる継承	コメント
VALID_FROM	いいえ	タスク・インスタンスでは必要ありません。

TASK_TEMPL ビューでの列名	タスク・インスタンスによる継承	コメント
CONTAINMENT_CTX_ID	いいえ	タスク・インスタンスは、対応するタスク・テンプレートと異なる一連のルールに従って削除されます。
IS_AD_HOC	いいえ	以下のタスク・インスタンスでは必要ありません。 • 随時タスク・テンプレートに、随時ではないタスク・インスタンスがあります。 • 随時タスク・インスタンスはタスク・テンプレートを持ちません。
IS_INLINE	通常	以下の場合、プロパティーは継承されません。 • テンプレートがインラインと定義されている場合でも、サブタスク・インスタンスはインライン化できません。 • テンプレートがインラインと定義されている場合でも、後続タスク・インスタンスはインライン化できません。 • ヒューマン・タスク・アクティビティー・インスタンスは、常にインライン・タスク・インスタンスに関連付けられます。
STATE	いいえ	タスク・インスタンスを作成して開始するには、タスク・テンプレートが STATE_STARTED 状態でなければなりません。その後、インスタンスが STATE_READY 状態になります。

さらに、タスク・テンプレート (TASK_TEMPL_CPROP ビュー) のカスタム・プロパティーは、すべてタスク・インスタンス (TASK_CPROP ビュー) のカスタム・プロパティー・インスタンスによって継承されます。タスク・テンプレート (TASK_TEMPL_DESC ビュー) のマルチリンガル記述では、ロケールごとに 1 行が対応しています。タスク・インスタンス (TASK_DESC ビュー) はそれらの行を継承します。

実行時のタスク・インスタンス・プロパティーの変更

実行時に、タスク・インスタンスのプロパティー (名前、タスクを操作できるユーザー、および期間の設定など) が、テンプレートからコピーされます。通常、タスク・インスタンスは、そのライフ・サイクルを通じて、これらのプロパティーの事

前定義値を保持します。ただし、ビジネス・ニーズに応じて、実行時にこれらの事前定義値をオーバーライドすることが必要な場合もあります。

さらに、いくつかのプロパティーの事前定義値には、他のプロパティーへの参照が含まれている可能性があります。この場合それらは置換変数式と呼ばれます。

Business Process Choreographer Explorer、Business Space、および Business Process Choreographer API では、実行時にタスク・インスタンスを変更する操作がサポートされています。

実行時に変更できるプロパティー

Business Process Choreographer API を使用して、タスク・インスタンスの以下のプロパティーを変更できます。これらのプロパティーの一部は、Business Process Choreographer Explorer または Business Space を使用して変更するともできます。

- ビジネス関連性
- 所有者のコンテキスト許可
- 削除時刻
- 説明
- 表示名
- 期限
- エスカレーション状態
- イベント・ハンドラー名
- 有効期限時刻
- 名前
- 名前空間
- 親コンテキスト ID
- 優先順位
- 読み取り
- 中断した場合の要求をサポート
- 代行をサポート
- 後続のタスクをサポート
- サブタスクをサポート
- タイプ
- ワーク・バスケット

実行時のタスクの有効期限時刻、削除時刻、および期限の変更

ビジネス・シチュエーションで、タスクに元から定義されている期限、有効期限時刻、または削除時刻を変更しなければならない場合があります。それらの時刻うちで実行時にスケジュール変更、キャンセル、開始できるものはどれか、およびどのような場合にそれらのアクションを実行できるかは、タスク状態によって決まります。Business Process Choreographer Explorer を使用してこれらの時刻を変更することも、Human Task Manager API の update メソッドを使用して、適切なタスク・プロパティーを変更することもできます。

以下のいずれかの方法で、タスクの期限、有効期限時刻、または削除時刻を変更できます。

- 特定の時刻を使用する。
- ビジネス・カレンダーなどのカレンダーに基づいて期間 (例えば 2 日間) を使用する。また、期間の式で以下の定数を使用することもできます。

  DURATION_ZERO
  タスクは、開始後に直ちに期限切れまたは満了となるか、(タスクの自動削除設定およびタスク終了状態に応じて) 完了後に直ちに削除されます。タスクが開始されている場合、そのタスクは直ちに期限切れになるか、または直ちに満了になります。タスクが終了状態の場合、そのタスクに対して自動削除が設定されていれば、直ちに削除されます。

  DURATION_INFINITE
  タスクが期限切れにならないか、満了しないか、または削除されません。

タスク・チェーンの場合、チェーン内のそれぞれのタスクに固有の期限が設定されます。ただし、有効期限時刻が設定されるのはチェーンの最初のタスクのみであり、その有効期限時刻がチェーン内の他のすべてのタスクによって共有されます。サブタスクには固有の期限と有効期限時刻が設定されます。該当するタスクの種類で削除がサポートされている場合は、削除時刻を変更することができます。

期限

dueTime プロパティーまたは durationUntilDue プロパティーを使用して、タスクの期限を変更することができます。タスクの状態によって、これらのうちのどちらのプロパティーを使用できるかが決まります。

非アクティブ状態
この状態では、durationUntilDue プロパティーを、タスクで使用されるカレンダーにおける有効な値、またはいずれかの定数値に設定することによって、タスクに対して定義されている「期限までの期間 (duration until due)」の値をオーバーライドすることができます。タスクの優先度が高いなどの理由で、開始直後にタスクが期限切れになるように指定するには、durationUntilDue プロパティーを DURATION_ZERO に設定します。タスクが期限切れにならないようにするには、durationUntilDue プロパティーを DURATION_INFINITE に設定します。

アクティブ状態 (作動可能、要求済み、実行中、または転送済み)
タスクがこれらのいずれかの状態である場合は、期限をスケジュール変更またはキャンセルすることができます。また、タスクが非アクティブ状態の間に DURATION_INFINITE が設定されたなどの理由で、タスクに期限が設定されていない場合は、期限を設定することができます。

dueTime プロパティーまたは durationUntilDue プロパティーのいずれかを、タスクで使用されるカレンダーにおける有効な値に設定することによって、期限を設定またはスケジュール変更することができます。期限をキャンセルするには、durationUntilDue プロパティーを DURATION_INFINITE に設定します。タスクが緊急の顧客要求に関連しているなどの理由で、タスクが直ちに強制的に期限切れになるようにするには、durationUntilDue プロパティーを DURATION_ZERO に設定します。

有効期限時刻

expirationTime プロパティーまたは durationUntilExpires プロパティーを使用して、タスクの有効期限時刻を変更することができます。タスクの状態によって、これらのうちのどちらのプロパティーを使用できるかが決まります。

非アクティブ状態

この状態では、durationUntilExpires プロパティーを、タスクで使用されるカレンダーにおける有効な値、またはいずれかの定数値に設定することによって、タスクに対して定義されている「有効期限時刻までの期間 (duration until expires)」の値をオーバーライドすることができます。タスクが開始直後に満了するように指定する場合は、durationUntilExpires プロパティーを DURATION_ZERO に設定します。タスクが満了しないようにするには、durationUntilExpires プロパティーを DURATION_INFINITE に設定します。

アクティブ状態 (作動可能、要求済み、実行中、または転送済み)

タスクがこれらのいずれかの状態である場合は、有効期限時刻をスケジュール変更またはキャンセルすることができます。また、タスクが非アクティブ状態の間に DURATION_INFINITE が設定されたなどの理由で、タスクに有効期限時刻が設定されていない場合は、有効期限時刻を設定することができます。

expirationTime プロパティーまたは durationUntilExpires プロパティーのいずれかを、タスクで使用されるカレンダーにおける有効な値に設定することによって、有効期限時刻を設定またはスケジュール変更することができます。有効期限時刻をキャンセルするには、durationUntilExpires プロパティーを DURATION_INFINITE に設定します。タスクが不要になったなどの理由で、タスクを直ちに強制的に満了させるには、durationUntilExpires プロパティーを DURATION_ZERO に設定します。

削除時刻

削除時刻になったときにタスクが削除されるかどうかは、タスクの自動削除設定およびタスクの終了状態によって決まります。deletionTime プロパティーまたは durationUntilDeleted プロパティーを使用して、タスクの削除時刻を変更することができます。タスクの状態によって、これらのうちのどちらのプロパティーを使用できるかが決まります。

非アクティブ状態またはアクティブ状態 (作動可能、要求済み、実行中、または転送済み)

タスクがこれらのいずれかの状態である場合、durationUntilDeleted プロパティーをいずれかの定数値に設定することによって、そのタスクに対して定義されている「削除までの期間 (duration until deleted)」の値をオーバーライドすることができます。durationUntilDeleted プロパティーを DURATION_ZERO に設定することによって、いずれかの終了状態に達したタスクを直ちに削除することができます。削除時刻をキャンセルするには、durationUntilDeleted プロパティーを DURATION_INFINITE に設定します。

終了状態 (終了、失敗、転送済み、強制終了、期限切れ)

タスクがこれらのいずれかの状態である場合は、削除時刻をスケジュール変更またはキャンセルすることができます。また、タスクが非アクティブ状態

の間に DURATION_INFINITE が設定されたなどの理由で、タスクに削除時刻が設定されていない場合は、削除時刻を設定することができます。

deletionTime プロパティーまたは durationUntilDeleted プロパティーのいずれかを、タスクで使用されるカレンダーにおける有効な値に設定することによって、削除時刻を設定またはスケジュール変更することができます。削除時刻をキャンセルするには、durationUntilDeleted プロパティーを DURATION_INFINITE に設定します。タスクが直ちに強制的に削除されるようにするには、durationUntilDeleted プロパティーを DURATION_ZERO に設定します。

実行時のエスカレーションのタイミングの変更

ビジネス・シチュエーションで、エスカレーションを定義したときに指定したエスカレーションのタイミングを変更することが必要な場合があります。エスカレーション状態によって、それらのうちどの時刻を変更できるか、およびそれらのアクションをいつ実行できるかが決まります。Human Task Manager API の update メソッドを使用して、適切なエスカレーション・プロパティーを変更できます。Business Space でタスク・リスト・ウィジェットまたはエスカレーション・リスト・ウィジェットを使用して、スケジュールされたエスカレーション時刻をオーバーライドし、エスカレーションを直ちに開始することもできます。

以下のいずれかの方法で、初期エスカレーションのタイミング、およびエスカレーションが反復される条件を変更することができます。

- 特定の時刻を使用する。
- ビジネス・カレンダーなどのカレンダーに基づいて期間 (例えば 2 日間) を使用する。また、期間の式で以下の定数を使用することもできます。

DURATION_INFINITE
初期エスカレーションまたは反復エスカレーションがキャンセルされます。

初期エスカレーション

escalationTime プロパティーまたは durationUntilEscalated プロパティーを使用して、エスカレーションのタイミングを変更することができます。エスカレーションの状態によって、これらのうちのどちらのプロパティーを使用できるかが決まります。

非アクティブ状態
この状態では、durationUntilEscalated プロパティーを、タスクで使用されるカレンダーにおける有効な値に設定することによって、エスカレーションに対して定義されている「エスカレートまでの期間」の値をオーバーライドすることができます。エスカレーションが開始されないようにするには、durationUntilEscalated プロパティーを DURATION_INFINITE に設定します。

待ち状態
この状態では、初期エスカレーションをスケジュール変更またはキャンセルすることができます。escalationTime プロパティーまたは durationUntilEscalated プロパティーのいずれかを、タスクで使用されるカレンダーにおける有効な値に設定することによって、エスカレーションのスケ

ジュールを変更することができます。エスカレーションをキャンセルするには、durationUntilEscalated プロパティーを DURATION_INFINITE に設定します。

また、Human Task Manager API の triggerEscalation メソッドを使用して、エスカレーションを手動で開始することもできます。

反復エスカレーション

タスクがエスカレーション済み状態になっており、期待されるタスク状態にまだ到達していない場合は、エスカレーションの状態に応じて escalationTime プロパティーまたは durationUntilRepeated プロパティーを使用して、反復エスカレーションのタイミングを変更することができます。

非アクティブかつ待ち状態

この状態では、durationUntilRepeated プロパティーを、タスクで使用されるカレンダーにおける有効な値に設定することによって、エスカレーションに対して定義されている「反復までの期間」の値をオーバーライドすることができます。エスカレーションが開始されないようにするには、durationUntilRepeated プロパティーを DURATION_INFINITE に設定します。

エスカレーション済み状態

この状態では、エスカレーションの反復をスケジュール変更またはキャンセルすることができます。escalationTime プロパティーまたは durationUntilRepeated プロパティーのいずれかを、タスクで使用されるカレンダーにおける有効な値に設定することによって、エスカレーションの反復のスケジュールを変更することができます。反復をキャンセルするには、durationUntilRepeated プロパティーを DURATION_INFINITE に設定します。

関連概念

93 ページの『エスカレーション』
エスカレーションとは、ヒューマン・タスクに対するアクションが指定された時間内に実行されない場合に自動的に発生するアラートです。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 つ以上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したりすることができます。

ヒューマン・タスクの置換変数

置換変数は、実行時に解決されるエレメントの値を参照するためにヒューマン・タスクの定義で使用されます。これらの変数は、タスクに割り当てられた担当者や、タスクのカスタム・プロパティーなどの、タスクおよびプロセス関連のデータを表します。このデータは、実行時に、タスク・インスタンスのライフ・サイクルの全体または一部で使用できます。

特定のタスク・エレメントのみが、置換変数を含むことができます。これらのタスク・エレメントは、タスク・テンプレートに対して、または Business Process Choreographer API を使用して実行時に作成されたタスク・モデルに対して定義され

ます。これらの定義は、タスク・インスタンスが作成されると、そのタスク・インスタンスによって継承されます。以下のタスク・エレメントに、置換変数を組み込むことができます。

- 削除されるまでの期間
- 満了までの期間
- 期限切れまでの期間
- エスカレーションのカスタム・プロパティー
- エスカレーションの説明
- エスカレーションの所要時間
- エスカレーション通知 E メールの E メール件名および本文
- 管理者の担当者割り当て
- 編集者の担当者割り当て
- エスカレーション受信者の担当者割り当て
- 潜在的インスタンス作成者の担当者割り当て
- 潜在的所有者の担当者割り当て
- 読者の担当者割り当て
- 潜在的スターターの担当者割り当て
- タスクのカスタム・プロパティー
- タスクの説明
- タスクの優先順位
- タスク・タイプ

ヒューマン・タスク・エレメントの置換変数を初期化または変更するアクション:

ここでは、特定のタスク関連アクションが完了する前に初期化されているものとみなせる Human Task Manager 置換変数を示します。これらのタスク関連アクションを実行することによって、置換変数の初期化とタスク・エレメントの評価の両方を同じタスク・アクションで完了することができます。

以下の表に、ヒューマン・タスクで使用可能な置換変数の要約を示します。この表には、タスク・アクションの実行時における置換変数の計算とタスク・エレメントの評価の両方についての初期化順序が示されています。

表 4. 置換変数を初期化または変更するアクション

置換変数	初期化アクション	変更アクション
htm:input.*[part|part¥XPath|¥XPath]*	タスクを作成または開始するすべてのアクション、setInputMessage メソッド	タスクを作成または開始するすべてのアクション、setInputMessage
htm:output.*[part|part¥XPath|¥XPath]*	complete メソッドまたは setOutputMessage メソッド	complete メソッドまたは setOutputMessage
htm:task.administrators	タスクを開始するすべてのアクション	管理者作業項目の転送、管理者ロールの担当者照会の再実行
htm:task.description	タスクを作成または開始するすべてのアクション	新しい説明によるタスクの更新

表 4. 置換変数を初期化または変更するアクション (続き)

置換変数	初期化アクション	変更アクション
htm:task.displayName	タスクを作成または開始するすべてのアクション	新しい表示名によるタスクの更新
htm:task.editors	タスクを開始するすべてのアクション	編集者作業項目の転送、編集者ロールの担当者照会の再実行
htm:task.instanceID	タスクを作成するすべてのアクション	なし
htm:task.originator	タスクを作成するすべてのアクション	オリジネーター作業項目の転送
htm:task.owner	タスクを要求するすべてのアクション	所有者作業項目の転送、タスクの要求を取り消すすべてのアクション
htm:task.potentialInstanceCreators	タスクを開始するすべてのアクション	潜在的インスタンス作成者ロールの担当者照会の再実行
htm:task.potentialOwners	タスクを開始するすべてのアクション	潜在的所有者作業項目の転送、潜在的所有者ロールの担当者照会の再実行
htm:task.potentialStarters	タスクを作成するすべてのアクション	潜在的スターター作業項目の転送、潜在的スターター・ロールの担当者照会の再実行
htm:task.property.*customPropertyName*	タスクを開始するすべてのアクション、または task.setCustomProperty メソッド	task.setCustomProperty メソッド
htm:task.readers	タスクを開始するすべてのアクション	読者作業項目の転送、読者ロールの担当者照会の再実行
htm:task.starter	タスクを開始するすべてのアクション	スターター作業項目の転送
htm:task.URLPrefix	ヒューマン・タスク・コンテナーの開始	管理コンソールでのヒューマン・タスク・コンテナー構成の変更
htm:task.URLPrefixAdmin	ヒューマン・タスク・コンテナーの開始	管理コンソールでのヒューマン・タスク・コンテナー構成の変更
htm:task.URLPrefixBPCExplorer	ヒューマン・タスク・コンテナーの開始	管理コンソールでのヒューマン・タスク・コンテナー構成の変更
htm:escalation(*escalationName*).receivers	エスカレーションがエスカレーション済み状態になる	指定されたエスカレーションのエスカレーション受信者作業項目の転送、エスカレーション受信者ロールの担当者照会の再実行
htm:escalation.activationState	エスカレーションの作成	なし
htm:escalation.description	エスカレーションの作成	なし
htm:escalation.displayName	エスカレーションの作成	なし
htm:escalation.expectedTaskState	エスカレーションの作成	なし
htm:escalation.instanceID	エスカレーションの作成	なし
htm:escalation.property.*customPropertyName*	エスカレーションの作成、または escalation.setCustomProperty メソッド	escalation.setCustomProperty メソッド
htm:escalation.receivers	エスカレーションがエスカレーション済み状態になる	現在のエスカレーションのエスカレーション受信者作業項目の転送、エスカレーション受信者ロールの担当者照会の再実行
htm:escalation.URLPrefix	ヒューマン・タスク・コンテナーの開始	管理コンソールでのヒューマン・タスク・コンテナー構成の変更

表 4. 置換変数を初期化または変更するアクション (続き)

置換変数	初期化アクション	変更アクション
htm:escalation.URLPrefixBPCExplorer	ヒューマン・タスク・コンテナーの開始	管理コンソールでのヒューマン・タスク・コンテナー構成の変更
wf:activity(*activityName*).editor	指定されたアクティビティーに属するインライン・タスクの作成	指定されたアクティビティーの編集者作業項目の転送、編集者ロールの担当者照会の再実行
wf:activity(*activityName*).owner	指定されたアクティビティーに属するインライン・タスクの作成	要求、要求のキャンセル、または指定されたアクティビティーの所有者作業項目の転送
wf:activity(*activityName*). potentialOwners	指定されたアクティビティーに属するインライン・タスクの作成	指定されたアクティビティーの潜在的所有者作業項目の転送、潜在的所有者ロールの担当者照会の再実行
wf:activity(*activityName*).reader	指定されたアクティビティーに属するインライン・タスクの作成	指定されたアクティビティーの読者作業項目の転送、読者ロールの担当者照会の再実行
wf:process.administrators	プロセスの開始	プロセスまたはプロセス管理タスクのプロセス管理者作業項目の転送、管理者ロールの担当者照会の再実行
wf:process.readers	プロセスの開始	プロセスまたはプロセス管理タスクのプロセス読者作業項目の転送、読者ロールの担当者照会の再実行
wf:process.starter	プロセスの開始	なし
wf:property.*customPropertyName*	プロセスの開始	setCustomProperty メソッド
wf:variable.[*variableName*]¥ *messagePartName* [¥*XPath*]	アクティビティーの結果、または processInstance.setVariable メソッド	アクティビティーの結果、または processInstance.setVariable メソッド

実行時のタスク・エレメントおよびその置換変数の評価:

タスク・エレメントを照会することにより、クライアント・アプリケーションはタスク関連情報を公開できます。これらのエレメントの定義に置換変数を含めることができます。これらの変数が実行時に値に置き換えられるためには、いくつかの条件を満たす必要があります。

変数が値に置き換えられるためには、以下の両方の条件を満たす必要があります。

1. タスク・エレメントが評価される前に変数が初期化されている
2. クライアント・アプリケーションからの照会の前にタスク・エレメントが評価される

次の表に、特定のタスク関連アクションが完了する前に初期化されているものとみなせる Human Task Manager 置換変数を示します。これらのタスク関連アクションを実行することによって、置換変数の初期化とタスク・エレメントの評価の両方を同じタスク・アクションで完了することができます。

表 5. タスク関連アクションあと、タスク・エレメントおよび置換変数の評価の順序

タスク関連アクション	評価の順序
タスクの作成	1. タスク・エレメント: 潜在的インスタンス作成者の担当者割り当てが評価される a. 置換変数: htm:task.potentialInstanceCreators が初期化される 2. タスク・エレメント: 潜在的スターターの担当者割り当てが評価される a. 置換変数: htm:task.potentialInstanceStarters が初期化される b. 置換変数: htm:task.instanceID が初期化される c. 置換変数: htm:task.displayName が初期化される d. 置換変数: htm:task.property.*customPropertyName* が初期化される 3. タスク・エレメント: タスク優先順位が評価される a. 置換変数: htm:task.originator が初期化される 4. タスク・エレメント: タスクの説明が評価される a. 置換変数: htm:task.description が初期化される
タスクの開始	1. タスク・エレメント: エレメントは評価されない a. 置換変数: htm:input.*[part|part¥XPath|¥XPath]* が初期化される 2. タスク・エレメント: タスクが期限切れになるまでの期間が評価される 3. タスク・エレメント: タスク満了までの期間が評価される 4. タスク・エレメント: 管理者ロールの担当者割り当てが評価される a. 置換変数: htm:task.administrators が初期化される 5. タスク・エレメント: 潜在的所有者の担当者割り当てが評価される a. 置換変数: htm:task.potentialOwners が初期化される 6. タスク・エレメント: 編集者の担当者割り当てが評価される a. 置換変数: htm:task.editors が初期化される 7. タスク・エレメント: 読者の担当者割り当てが評価される a. 置換変数: htm:task.readers が初期化される b. 置換変数: htm:task.starter が初期化される 8. タスク・エレメント: タスクのカスタム・プロパティーが評価される a. 置換変数: htm:task.property.*customPropertyName* が初期化される 9. タスク・エレメント: タスク・タイプが評価される 10. タスク・エレメント: タスク優先順位が評価される 11. タスク・エレメント: タスクの説明が評価される a. 置換変数: htm:task.description が初期化される

表 5. タスク関連アクションあと、タスク・エレメントおよび置換変数の評価の順序 (続き)

タスク関連アクション	評価の順序
エスカレーションが作成される	タスク・エレメントは評価されません。以下の置換変数が初期化されます。 • htm:escalation.instanceID • htm:escalation.activationState • htm:escalation.expectedTaskState • htm:escalation.displayName • htm:escalation.description • htm:escalation.property.*customPropertyName*
エスカレーションがアクティブ化される	1. タスク・エレメント: エスカレーションの所要時間が評価される 2. タスク・エレメント: エスカレーション・プロパティーが評価される a. 置換変数: htm:escalation.property.*customPropertyName* が初期化される 3. タスク・エレメント: エスカレーションの説明が評価される a. 置換変数: htm:escalation.description が初期化される
エスカレーションがトリガーされる	1. タスク・エレメント: エスカレーション受信者が評価される a. 置換変数: htm:escalation.receivers が評価される 他のエスカレーションのエスカレーション受信者がエレメントの定義で参照される場合があることに注意してください。例えば、htm:escalation(otherEscalationName).receivers などの変数が定義に含まれていることがあります。 2. タスク・エレメント: E メール通知の場合、E メール件名と本文が評価される
タスクが終了状態に達する	1. タスク・エレメント: タスクの削除までの期間が評価される

この表は、タスク・エレメントの Human Task Manager 置換変数を指定するときに従う必要がある以下の制限事項を示しています。

- タスク・エレメントに含めることができる置換変数は、そのタスク・エレメントが評価される前に初期化されている置換変数のみです。
  - タスク・エレメントが含まれている行よりも前の行に示されている置換変数は、以前のタスク・アクションによって初期化されています。

    例えば、置換変数 %htm:task.originator% は、タスクが作成されるときに初期化されます。したがって、タスク・インスタンスの開始時に評価されるタスク・エレメント 『People assignment for potential owners』 の定義にこの置換変数を含めることができます。
  - タスク・エレメントの評価と同じタスク・アクションの一環として、タスク・エレメントが評価される前に初期化されている置換変数は、同じ行に示されています。この表では、特定のタスク・アクション (タスクの作成、タスクの更新、エスカレーションのアクティブ化、エスカレーションの期限切れ) につい

て、タスク・エレメントと置換変数の両方の評価の順序が定義されています。この順序に基づいて、タスク・エレメントの評価の前にどの置換変数が初期化されるかがわかります。

例えば、置換変数 %htm:input.¥param1% は、タスクの開始時に、あらゆるタスク・エレメントが評価される前に初期化されます。したがって、タスク・インスタンスの開始時に評価されるタスク・エレメント 『People assignment for potential owners』 の定義に含めることができます。評価の順序が表の行で指定されていない場合、その行については特定の評価順序は保証されません。

- タスク・エレメントが評価されると、対応する置換変数が初期化されることがあります。
  - 一部のタスク・エレメントには、他のタスク・エレメントによって使用可能な、対応する置換変数があります。例えば、タスク・エレメント 『People assignment for potential owners』 には、対応する置換変数 %htm:task.potentialOwners% があり、このタスク・エレメントの評価の後に初期化されます。つまり、タスク・エレメントは、他のタスク・エレメントとの関連で定義される場合があります。

置換変数: タスク・エレメント内の使用パターンの例:

多くの異なる方法で置換変数を使用できます。実行時に、多くのソースからの値を変数に使用できます。例えば、以前のスタッフ解決やカスタム・プロパティーの値が使用されることがあり、インライン・タスクの場合は周囲のビジネス・プロセスの値が使用されることもあります。

タスクの説明にランタイム固有情報が含まれる

このパターンは、タスクの説明を照会するクライアント・アプリケーションに使用します。これらの説明には、タスクの作成時または開始時に初期化された置換変数が含まれます。このパターンでは、タスク・テンプレート内のタスク・エレメントに以下の定義が必要です。

表 6. タスクの説明にランタイム固有の情報を含めるためのタスク・テンプレート定義

タスク・エレメント	定義
タスクのカスタム・プロパティー	name: 'property1' value: 'a default value'
タスクの説明	"This task instance has as ID: %htm:task.instanceID% as originator: %htm:task.originator% as administrators: %htm:task.administrators% as owner: %htm:task.owner% a custom property with the name 'property1' which is set to %htm:task.property.*property1*%

タスク・インスタンスで setCustomProperty メソッドを使用する場合、タスク・インスタンス用に個別のカスタム・プロパティーを設定できます。タスクが開始すると、タスクの説明が評価され、クライアント・アプリケーションに表示される説明にこの変数が組み込まれます。

カスタム・プロパティーを持つタスクの所要時間を制御する

このパターンは、クライアント・アプリケーションがタスク・インスタンスの所要時間を制御できるようにするために使用します。このパターンでは、タスク・テンプレート内のタスク・エレメントに以下の定義が必要です。

表 7. タスクの所要時間を制御するためのタスク・テンプレート定義

タスク・エレメント	定義
タスク・カレンダー	'Simple'
タスクのカスタム・プロパティー	name: 'property1' value: '2days 3hours'
期限切れまでの期間	%htm:task.property.*property1*%

タスク・インスタンスで setCustomProperty メソッドを使用する場合、単純カレンダーで許可されるフォーマットで、タスク・インスタンス用にカスタム・プロパティーを設定できます。タスクが開始すると、期限切れまでの期間が評価され、この値が期間に挿入されます。

インライン・タスクの担当者割り当てを制御する

このパターンは、ビジネス・プロセス内の前のタスクの担当者割り当てに基づいて、タスク・インスタンスの担当者割り当てを制御するために使用します。このパターンでは、タスク・テンプレート内のタスク・エレメントに以下の定義が必要です。

表 8. インライン・タスクの担当者割り当てを制御するためのタスク・テンプレート定義

タスク・エレメント	定義
潜在的所有者ロールの担当者割り当て	Users by User ID userId: %wf:activity(*activity1*).owner%

置換変数で、*activity1* は、要求済み状態にあるビジネス・プロセス内のヒューマン・タスク・アクティビティーです。このことは、タスクの所有者が既知であることを意味します。2 番目のタスクが開始されると、潜在的所有者の担当者割り当てが評価されます。最初のタスクの所有者が、担当者割り当て式のパラメーター値として挿入されます。

スタンドアロン・タスクおよびインライン・タスク

サービス指向アーキテクチャー (SOA) パターンでは、疎結合コンポーネントのセットを使用してソフトウェア・ソリューションを実現することが推奨されています。SOA パターンに従うヒューマン・タスクが*スタンドアロン・タスク* と呼ばれるのに対し、ビジネス・プロセスの一部として定義されているヒューマン・タスクは*インライン・タスク* と呼ばれます。

以下の表は、スタンドアロン・タスクおよびインライン・タスクに使用可能なタスクの種類を示しています。

表 9.

実装	呼び出しタスク	予定タスク	コラボレーション・タスク	管理タスク
スタンドアロン	はい	はい	はい	いいえ
インライン	はい	はい	いいえ	はい

スタンドアロン・タスク

スタンドアロン・タスクはサービス指向アーキテクチャー (SOA) パターンに従っており、スタンドアロン・タスクを呼び出すコンポーネント (予定タスク)、またはスタンドアロン・タスクによって呼び出されるコンポーネント (呼び出しタスク) と疎結合しています。これらのタスクは、Service Component Architecture (SCA) インフラストラクチャーを使用している別のコンポーネントに結合できます。

スタンドアロン・タスクには、peer または child のいずれかの autonomy 設定があります。autonomy 設定が peer になっているスタンドアロン・タスクは、SCA によって、それぞれのパートナー・コンポーネントと排他的に通信します。すなわち、予定タスクは入力メッセージを受け取り、出力メッセージまたは障害メッセージを返し、呼び出しタスクは入力メッセージを送信して、出力メッセージまたは障害メッセージを受け取ります。それ以上の情報交換やライフ・サイクル制御は行われません。

スタンドアロン・タスクは個別にモデル化されるため、再使用が可能です。スタンドアロン・タスクでは、Common Event Infrastructure (CEI) イベントと監査ログ・イベントが常にヒューマン・タスク・イベントとして発行されます。

スタンドアロン・タスクは、以下の状態で SCA コンポーネントとして使用可能になります。

- 予定タスクが、クライアント・コンポーネントに結合できるインターフェースを持っている。
- 呼び出しタスクが、呼び出されるサービスに結合できる参照を持っている。
- コラボレーション・タスクが、内蔵タイプの SCA コンポーネントである。コラボレーション・タスクはスタンドアロン・タスクのコンポーネントですが、SCA 参照または SCA インターフェースがないため、他のサービス・コンポーネントには結合できません。代わりに、担当者が Human Task Manager API を使用してそれらを開始したり操作したりするためのインターフェースが用意されています。

インライン・タスク

インライン・タスクは、ビジネス・プロセスに不可欠な部分です。予定タスク、呼び出しタスク、または管理タスクをインライン・タスクとすることができます。コラボレーション・タスクは人と人との対話を利用し、プロセスとは直接対話しないため、コラボレーション・タスクをインライン・タスクにすることはできません。インライン・タスクを SCA コンポーネントとして表示することはできません (結合できません)。また、それらを他のプロセスやアクティビティーで再利用することもできません。

インライン・タスクは、プロセス変数、カスタム・プロパティー、およびアクティビティー・データなどのプロセス・コンテキストへのアクセス権限を持っています。このことは、業務の分離に関係するタスクに役立ちます。インライン予定タスクは、その CEI および監査ログ・イベントをビジネス・プロセス・アクティビティー・イベントおよびヒューマン・タスク・イベントとして発行できます。それぞれのサブタスクおよび後続タスクは、イベントをヒューマン・タスク・イベントとして発行します。

以下の規則は、インライン・タスクに適用されます。

- 予定タスクは、プロセスにおけるヒューマン・タスク・アクティビティーである。これらは同じ状態を共有しますが、ヒューマン・タスク・アクティビティーは転送済み状態およびタスクの副状態を反映しません。
- 呼び出しタスクは、receive アクティビティー、pick (receive choice) アクティビティー、または on-event イベント・ハンドラーに関連付けられる。
- 管理タスクは、プロセスまたはプロセス内のアクティビティーのいずれかに接続される。
- ライフ・サイクルは通常、プロセスによって決まる。
  - 予定タスクおよび管理タスクはビジネス・プロセスによって作成され、そのビジネス・プロセスと一緒に削除されます。
  - インライン予定タスク自体には有効期限はなく、対応するヒューマン・タスク・アクティビティーに有効期限が定義されています。ヒューマン・タスク・アクティビティーの有効期限が切れると、予定タスクは終了します。インライン予定タスクの更新およびスケジュール変更は、Human Task Manager API と Business Flow Manager API の両方でサポートされています。Human Task Manager API が使用されている場合、要求はヒューマン・タスク・アクティビティーに転送されます。
  - 呼び出しタスクがビジネス・プロセスによって作成され、開始される場合、呼び出しタスクのライフ・サイクルはそのビジネス・プロセスによって決まり、そのビジネス・プロセスと一緒に削除されます。呼び出しタスクが Human Task Manager API を使用して開始される場合、呼び出しタスクのライフ・サイクルは、作成された方法に関わらずプロセスに依存することはなく、その結果はプロセスが削除された後でも表示できます。
  - インライン・ヒューマン・タスクのインスタンスは、関連するプロセス・インスタンスと共にマイグレーションできます。
- 予定タスクおよび呼び出しタスクの説明、表示名、および資料は 1 つの言語のみをサポートする。
- インライン呼び出しタスクは、有効期限までの期間および削除されるまでの期間の両方の値を使用してモデル化できます。これらの設定は、タスクが Human Task Manager API を使用して作成される場合のみ使用可能です。これらの期間は、タスクが開始する前には更新することができ、タスクの開始後はスケジュールを変更できます。
- インライン・タスクに対する更新アクションは、タスク・プロパティーのサブセットのみをサポートする。更新が可能なのは、プロセスまたはアクティビティーに現れていないタスク・プロパティーのみです。update メソッドについて詳しくは、com.ibm.task.api パッケージにある HumanTaskManager インターフェースに

関する Javadoc API 資料と、どのロールがタスクに対する特定の更新アクションの実行を許可されているかに関する情報を参照してください。

インライン・タスクはプロセスの許可に使用されます。

- 予定タスクの読者、管理者、潜在的所有者、所有者、およびエディターのロールは、プロセス内のヒューマン・タスク・アクティビティーの対応するロールと同一です。
- インラインの呼び出しタスクの潜在的スターター・ロールにより、対応する receive アクティビティー、pick (receive choice) アクティビティー、または on-event イベント・ハンドラーを呼び出し、それにメッセージを送信することを許可される担当者が決まります。潜在的スターター・ロールおよび潜在的インスタンス作成者ロールは、同一の担当者割り当てを持っていることに注意してください。インライン呼び出しタスクが定義されていない場合、すべてのユーザーにアクティビティーまたはイベント・ハンドラーを開始する権限が付与されます。
- プロセス管理タスクの管理者ロールおよび読者ロールにより、プロセス管理者またはプロセス・リーダーが決まります。プロセス管理者は、例えばプロセス・インスタンスを強制終了できます。
- アクティビティー管理タスクの管理者ロールにより、対応するアクティビティーの管理を許可される担当者が決まります。アクティビティー管理者およびプロセス管理者は、例えばアクティビティーを強制的に再試行できます。
- プロセス・リーダーおよびプロセス管理者権限は、すべてのプロセス・アクティビティーまたはインラインのヒューマン・タスクによって継承されます。
- スコープ・リーダーおよびスコープ管理者の権限は、スコープ内のすべてのアクティビティーに継承されます。

注: プロセス管理がシステム管理者に制限される場合、インスタンス・ベースの管理は使用不可になります。つまり、プロセス、スコープ、およびアクティビティーに対する管理アクションは BPESystemAdministrator ロールのユーザーに制限されます。さらに、プロセス・インスタンスまたはその一部の読み取り、表示、およびモニターを実行できるのは、BPESystemAdministrator ロールまたは BPESystemMonitor ロールのユーザーのみになります。この管理モードについて詳しくは、58 ページの『代替プロセス管理許可モード』を参照してください。

ヒューマン・タスクとビジネス・プロセスの関係

インライン・タスクは、関連するプロセスを認識しており、プロセスは、自身のインライン・タスクを認識しています。スタンドアロン予定タスクの場合は、その関係を child の autonomy 属性で定義できます。ビジネス・プロセスの invoke アクティビティーによって直接インスタンス化される子タスクは、そのビジネス・プロセスのライフ・サイクルに関わります。つまり、終了や削除などのライフ・サイクル操作は、ビジネス・プロセスから子タスクに伝搬します。アクティビティーの有効期限が切れると、予定タスクは終了します。

呼び出しタスクは、receive アクティビティー、pick (receive choice) アクティビティー、または on-event イベント・ハンドラーに関連付けることができます。これらのタスクは、インライン・タスクにすることも、スタンドアロン・タスクにすることもできます。Business Flow Manager API を使用している場合は、インライン呼び出しタスクのみが、receive アクティビティーまたは pick アクティビティーを呼び出

すための許可に影響を与える可能性があります。デフォルトでは、すべてのユーザーが receive アクティビティー、pick アクティビティー、または on-event イベント・ハンドラーにメッセージを送信することが許可されています。これには、receive アクティビティーまたは pick アクティビティーを開始する場合のビジネス・プロセスの呼び出しも含まれます。

管理タスクはすべてのビジネス・プロセスに関連付けられます。管理タスクにより、プロセスの管理と読み取りを許可される担当者が決まります。WebSphere Integration Developer でプロセスの管理タスクがモデル化されていない場合は、実行時に管理タスクが作成されます。そのタスクによって、ビジネス・プロセスのデフォルトの許可が設定されます。つまり、プロセス・スターターがプロセスの唯一の管理者になり、そのプロセスに読者は割り当てられません。

管理タスクは、invoke アクティビティーまたは断片アクティビティーごとにモデル化できます。このタスクにより、プロセス管理者以外にアクティビティーの管理を許可される担当者が決まります。明示的な管理タスクが割り当てられていないすべての invoke アクティビティーまたは断片アクティビティーに適用される、デフォルトのアクティビティー管理タスクをモデル化することもできます。

invoke アクティビティーには、管理タスクが関連付けられています。断片アクティビティーおよび同期 invoke アクティビティーの場合、このタスクは、そのアクティビティーが呼び出しの失敗の後に停止したときにのみ作成されます。その後に、管理タスクを使用して、強制終了や強制再試行などの修復要求が処理されます。非同期 invoke アクティビティーの場合、管理タスクは常に作成されます。したがって、管理者は、アクティビティーが非同期応答を待っている間に、そのアクティビティーを強制的に再試行または終了させることができます。

スタンドアロン予定タスクは、非同期 invoke アクティビティーを実装できます。これらのアクティビティーにも、管理タスクが関連付けられています。インライン予定タスクは、ヒューマン・タスク・アクティビティーを実装します。管理タスクは、実行時にこれらのアクティビティーに対して作成されます。

注: プロセス管理がシステム管理者に制限される場合、インスタンス・ベースの管理は使用不可になり、管理タスク・インスタンスは作成されません。つまり、プロセス、スコープ、およびアクティビティーに対する管理アクションは BPESystemAdministrator ロールのユーザーに制限されます。この管理モードについて詳しくは、58 ページの『代替プロセス管理許可モード』を参照してください。

サブタスク

サブタスクは、担当者が自分に割り当てられている作業の一部を別の担当者に代行してもらう必要があるが、全体の結果は管理し続けたい場合をサポートしてくれます。サブタスクは、担当者が作業対象のタスクを完遂するのに役立つサポート・サービスを呼び出す場合にも使用できます。

サブタスクは、Business Process Choreographer データベースに保管されているスタンドアロン・タスク・テンプレートから作成することも、実行時にタスク・テンプレートから作成することも、実行時に新規タスク・モデルを提供することによって作成することもできます。親タスクは予定タスクにすることも、コラボレーション・タスクにすることもできますが、**supportsSubtask** 属性を `true` に設定する必要

があります。作成されるサブタスクは、コラボレーション・タスクまたは呼び出しタスクのいずれかにすることができます。これらのサブタスクも、サブタスクまたは後続タスクを持つことができます。

入力メッセージ・タイプにも出力メッセージ・タイプにも制限はありません。ただし、サブタスクのスターターは、入力メッセージを提供する必要があります。サブタスクが完了すると、親タスクの所有者はそのサブタスクの出力データを親タスクの出力メッセージにマップできます。

許可の考慮事項

サブタスクは、開始時にそのサブタスクに指定されている許可のロール以外に、親タスクから許可のロールを継承することもできます。継承されるロールは、WebSphere Integration Developer でサブタスクに定義されたロール伝搬設定によって、次のように異なります。

すべて 親タスクの読者、編集者、オリジネーター、潜在的所有者、および所有者は、サブタスクおよびそのエスカレーションの読者になります。

すべてまたは管理者
親タスクの管理者は、サブタスクとそのエスカレーションの管理者になります。

ライフ・サイクルの考慮事項

最初のサブタスクが開始すると、その親タスクは、サブタスク待機副状態になります。親タスクは、すべてのサブタスクが、完了、失敗、期限切れ、または強制終了のいずれかの終了状態に到達するまでこの副状態に留まります。親タスクのいくつかのライフ・サイクル操作 (状態変更) は、そのサブタスクに伝搬されます。そのため、親タスクが中断、再開、強制終了、または削除されるか、期限切れになると、そのサブタスクもすべて、中断、再開、強制終了、削除されるか、期限切れになります。親タスクのエスカレート済みの副状態は伝搬しません。つまり、親タスクがエスカレートされても、サブタスクはエスカレートされません。サブタスクは、それぞれエスカレーションを持っており、そのエスカレート済み副状態は、それぞれのエスカレーションの 1 つがトリガーされた場合にのみ設定されます。

サブタスクに対しては、以下の操作を実行できます。

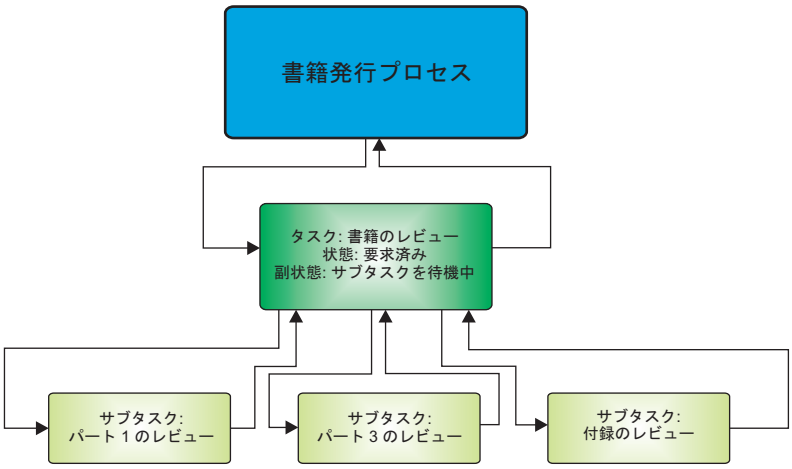
- 親タスクと競合しない操作は常にサポートされます。サブタスクまたは後続タスクの要求、取り消し要求、完了、作成、および開始などの操作がこれにあたります。
- サブタスクに有効期限を設定できます。
- 親タスクで作業が続行しているのに、サブタスクでの作業を停止しなければならない場合があるため、サブタスクを中断して、再開することができます。
- サブタスクを強制終了できます。
- 親タスクの所有者およびサブタスクのオリジネーターがサブタスクの進行を細かく制御できるようにするために、サブタスクはそれぞれにエスカレーションを持つことができます。

サブタスクに対するいくつかのライフ・サイクル操作は、親タスクのライフ・サイクル操作と競合することがあるため、許可されていません。これらの操作は主に、

サブタスクのライフ・サイクルの終了に影響する操作であり、親タスクとの間で調整が必要になります。サブタスクとして開始されるタスクに対しては、自動削除の設定値は無視されます。親タスクが削除されるか再始動されると、サブタスクは削除されます。Business Process Choreographer API を使用した個々のサブタスクの削除はサポートされていません。

例: 親タスクとコラボレーション・タスクの間の対話

以下の図は、ヒューマン・タスク・アクティビティーのサブタスクが含まれている書籍出版プロセスを示しています。



書籍出版プロセスでは、「書籍のレビュー」タスクが Linda によって要求されています。彼女は、この書籍が自分にとって大きすぎて 1 人ではレビューできないこと、およびその一部については専門知識が必要なことを理解しています。彼女は、標準的な出版プロセスからそれることにし、彼女のタスクの一部を何人かの同僚に割り当てます。彼女は 3 つの後続タスク (「パート 1 のレビュー」、「パート 3 のレビュー」、および「付録のレビュー」) を、「書籍セクションのレビュー」テンプレートから作成します。彼女自身は、書籍のパート 2 をレビューします。

彼女は、同僚が十分なコンテキスト情報を得られるように、書籍全体をサブタスクへの入力として組み込みますが、割り当てられたパートのみをレビューするように同僚に伝えるメモを対応するタスクに追加します。彼女が同僚に割り当てたタスクは、John にはパート 1 のレビュー、Cindy にはパート 3 のレビュー、Mary には付録のレビューです。次に、これら 3 つのタスクを彼女自身の「書籍のレビュー」タスクのサブタスクとして開始します。要求済み状態にあった彼女のタスクは、3 つのサブタスクがすべて完了するまでサブタスク待機副状態になります。

Cindy、John、および Mary は、自分のサブタスクを要求し、自分が受け持っている書籍のパートのレビューを開始します。一方、Linda はその書籍のパート 2 をレビューします。彼女は、自分のパートのレビューを終了すると、同僚の進捗状況を確認します。Cindy と John は各自のレビューを完了していましたが、Mary はまだ大部の付録をレビューしています。Linda のタスクはまだサブタスク待機副状態にあります。彼女はタスクを完了できませんが、Cindy と John のサブタスクの出力を基に、レビュー・コメントの統合を開始します。

一方、Mary も自分のサブタスクを完了したので、Linda の「書籍のレビュー」タスクはサブタスク待機副状態から出ます。Linda は Mary のレビュー・コメントをその書籍の残りの部分と統合し、タスクを完了します。書籍出版プロセスは続きます。「書籍のレビュー」タスクはインライン・ヒューマン・タスクであるため、ビジネス・プロセス・インスタンスが削除されると、このタスクはそのサブタスクと一緒に削除されます。

例: 親タスクと呼び出しタスクの間の対話

親タスクと呼び出しタスクの間の対話は、親タスクとコラボレーション・タスクの対話に類似しています。タスク所有者は、既存の呼び出しタスク・テンプレートからタスクを作成し、それを自分自身のタスクのサブタスクとして開始します。親タスクはサブタスク待機副状態に入り、呼び出しタスクが戻ってくるのを待ちます。サブタスクが完了すると、親タスクはサブタスク待機副状態ではなくなり、完了できるようになります。

関連概念

120 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、システム・レベルの Java EE 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・ベースの許可では、管理およびアプリケーション・セキュリティーがアプリケーション・サーバーに対して使用可能にされている必要があります。

97 ページの『ヒューマン・タスクのライフ・サイクル』
ヒューマン・タスクは、Web サービスまたはビジネス・プロセスと対話する人をサポートします。タスクの存続期間を通して発生する可能性のある対話は、そのタスクが予定タスクであるか、コラボレーション・タスクであるか、呼び出しタスクであるか、それとも管理タスクであるかによって異なります。特定の対話は特定のタスク状態でのみ可能であり、これらの対話が今度はヒューマン・タスクの状態に影響を与えます。

後続のタスク

後続タスクは、ユーザーが自分に割り当てられた作業の一部、および作業の完了に対する制御を他のユーザーに委任する場合に役立ちます。

後続タスクは、Business Process Choreographer データベースに保管されているスタンドアロン・タスク・テンプレートから作成することも、実行時にタスク・テンプレートから作成することも、実行時に新規タスク・モデルを提供することによって作成することもできます。後続タスクは、予定タスクから開始することも、**supportsFollowOnTask** 属性が `true` に設定されているコラボレーション・タスクから開始することもできます。後続タスクは、それ自体の後続タスクを伴ってタスクのチェーンを形成することができます。

後続タスクの入力メッセージのタイプは、その先行タスクとは異なる場合があります。後続タスクの入力メッセージ・タイプが先行タスクの入力メッセージ・タイプと同じである場合、先行タスクの入力メッセージの内容は自動的に後続タスクに受け渡されます。後続タスクが作成または開始されると、メッセージの内容が上書きされる場合があります。

後続タスクのチェーンの場合、各後続タスクの出力メッセージと障害メッセージのタイプは、チェーン内の最初のタスクの該当するメッセージ・タイプと同一でなければなりません。これは、チェーン内の最後の後続タスクが呼び出し側のコンポーネントまたはユーザー (オリジネーター) にメッセージを返すためです。親タスクの出力メッセージまたは障害メッセージの内容は、後続タスクの出力メッセージまたは障害メッセージに常にコピーされます。これらのメッセージは後続タスクで変更でき、変更内容は親タスクにコピーされます。

許可の考慮事項

後続タスクは、以下のように先行タスクから許可のロールを継承します。

- 先行タスクの読者、編集者、オリジネーター、潜在的所有者、および所有者は、後続タスクおよびそのエスカレーションの読者になります。
- 先行タスクの管理者は、後続タスクおよびそのエスカレーションの管理者になります。

ライフ・サイクルの考慮事項

後続タスクが開始されると、先行タスクは転送済み状態になります。後続タスクのチェーンは、単一のタスクであるかのように処理されます。そのため、チェーン内の任意のタスクで一部のライフ・サイクル操作を実行でき、正しい動作が適用されます。以下に例を示します。

- チェーン内のいずれかのタスクが中断されると、チェーン全体が中断されます。各タスクは中断副状態になります。
- 中断状態にある後続タスクのチェーンは、チェーン内の任意のタスクによって再開できます。
- チェーン内のタスクがエスカレートされると、チェーン内のすべての後続タスクがエスカレートされます。
- チェーン内のいずれかのタスクが強制終了されると、チェーン全体が強制終了されます。
- チェーン内の最初のタスクの有効期限が切れると、チェーン内の最後のタスクが期限切れの状態になります。

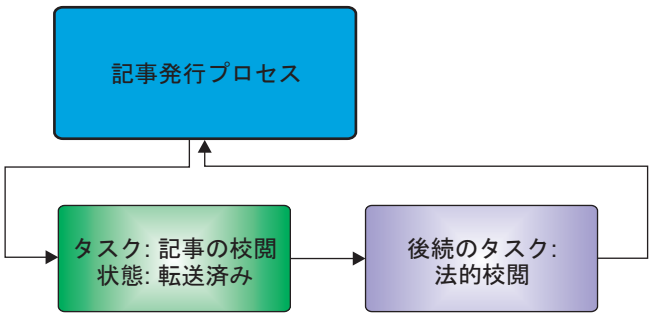
後続タスクの一部のライフ・サイクル操作が、先行タスクのライフ・サイクル操作と競合することがあります。この場合、それらは許可されません。以下に示すのは、後続タスクのライフ・サイクルの終了に影響を及ぼし、先行タスクとの調整が必要な主な操作です。以下の操作は後続タスクで実行できます。

- 親タスクと競合しないライフ・サイクル操作は常にサポートされます。このような操作には、サブタスクまたは後続タスクの要求、取り消し要求、完了、作成、開始などがあります。
- 後続タスクのチェーンは、呼び出し側のコンポーネントまたはユーザー (オリジネーター) に対して単一タスクのように振る舞うため、後続タスクでは有効期限までの期間はサポートされませんが、チェーン内の最初のタスクの有効期限タイマーが終了すると有効期限が切れます。
- トップレベル・タスクおよび後続タスクは中断および再開が可能です。このアクションにより、チェーン内のすべてのタスクが中断および再開されます。
- 後続タスクは終了できます。

- 後続タスクに独自のエスカレーションを持たせることにより、先行タスクの所有者と後続タスクのオリジネーターは、後続タスクの進行をより効果的に制御できます。
- 後続タスクは、その親タスクが削除または再開されると削除されます。Business Process Choreographer API の使用による個々の後続タスクの削除はサポートされません。

例: 後続タスク

以下の図は、ヒューマン・タスク・アクティビティーの後続タスクを使用したプロセスの公開を示しています。



記事の公開プロセス内で、「記事の校閲」タスクが John によって要求されます。彼はこのプロセスにより、記事の法的な側面を検討および承認する権限も与えられています。しかし、この記事には競合他社の製品とのコラボレーションについての記載があるため、法的な観点から細心の注意を払う必要があります。彼はこの記事の情報面について確認し、その記事を法務部門の Sarah に渡して、追加の校閲を依頼することを決定します。「法的校閲」タスクを作成し、自身の法的な関心事を示す説明を付加します。この記事をタスクへの入力データとして組み込み、それを Sarah に割り当てます。次に、自身の「記事の校閲」タスクの後続タスクとして新規タスクを開始します。彼のタスクは転送済み状態になり、それに対する作業は終了します。プロセスは、呼び出された「記事の校閲」タスクからの応答を待ちます。

Sarah は自身の「法的校閲」後続タスクを要求し、法的側面の校閲を開始します。彼女はコメントをいくつか作成し、タスクを完了します。後続タスクの出力メッセージがビジネス・プロセスに受け渡されます。記事の公開プロセスは、「記事の校閲」タスクに関連付けられている出力を続行しますが、これは「法的校閲」後続タスクからの出力です。「記事の校閲」タスクはインライン・ヒューマン・タスクであるため、ビジネス・プロセス・インスタンスが削除されるときに「法的校閲」タスクとともに削除されます。

並列所有権を持つ予定タスクおよびコラボレーション・タスク

並列所有権を持つタスクでは、複数の潜在的所有者がタスクに対して同時に作業することができます。並列所有権の一般的な例としては、一連の潜在的所有者がビジネス・プロセスの予定タスクを承認する必要がある場合が挙げられます。並列所有権は、予定タスクとコラボレーション・タスクに指定できます。

並列所有権を持つタスクが開始されると、潜在的所有者ごとにサブタスクが作成されて開始され、親タスクは実行状態になります。サブタスクは、常にコラボレーション・タスクです。親タスクの入力メッセージおよびその他のすべての関連情報は、各サブタスクにコピーされます。

サブタスクの開始後、サブタスクは作動可能状態になるか、親タスクに自動要求が指定されている場合は、要求済み状態になります。サブタスクが作成されると、親タスクはサブタスク待機副状態になります。次に、サブタスクはコラボレーション・タスクの通常のライフ・サイクルを完了します。親タスクは、サブタスクがすべて終了状態になるまでサブタスク待機副状態に留まります。親タスクの完了条件が true になると、まだ終了状態になっていないすべてのサブタスクが強制終了されます。I

親タスクには所有者がいないため、親タスクに対して claim または cancelClaim などの API 操作を使用することはできません。サブタスクが自動的に要求されるように親タスクがモデル化されている場合、サブタスクは、潜在的所有者のそれぞれに自動的に割り当てられます。

許可の考慮事項

潜在的所有者ごとにサブタスクを作成できるように、並列所有権を持つタスクに割り当てる担当者割り当て基準に対して、以下の制限が適用されます。

- Nobody または Everybody 担当者割り当て基準を使用しない。
- グループを返す担当者割り当て基準 (例えば Group など) を使用しない。

作成されたサブタスクには、以下の許可のロールがあります。

- 並列所有権を持つタスクの管理者は、サブタスクのそれぞれの管理者になります。
- 並列所有権を持つタスクを開始したユーザーは、サブタスクのそれぞれのオリジネーターになります。
- 1 人の潜在的所有者

さらに、サブタスクは、並列所有権を持つタスクから許可のロールを継承することもできます。継承されるロールは、WebSphere Integration Developer でこのタスク用に定義されたロール伝搬設定によって異なります。

すべて 親タスクの読者、編集者、オリジネーター、潜在的所有者、および所有者は、サブタスクおよびそのエスカレーションの読者になります。

すべて または 管理者
親タスクの管理者は、サブタスクとそのエスカレーションの管理者になります。

完了条件

一般に親タスクは、すべてのサブタスクが終了状態になるまでサブタスク待機副状態で待機しています。 ただし、並列所有権を使用する特定の状況では、すべてのサブタスクが終了状態になるのを待機せずに、親タスクが終了することを望む場合があります。例えば、サブタスクが文書の承認用である場合に、すべてのサブタスク所有者が文書を承認したわけではなくても、親タスクを終了させたい場合がありま

す。このタイプのシナリオを使用可能にするには、WebSphere Integration Developer で、並列所有権を持つタスクの完了条件を指定します。以下のタイプの完了条件を使用できます。

XPath ベースの完了条件
この完了条件では、完了条件関数と結果構造関数の両方を利用できます。サブタスクが作成される前と、各サブタスクが終了状態、期限切れ状態、強制終了状態、または失敗状態になった後に、条件が評価されます。並列所有権を持つタスクが終了するためには、条件が true に評価されなければなりません。

例えば、少なくとも 50% のサブタスク所有者が自分のデータを提供したときに終了するという、並列所有権を持つタスクの完了条件は、以下のコード・スニペットのようになります。

```
tel:getCountOfFinishedSubtasks() div tel:getCountOfSubtasks() > 0.5
```

カレンダー・ベースの完了期間
並列所有権を持つタスクが終了する最も遅いタイミングを指定する期間。期間の構文は、タスク定義用に指定されたカレンダーによって決まります。

完了条件のいずれかが適用されると、並列所有権を持つタスクが終了し、すべてのサブタスクの結果を集約したものが構成されます。まだ終了していないサブタスクが存在する場合、それらは自動的に強制終了されます。

結果構造

並列所有権を持つタスクの結果は、サブタスクの結果を集約することによって構成されます。 タスク定義で XPath 式を使用することにより、並列所有権を持つタスクの出力メッセージ内のフィールドに、サブタスクの結果に基づく内容を入れることができます。特定のフィールドの結果は、以下の属性からなります。これらは、タスクの出力メッセージのコンテキスト内で値を識別する XPath 式によって定義されます。

part
この属性は、使用されるロケーション・フィールドを含む出力メッセージ内の部分を示します。document/literal wrapped バインディング・スタイルを使用するメッセージ定義では、このフィールドは省略する必要があります。

location
この属性は、以下のフィールドを識別します。

- 結果の集約のソースである各サブタスクの出力メッセージ内のフィールド
- 集約の結果のターゲットである並列所有権を持つタスクの出力メッセージ内のフィールド

condition
この属性は、サブタスク・フィールドが結果構造に該当することを指定します。フィールドはロケーション属性によって識別されます。サブタスク・フィールドが該当しない場合、結果を構成するときにそのフィールドは無視されます。

aggregationFunction
この属性は、サブタスク・フィールドの値がどのように結合されて 1 つの集約された結果となるかを定義します。フィールドは、part 属性、location 属性およ

び condition 属性によって識別されます。集約された結果は、並列所有権を持つタスクの出力メッセージ内の、ロケーション属性によって識別されるフィールドに格納されます。

並列所有権を持つタスクの TEL 定義の例

以下の例では、並列所有権を持つタスクの TEL 定義を示します。

```
<tel:result>
   <tel:aggregate location="/reviewresult" function="tel:and()"/>
   <tel:aggregate location="/reviewcomments"
                  condition="/reviewresult=true()"
                  function="tel:concatWithDelimiter('|')"/>
</tel:result>
```

この例では、結果の集約が出力メッセージの reviewresult および reviewcomments フィールドに指定されています。ロケーション /reviewcomments は、サブタスクの出力メッセージ内の対応するフィールドが集約のソースとして使用されることを示します。XPath 標識の「/」は、出力メッセージ定義のルートを示します。/reviewresult=true() 条件は、サブタスクの出力メッセージ内の reviewresult フィールドの値が true に設定されている場合のみ、そのサブタスクが考慮されることを示します。集約関数は、適格な出力メッセージの値が、指定された区切り文字を使用して連結され、集約ストリングにまとめられることを指定します。

関連概念

97 ページの『予定タスクの状態遷移図』
予定タスクは、クライアント・アプリケーションによって、またはコンポーネントを呼び出すことによって自動的に作成されます。予定タスクは、作業をビジネス・プロセスの一部として実行するとき (インライン・タスク)、または公開して提供する Web サービスを実装するとき (スタンドアロン・タスク) に、ユーザーをサポートします。予定タスクのライフ・サイクル内では、特定の対話は特定のタスク状態でのみ可能であり、これらの対話が今度はタスクの状態に影響を与えます。

102 ページの『コラボレーション・タスクの状態遷移図』
コラボレーション・タスクは、担当者が他の担当者の作業を実行するときに使用できるタスクです。コラボレーション・タスクのライフ・サイクル内では、特定の対話は特定のタスク状態でのみ可能であり、これらの対話が今度はタスクの状態に影響を与えます。

並列所有権を持つヒューマン・タスクの XPath 拡張機能

並列所有権を持つヒューマン・タスクの場合は、XPath 拡張機能を使用すると、個々の応答をどのように組み合わせて結果を得るかを制御できます。

XPath 式および条件では、XPath 1.0 仕様 (http://www.w3.org/TR/xpath) に記載されている標準 XPath 機能に加えて、拡張機能と呼ばれる以下の機能を使用できます。

並列所有権を持つヒューマン・タスクの結果を集約する XPath 拡張機能

並列所有権を持つヒューマン・タスクに対して結果集約関数を使用して、個々のサブタスクの結果に基づいて出力メッセージの各フィールドを埋める方法を指定します。

string 型フィールドの集約関数

以下の関数は、並列所有権タスクのすべてのサブタスク用の出力メッセージを集約し、String 型の値を返します。出力メッセージは、関数の実行前に XPath ストリング関数に従ってストリングに変換されます。

表 10. ストリング値を返す集約関数

関数名	パラメーター	説明
concat	なし	この関数は、すべてのサブタスク用の出力メッセージの値を連結します。出力メッセージは、XPath ストリング関数に従ってストリングに変換されます。出力メッセージが存在しない場合、空ストリングが返されます。
concatWithDelimiter	区切り文字	この関数は、すべてのサブタスク用の出力メッセージの値を連結します。連結された出力内のそれぞれの値は、指定された区切り文字ストリングで区切られます。出力メッセージが存在しない場合、空ストリングが返されます。
leastFrequentOccurence	なし	この関数は、最も出現頻度の低い値を返します。出力メッセージが存在しないか、最も出現頻度の低い値を持つメッセージが複数ある (順位が同じ) 場合、空ストリングが返されます。
mostFrequentOccurence	なし	この関数は、最も出現頻度の高い値を返します。出力メッセージが存在しないか、出現頻度の同じものが存在する場合、空ストリングが返されます。
voteOnString	パーセンテージ	この関数は、最も出現頻度の高い値が、指定されたパーセンテージよりも多く出現し、なおかつ出現頻度の同じものが存在しない場合に、その値を返します。投票の結果は、以下のようにして決定されます。 1. パーセンテージと、並列所有権を持つタスクのサブタスクの総数を乗算することにより、最小出現回数が決まります。 2. ストリング値は、ステップ 1 で決められた最小出現回数を超える値のみが残るようにフィルタリングされます。 3. ステップ 1 で適格となった値のうち、最も出現頻度の高い値が返されます。 出力メッセージが存在しないか、出現頻度の同じものが存在する場合、空ストリングが返されます。

boolean 型フィールドの集約関数

以下の関数は、並列所有権タスクのすべてのサブタスク用の出力メッセージを集約し、Boolean 型の値を返します。出力メッセージは、関数の実行前に XPath ブール関数に従ってブール値に変換されます。

表 11. ブール値を返す集約関数

関数名	パラメーター	説明
and	なし	この関数は、すべての値の論理積を返します。すべての値が true の場合は結果が true となります。少なくとも 1 つの値が false であるか、出力メッセージが存在しない場合、結果は false となります。
or	なし	この関数は、すべての値の論理和を返します。いずれかの値が true の場合は結果が true となります。すべての値が false であるか、出力メッセージが存在しない場合、結果は false となります。
vote	パーセンテージ	この関数は、最も出現頻度の高いブール値が、指定されたパーセンテージよりも多く出現し、なおかつ出現頻度の同じものが存在しない場合に、そのブール値を返します。投票の結果は、以下のようにして決定されます。 1. 指定されたパーセンテージとサブタスクの数によって最小値が得られます。単一ブール値はこの最小値よりも多く出現しなければなりません。 2. ステップ 1 で適格となった値のうち、最も出現頻度の高い値を返します。 出力メッセージが存在しないか、出現頻度の同じものが存在する場合は、false が返されます。

numeric、duration、および dateTime の値の集約関数

以下の関数は、並列所有権タスクのすべてのサブタスク用の出力メッセージ・フィールドを集約し、値をオブジェクトとして返します。これらの関数は、以下の型の出力メッセージ・フィールドに適用されます。

数値 xsd:decimal、xsd:float、xsd:double、xsd:integer、byte、xsd:int、xsd:long、xsd:short

カレンダー

xsd:duration、xsd:dateTime

表 12. オブジェクトを返す集約関数

関数名	パラメーター	説明
avg	なし	この関数は、numeric、duration、または dateTime の値の平均値を返します。計算モードおよび戻りの型は、セット内で値を持つ最初のメッセージによって決まります。戻り値の型は、ノード・セット内のノードの型と同じです。出力メッセージが存在しない場合は null が返され、このフィールドの出力は設定されません。
max	なし	この関数は、numeric、duration、または dateTime の値の最大値を返します。比較モードおよび戻りの型は、セット内で値を持つ最初のメッセージによって決まります。出力メッセージが存在しない場合は null が返され、このフィールドの出力は設定されません。

表 12. オブジェクトを返す集約関数 (続き)

関数名	パラメーター	説明
min	なし	この関数は、numeric、duration、または dateTime の値の最小値を返します。比較モードおよび戻りの型は、セット内で値を持つ最初のメッセージによって決まります。出力メッセージが存在しない場合は null が返され、このフィールドの出力は設定されません。
sum	なし	この関数は、numeric または duration の値の合計を返します。dateTime 型はサポートされません。計算モードおよび戻りの型は、セット内で値を持つ最初のメッセージによって決まります。duration の値の場合、ミリ秒単位の期間の値を使用して合計を計算します。 アルゴリズムでは、すべての numeric 値を doubles に変換し、それらの値を加算します。すべての計算は doubles を使用して行われます。戻り値の型は、出力メッセージの型と同じです。出力メッセージが存在しない場合、numeric 値としては `0` が返され、duration 値としては `P0S` が返されます。

並列所有権を持つヒューマン・タスクの XPath 関数の完了条件

完了条件は、どのような場合に、親タスクに関連付けられたサブタスクのセットが完了したと見なされるかを定義します。これはブール XPath 式として指定され、終了したサブタスクからの出力データ、またはヘルパー関数から取得したその他のデータ (例えばサブタスクの数など) を参照することができます。

サブタスク・セットの出力データにアクセスするための Node-set 関数

以下の関数を使用して、終了したサブタスクの出力データにアクセスできます。結果を集約するには、返されたノード・セットを使用して、XPath 拡張機能のいずれかを呼び出します。

node-set tel:getSubtaskOutputs(string partName, string locationPath)

この関数は、ルーティング・パターンのサブタスクの出力文書から構成される単純型または複合型エレメントのノード・セットを返します。

partName

このストリング・パラメーターには、サブタスクの出力文書のパーツの名前が含まれます。

locationPath

このストリング・パラメーターには、サブタスクの出力文書内のロケーション・パスが含まれます。このパラメーターの値により、ノード・セットで返されるエレメントの数が次のように決まります。

maxOccurs=1

この関数は、完了したサブタスクごとに 1 つのノードが含まれているノード・セットを返します。

maxOccurs>1

この関数は、各リーフのノード・セットを結合したノード・セットを返します。

minOccurs=0
この関数は、完了したサブタスク数よりも少ないエレメントを格納できるノード・セットを返します (このノード・セットには空でないエレメントしか格納できないためです)。

node-set tel:getSubtaskOutputs(string locationPath)
この関数は、ルーティング・パターンのサブタスクの出力文書から構成される単純型または複合型エレメントのノード・セットを返します。この関数の動作は、**tel:getSubtaskOutputs(string partName, string locationPath)** とまったく同じです。この関数 (**tel:getSubtaskOutputs(locationPath**) は、document/literal wrapped バインディング・スタイルを使用する出力文書のみに適用されます。出力文書の他のすべてのフォーマット (multipart 書式であるが document/literal wrapped でない) では、**tel:getSubtaskOutputs (partName, locationPath)** 関数を使用する必要があります。

ヘルパー関数

以下の関数を使用して、並列所有権を持つタスクのサブタスクに関する情報にアクセスできます。これらの関数の結果は、完了条件の XPath 式のみで使用できます。

number tel:getCountOfSubtasks()
この関数は、ルーティング・パターン用に作成されたサブタスクの数を返します。

number tel:getCountOfCompletedSubtasks()
この関数は、完了しているルーティング・パターン用サブタスクの数を返します。

number tel:getCountOfFinishedSubtasks()
この関数は、終了状態にあるルーティング・パターン用サブタスクの数を返します。

エスカレーション

エスカレーションとは、ヒューマン・タスクに対するアクションが指定された時間内に実行されない場合に自動的に発生するアラートです。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 つ以上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したりすることができます。

エスカレーションは、モデル化中に、または実行時にタスクを作成するときに動的に、任意のタスクに対して定義できます。

エスカレーションは、特定のタスク状態でアクティブ化されます。タスクは通常、エスカレーションの制限時間が満了したときに、予期されたタスク状態に達していない場合にのみエスカレートされます。ただし、制限時間が満了する前に、適切な権限を持つユーザーが任意のタイミングで手動でトリガーすることもできます。エスカレーションの制限時間は、タスクに指定されたカレンダーによって解釈されます。同じアクティブ化状態にある複数のエスカレーション (またはエスカレーション・チェーン) を指定できます。エスカレートされたタスクは、エスカレート済み副状態になります。

タスクが以下のタスク状態になるときに活動状態にされるエスカレーションを定義できます。

作動可能
作動可能状態のタスクの場合、以下の状態についてエスカレーションを定義できます。

- タスクが、予期されたタスク状態である「要求済み」を使用して時間内に要求されない場合にエスカレートします。
- タスクが、予期されたタスク状態である「終了」を使用して時間内に完了されない場合にエスカレートします。

要求 要求済み状態にある予定タスクまたはコラボレーション・タスクが時間内に完了しない場合のタスクのエスカレーションを定義するには、予期されるタスクの状態が「終了」であると指定する必要があります。

実行中 呼び出されたサービスが時間内に制御を戻さない場合に、実行状態にある呼び出しタスクのエスカレーションを定義するには、予期されるタスクの状態が「終了」であると指定する必要があります。実行状態のタスクにエスカレーションが定義されている場合、定義された制限時間が満了する前に、許可ユーザーは手動でエスカレーションをトリガーできます。

サブタスク開始済み
予定タスクまたはコラボレーション・タスクの場合、最初のサブタスクの開始時に開始するエスカレーションを定義できます。この状態は、単一所有権を持つタスクに使用できますが、ほとんどの場合、並列所有権を持つタスクに使用されます。これらのタスクでは、予期される終了状態は「終了」です。

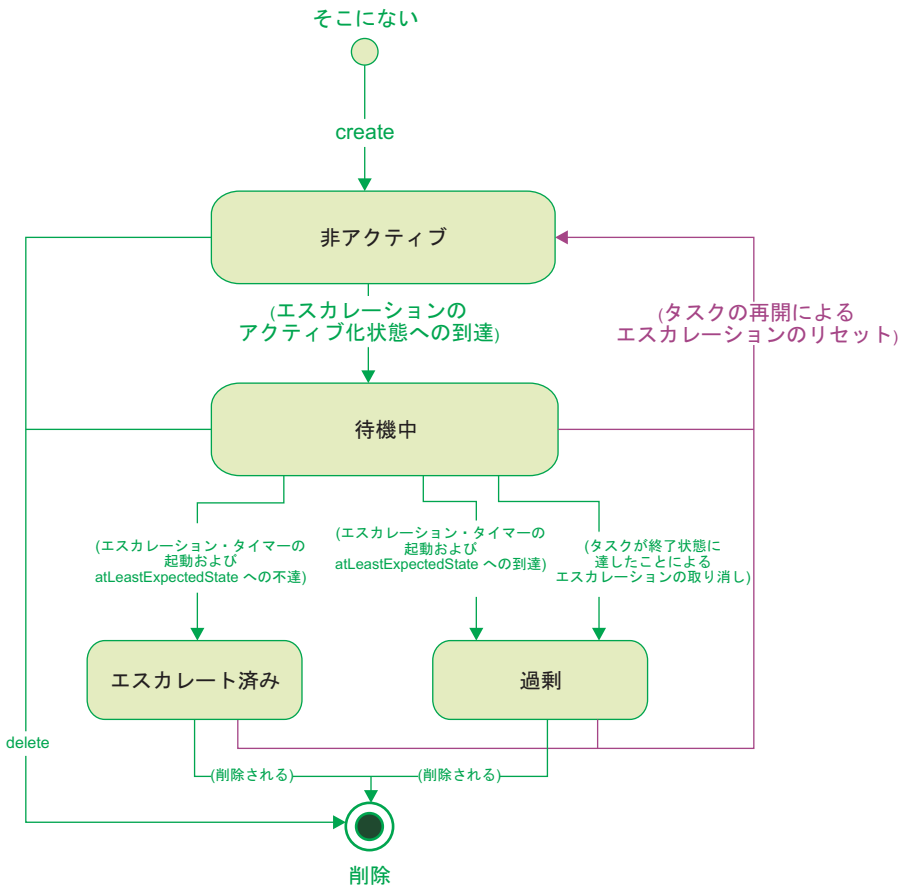
反復エスカレーションを定義できます。これらのエスカレーションは、予期された同じタスク状態をタイムアウトごとに確認し、予期されたタスク状態になるまで、定義されたエスカレーション・アクションを実行します。

エスカレーションが発生すると、エスカレーションの影響を受ける人物 (エスカレーション受信者) は作業項目を受け取ります。エスカレーションの定義に応じて、エスカレーション受信者はタスクがエスカレートされることを通知する E メールを受け取ることもあります。通知対象のユーザーのリストは担当者照会によって定義されます。この照会では、個々のユーザー ID のセットに解決する必要があります。

最初の反復のみで、またはエスカレーションの反復ごとに、タスクの優先順位が自動的に大きくなるように、エスカレーションを定義することができます。

エスカレーションのライフ・サイクル

エスカレーションのライフ・サイクル中に発生する可能性がある状態遷移を次の図に示します。



- タスクが作成されると、定義済みエスカレーションが作成され、非アクティブ状態にされます。
- タスクがアクティブ化状態になり、エスカレーション可能になると、エスカレーションは待ち状態に置かれ、タイマーが開始されます。
- 以下のいずれかが発生すると、待機中のエスカレーションがエスカレートされ、エスカレーション・アクションが実行され、関連タスクがエスカレート済み副状態になります。
  - タスクがまだ予期される状態に達しておらず、許可ユーザーが手動でエスカレーションをトリガーした。
  - タスクがまだ予期される状態に達しておらず、タイムアウトがトリガーされた。
- 以下のいずれかが発生すると、待機中のエスカレーションが不要になり、削除されます。
  - タスクが予期される状態に達し、許可ユーザーが手動でエスカレーションをトリガーした。
  - タスクが予期される状態に達し、タイムアウトがトリガーされた。
  - タスクが終了状態に達し、エスカレーションがキャンセルされた。
- エスカレーション期間と繰り返し期間を変更できます。

エスカレーション・アクションは繰り返し実行できます。

チェーン・エスカレーション

エスカレーションのチェーンとは、タスクの同じアクティブ化状態と予期される終了状態を持つ、一連のエスカレーションです。これらのエスカレーションは、どの時点でも一度に 1 つのエスカレーションのみが待機中になるように、順次に処理されます。エスカレーションのチェーンは、チェーン内の最初のエスカレーションでタスクがアクティブ化状態に到達したときにアクティブ化されます。チェーン内では、一度に 1 つのエスカレーションのみがアクティブになります。ただし、繰り返されるエスカレーションの場合は、アクティブのままになっているため、例外です。シーケンスとして定義されるエスカレーションは順次処理されます。最初のエスカレーションが発生すると、チェーン内の次のエスカレーションが活動状態にされ、その後同様に続きます。

チェーン・エスカレーションの待機期間は、前のエスカレーションのタイムアウトを基準にして計算されます。タスクがエスカレーション・アクティブ化状態に達した時点を基準にするのではありません。このため、チェーン内の最初のエスカレーションの待機期間が 2 時間で、2 回目のエスカレーションの待機時間が 3 時間の場合、最初のタイムアウトはタスクがアクティブ化状態に達した 2 時間後に発生し、2 回目のタイムアウトは 3 時間後に発生するので、タスクがアクティブ化状態になってから 5 時間後ということになります。この動作により、チェーン内の後の方のエスカレーションは、それに先行するエスカレーションより前にタイムアウトになることはありません。

エスカレーションの動的所要時間

エスカレーションによっては、エスカレーション期間を実行時に動的に設定できるものがあります。これを行うには、エスカレーションを定義する際に固定値ではなく置換式を指定します。期間変数はパーセント記号 (%) で囲む必要があります。

変数は以下のいずれかにすることができます。

- タスク変数。*%htm:input.myEscalationDurationValue%* など。
- カスタム・プロパティー。*%htm:task.property.myEscalationDurationValue%* など。
- インライン・タスクの場合はプロセス変数。
  %wf:variable.myVariable¥myPart¥myEscalationDurationValue% など。

エスカレーションが評価される際にアクセスするコンテキスト・データが使用可能であることを確認する必要があります。変数の解決に失敗する場合は、期間が正しく設定されていません。`SystemOut.log` ファイルに CWTKE0038E エラーが出力され、エスカレーションはセットアップされません。

以下の表は、エスカレーション期間がいつ評価されるのかを示しています。

期間	評価する時	タスクが以下の状態になる前にコンテキスト日付を設定する必要がある
エスカレーション	タスクがエスカレーションのアクティブ化状態に達した。チェーン・エスカレーションの場合は、各エスカレーションの期間が開始時に評価されます。	エスカレーションのタスク・アクティブ化状態。
エスカレーションの反復	エスカレーションが発生した。	エスカレート済み

関連概念

69 ページの『実行時のエスカレーションのタイミングの変更』
ビジネス・シチュエーションで、エスカレーションを定義したときに指定したエスカレーションのタイミングを変更することが必要な場合があります。エスカレーション状態によって、それらのうちどの時刻を変更できるか、およびそれらのアクションをいつ実行できるかが決まります。Human Task Manager API の update メソッドを使用して、適切なエスカレーション・プロパティーを変更できます。Business Space でタスク・リスト・ウィジェットまたはエスカレーション・リスト・ウィジェットを使用して、スケジュールされたエスカレーション時刻をオーバーライドし、エスカレーションを直ちに開始することもできます。

70 ページの『ヒューマン・タスクの置換変数』
置換変数は、実行時に解決されるエレメントの値を参照するためにヒューマン・タスクの定義で使用されます。これらの変数は、タスクに割り当てられた担当者や、タスクのカスタム・プロパティーなどの、タスクおよびプロセス関連のデータを表します。このデータは、実行時に、タスク・インスタンスのライフ・サイクルの全体または一部で使用できます。

ヒューマン・タスクのライフ・サイクル

ヒューマン・タスクは、Web サービスまたはビジネス・プロセスと対話する人をサポートします。タスクの存続期間を通して発生する可能性のある対話は、そのタスクが予定タスクであるか、コラボレーション・タスクであるか、呼び出しタスクであるか、それとも管理タスクであるかによって異なります。特定の対話は特定のタスク状態でのみ可能であり、これらの対話が今度はヒューマン・タスクの状態に影響を与えます。

関連概念

81 ページの『サブタスク』
サブタスクは、担当者が自分に割り当てられている作業の一部を別の担当者に代行してもらう必要があるが、全体の結果は管理し続けたい場合をサポートしてくれます。サブタスクは、担当者が作業対象のタスクを完遂するのに役立つサポート・サービスを呼び出す場合にも使用できます。

予定タスクの状態遷移図

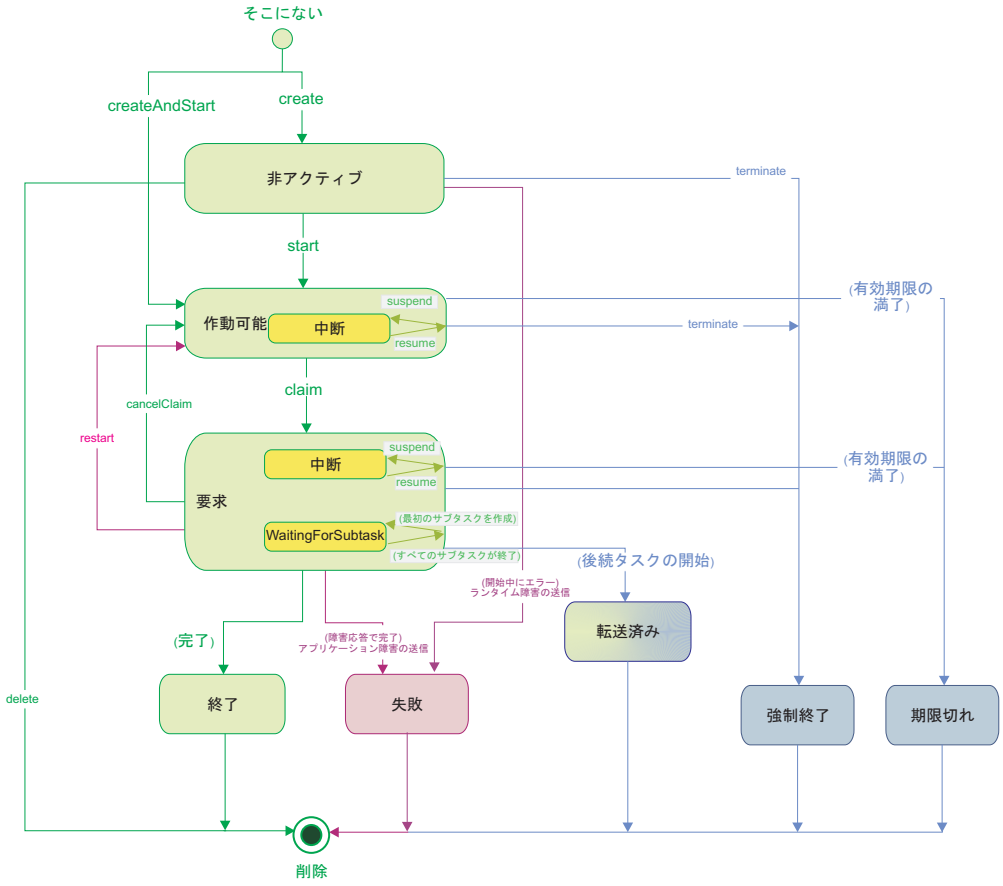
予定タスクは、クライアント・アプリケーションによって、またはコンポーネントを呼び出すことによって自動的に作成されます。予定タスクは、作業をビジネス・

プロセスの一部として実行するとき (インライン・タスク)、または公開して提供する Web サービスを実装するとき (スタンドアロン・タスク) に、ユーザーをサポートします。予定タスクのライフ・サイクル内では、特定の対話は特定のタスク状態でのみ可能であり、これらの対話が今度はタスクの状態に影響を与えます。

タスクのライフ・サイクル中に発生する状態遷移は、タスクが単一所有権を持つか並列所有権を持つかによっても異なります。

単一所有権を持つ予定タスク

次の図は、単一所有者を持つ予定タスクのライフ・サイクル中に発生する可能性がある状態遷移を示しています。スタンドアロン予定タスクでは、タスクの autonomy 属性が peer に設定されていることが前提です。



タスクが作成されると、そのタスクは非アクティブ状態になります。この状態で、タスク・プロパティーの更新またはカスタム・プロパティーの設定を行うことができます。例えば、タスクの期限までの期間、有効期限までの期間、または削除されるまでの期間を変更できます。予定タスクで作業するには、そのタスクを開始する必要があります。

タスクは、開始後に作動可能状態になります。このタスクは、潜在的所有者の 1 人がこのタスクを要求し、このタスクに関連付けられている作業を実行するのをこの状態で待ちます。この状態では、以下の例外イベントが発生する可能性があります。

- 時間内にタスクが要求されず完了もしない場合、または許可されたユーザーがエスカレーションを手動で起動した場合に、タスクがエスカレートされます。タスクはエスカレートされた副状態になり、タスクの残りのライフ・サイクルの間、この副状態に留まります。
- タスクが手動で中断される。タスクは中断副状態になります。この状態では、このタスクでのほとんどのアクションはブロックされます。タスクは手動で再開するか、または中断アクションで設定されるタイマーによって自動的に再開することができます。
- タスクの有効期限が切れる。この状態変更により、タスクは終了します。
- タスクの期限、有効期限時刻、または削除時刻をスケジュール変更するには、タスクのオリジネーター、スターター、または管理者が、期間または特定の時点に関する適切なプロパティーを編集します。
- タスクが、強制終了アクションによって、手動で終了させられる。この状態変更により、タスクは終了します。

通常のタスク・フローでは、潜在的所有者の 1 人がタスクを要求し、所有者になります。タスクは要求済み状態になり、所有者およびエディターはその状態で作業できます。タスクが要求済み状態にある場合、タスク所有者は以下のアクションを実行できます。

- タスク所有者が自分の作業にサポートを必要としている場合は、サブタスクを使用して作業の一部を代行してもらうことができます。これらのサブタスクは、コラボレーション・タスクか呼び出しタスクのいずれかです。次に親タスクがサブタスク待機副状態に入り、そのサブタスクがすべて終了状態になるまでこの状態に留まります。親タスクは、サブタスクを待っている間は中断することができますが、その親タスクを完了したり、要求を取り消したりすることはできません。親タスクが中断された場合、そのサブタスクも中断されます。
- タスク所有者が作業の完了を他の担当者に委任する場合は、例えば、後続タスクとしてコラボレーション・タスクを作成して作業を完了することが可能です。親タスクは転送済み終了状態になります。
- タスク所有者がタスクに対する全責任を代行してもらう場合は、所有者の作業項目を別の潜在的所有者または管理者に転送することができます。
- タスク所有者がタスクの所有権を放棄する場合は、タスクの要求を取り消すことができます。そのタスクはもう一度作動可能状態に置かれ、潜在的所有者の 1 人がそれを要求することができます。タスクの要求が取り消された場合、このアクションは、タスクの期限または有効期限時刻にも、エスカレーションのタイミングにも影響しません。

要求済み状態では、以下の例外イベントが発生する可能性があります。

- タスクが時間内に完了しない場合、あるいはタスクがサブタスクの完了を待機している時間が長すぎる場合に、タスクがエスカレートされます。許可されたユーザーは、エスカレーションを手動で起動することもできます。タスクは、エスカレート済み副状態になります。
- タスクが手動で中断される。タスクは中断副状態になります。この状態では、このタスクでのほとんどのアクションはブロックされます。タスクは手動で再開するか、または中断アクションで設定されるタイマーによって自動的に再開するこ

とができます。または、タイマーが満了になると、このタスクに対する要求は取り消され、そのタスクはもう一度作動可能状態になります。

- タスクの有効期限が切れる。これはタスクを終了させる状態変更です。
- タスクの期限、有効期限時刻、または削除時刻をスケジュール変更するには、タスクのオリジネーター、スターター、または管理者が、期間または特定の時点に関する適切なプロパティーを編集します。
- タスクが、強制終了アクションによって、手動で終了させられる。これはタスクを終了させる状態変更です。
- タスクが再開される。タスクは、作動可能状態に戻されます。タスクに副状態があれば、副状態は取り消されます。タスクに関連したエスカレーションは非アクティブ状態にリセットされ、通常のライフ・サイクルが始まります。タスクにサブタスクがあれば、サブタスクは終了し、削除されます。

タスクの作業が終了したら、所有者はタスクを完了します。このタスクが正常に完了すると、完了状態になります。エラーが発生した場合は、失敗状態になります。

失敗、強制終了、完了、および期限切れの各状態は、作業を実行できない終了状態です。タスク・テンプレートで自動削除が指定されている場合、タスクは即時に削除されるか、または削除タイマーが満了になると削除されます。自動削除が指定されていない場合は、タスクは明示的に削除されるまでその終了状態に留まります。親タスクが削除されると、そのサブタスクと後続タスクも削除されます。

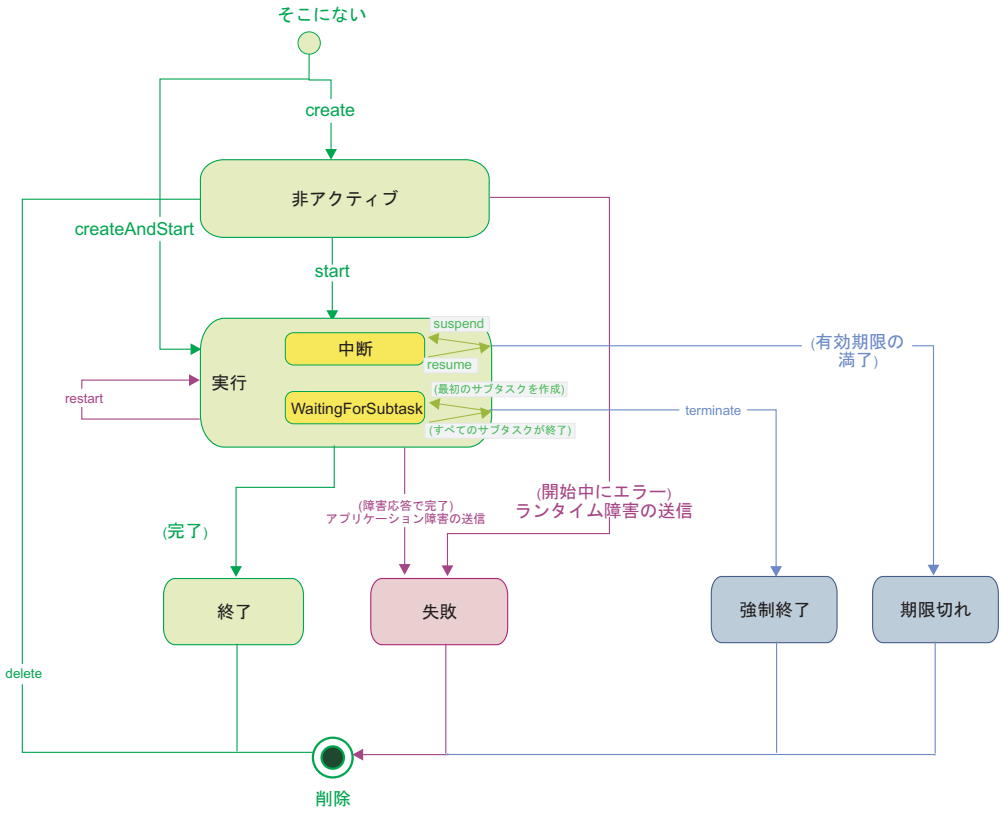
転送済み状態の場合は、後続タスクで引き続き作業が必要です。後続タスクが終了状態に達すると、直ちに親タスクの自動削除が適用されます。自動削除が行われない場合は、親タスクが明示的に削除されるまで、親タスクと後続タスクの両方がそれぞれの状態で残ります。親タスクが削除されると、後続タスクも削除されます。

インライン予定タスクに適用される追加規則もいくつかあります。インライン・タスクは、ビジネス・プロセスに不可欠の部分であり、そのライフ・サイクルはプロセスのライフ・サイクルによって制御されます。

- タスクは、ビジネス・プロセスによって暗黙で作成され、開始されます。
- ビジネス・プロセスでは、タスクはヒューマン・タスク・アクティビティーによって表されます。タスクおよびアクティビティーはどちらも、同じ状態を保持します。例えば、タスクが作動可能状態にある場合は、ヒューマン・タスク・アクティビティーも作動可能状態にあります。ヒューマン・タスク・アクティビティーは、転送済み状態またはタスクの副状態を反映しません。
- インライン・タスクにサブタスクがある場合、ヒューマン・タスク・アクティビティーはそれらのサブタスクを認識せず、親タスクが完了するまで要求済み状態で待機します。
- インライン・タスクに後続タスクがある場合、ヒューマン・タスク・アクティビティーはそれらを認識せず、後続タスクが完了するまで要求済み状態で待機します。
- インライン予定タスクには期限切れまでの期間がなく、手動で終了させることはできません。期限切れおよび強制終了はどちらも、ヒューマン・タスク・アクティビティーまたはビジネス・プロセスによって制御されます。
- タスクは、ビジネス・プロセスと一緒に削除されます。これらのタスクは、手動で削除することも、削除までの期間を作ることもできません。

並列所有権を持つ予定タスク

次の図は、並列所有権を持つ予定タスクのライフ・サイクル中に発生する可能性がある状態遷移を示しています。



親タスクを要求したり、手動で完了させることはできません。親タスクは、実行状態になると、完了条件が true になるか、有効期限に達するまでは、実行状態にとどまります。

関連概念

86 ページの『並列所有権を持つ予定タスクおよびコラボレーション・タスク』
並列所有権を持つタスクでは、複数の潜在的所有者がタスクに対して同時に作業することができます。並列所有権の一般的な例としては、一連の潜在的所有者がビジネス・プロセスの予定タスクを承認する必要がある場合が挙げられます。並列所有権は、予定タスクとコラボレーション・タスクに指定できます。

66 ページの『実行時のタスクの有効期限時刻、削除時刻、および期限の変更』
ビジネス・シチュエーションで、タスクに元から定義されている期限、有効期限時刻、または削除時刻を変更しなければならない場合があります。それらの時刻うちで実行時にスケジュール変更、キャンセル、開始できるものはどれか、およびどのような場合にそれらのアクションを実行できるかは、タスク状態によって決まります。Business Process Choreographer Explorer を使用してこれらの時刻を変更することも、Human Task Manager API の update メソッドを使用して、適切なタスク・プロパティーを変更することもできます。

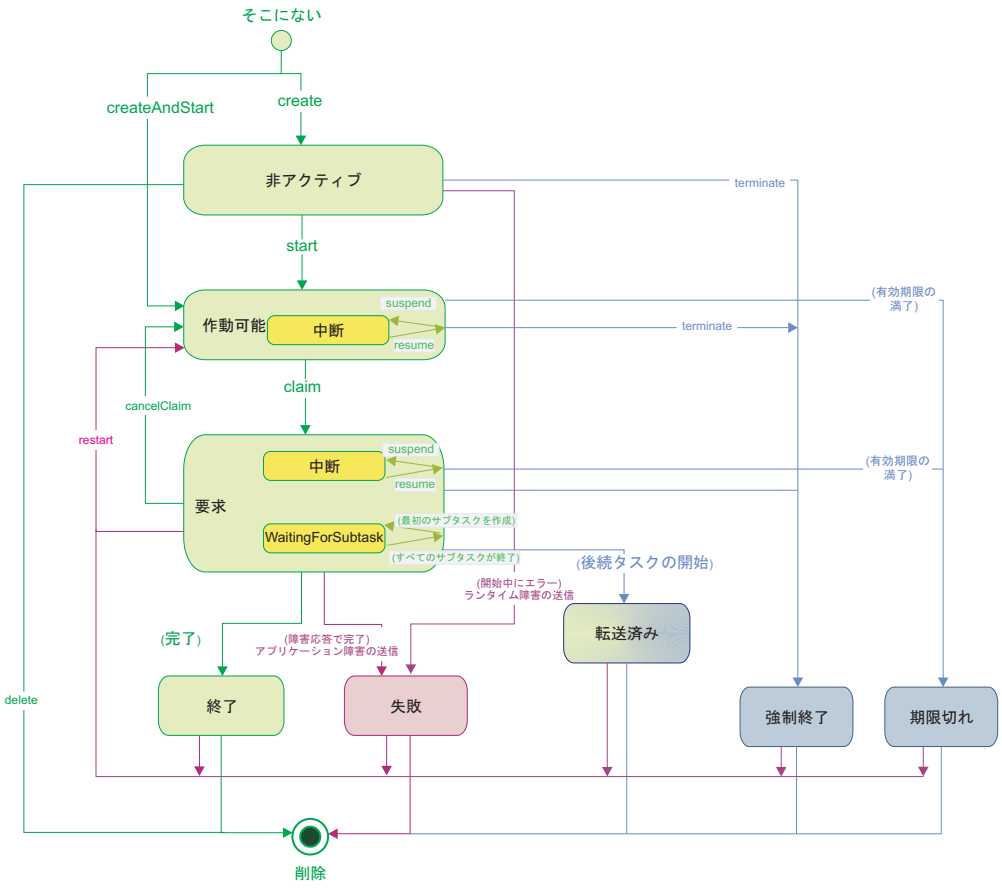
コラボレーション・タスクの状態遷移図

コラボレーション・タスクは、担当者が他の担当者の作業を実行するときに使用できるタスクです。コラボレーション・タスクのライフ・サイクル内では、特定の対話は特定のタスク状態でのみ可能であり、これらの対話が今度はタスクの状態に影響を与えます。

タスクのライフ・サイクル中に発生する状態遷移は、タスクが単一所有権を持つか並列所有権を持つかによっても異なります。

単一所有権を持つコラボレーション・タスク

次の図は、単一所有者を持つコラボレーション・タスクのライフ・サイクル中に発生する可能性がある状態遷移を示しています。



タスクが作成されると、そのタスクは非アクティブ状態になります。この状態ではタスクは要求できませんが、タスク・プロパティーの更新またはカスタム・プロパティーの設定を行うことができます。コラボレーション・タスクで作業するには、そのタスクを開始する必要があります。

タスクは、開始後に作動可能状態になります。このタスクは、潜在的所有者の 1 人がこのタスクを要求し、このタスクに関連付けられている作業を実行するのをこの状態で待ちます。この状態では、以下の例外イベントが発生する可能性があります。

- 時間内にタスクが要求されず完了もしない場合、または許可されたユーザーがエスカレーションを手動で起動した場合に、タスクがエスカレートされます。タスクはエスカレートされた副状態になり、タスクの残りのライフ・サイクルの間、この副状態に留まります。
- タスクが手動で中断される。タスクは中断副状態になります。この状態では、このタスクでのほとんどのアクションはブロックされます。タスクは手動で再開するか、または中断アクションで設定されるタイマーによって自動的に再開することができます。
- タスクの有効期限が切れる。この状態変更により、タスクは終了します。
- タスクの期限、有効期限時刻、または削除時刻をスケジュール変更するには、タスクのオリジネーター、スターター、または管理者が、期間または特定の時点に関する適切なプロパティーを編集します。

- タスクが、強制終了アクションによって、手動で終了させられる。この状態変更により、タスクは終了します。
- タスクが再開される。タスクが中断されると、中断された副状態はクリアされます。タスクがエスカレートされると、エスカレート済みの副状態はクリアされます。タスクにエスカレーションがある場合は、すべてのエスカレーションが非アクティブ状態に戻され、すべての実行中のエスカレーションが取り消されます。有効期限タイマーが設定されている場合は、そのタイマーがいったん取り消されてから再始動され、有効期限が再計算されます。タスクがサブタスクを待機している場合は、サブタスク待機副状態がクリアされ、サブタスクが削除されます。

通常のタスク・フローでは、潜在的所有者の 1 人がタスクを要求し、所有者になります。タスクは要求済み状態になり、所有者およびエディターはその状態で作業できます。タスクが要求済み状態にある場合、タスク所有者は以下のアクションを実行できます。

- タスク所有者が自分の作業でサポートを必要としている場合は、サブタスクを作成して作業の一部を他の担当者に委任できます。これらのサブタスクは、コラボレーション・タスクか呼び出しタスクのいずれかです。次に親タスクがサブタスク待機副状態に入り、そのサブタスクがすべて終了状態になるまでこの状態に留まります。親タスクは、サブタスクを待っている間は中断することができますが、その親タスクを完了したり、要求を取り消したりすることはできません。親タスクが中断された場合、そのサブタスクも中断されます。
- タスク所有者が作業の完了を他の担当者に委任する場合は、例えば、後続タスクとしてコラボレーション・タスクを作成して作業を完了することが可能です。親タスクは転送済み終了状態になります。
- タスク所有者がタスクに対する全責任を代行してもらう場合は、所有者の作業項目を別の潜在的所有者または管理者に転送することができます。
- タスク所有者がタスクの所有権を放棄する場合は、タスクの要求を取り消すことができます。そのタスクはもう一度作動可能状態に置かれ、潜在的所有者の 1 人がそれを要求することができます。タスクの要求が取り消された場合、このアクションは、タスクの期限または有効期限時刻にも、エスカレーションのタイミングにも影響しません。

要求済み状態では、以下の例外イベントが発生する可能性があります。

- タスクが時間内に完了しないため、あるいはタスクがサブタスクの完了を待機している時間が長すぎる場合に、タスクがエスカレートされる可能性があります。許可されたユーザーは、エスカレーションを手動で起動することもできます。タスクは、エスカレート済み副状態になります。
- タスクが手動で中断される。タスクは中断副状態になります。この状態では、このタスクでのほとんどのアクションはブロックされます。タスクは手動で再開するか、または中断アクションで設定されるタイマーによって自動的に再開することができます。または、タイマーが満了になると、このタスクに対する要求は取り消され、そのタスクはもう一度作動可能状態になります。
- タスクの有効期限が切れる。これはタスクを終了させる状態変更です。
- タスクが、強制終了アクションによって、手動で終了させられる。これはタスクを終了させる状態変更です。

- タスクが再開される。タスクは、作動可能状態に戻されます。タスクに副状態があれば、副状態は取り消されます。タスクに関連したエスカレーションは非アクティブ状態にリセットされ、通常のライフ・サイクルが始まります。タスクにサブタスクがあれば、サブタスクは終了し、削除されます。

所有者がこのタスクでの作業を終了すると、タスクが完了します。このタスクが正常に完了すると、完了状態になります。エラーが発生した場合は、失敗状態になります。

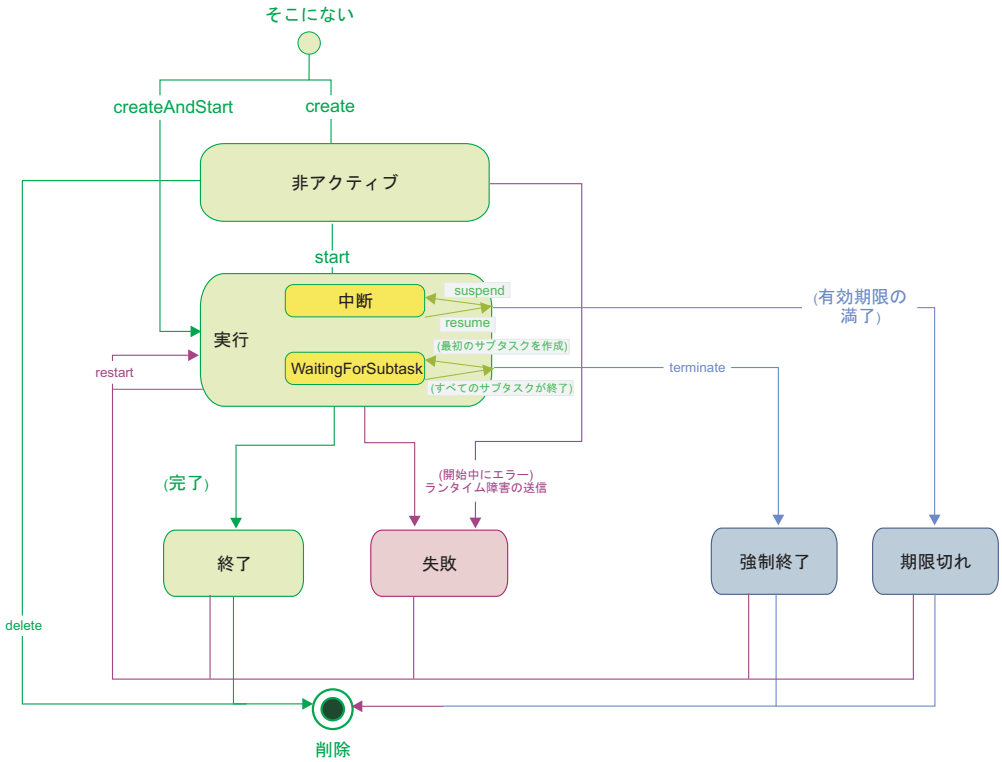
失敗、強制終了、完了、および期限切れの各状態は、作業を実行できない終了状態です。タスク・テンプレートで自動削除が指定されている場合、タスクは即時に削除されるか、または削除タイマーが満了になると削除されます。自動削除が指定されていない場合は、タスクは明示的に削除されるまでその終了状態に留まります。親タスクが削除されると、そのサブタスクと後続タスクも削除されます。

転送済み状態の場合は、後続タスクで引き続き作業が必要です。後続タスクが終了状態に達すると、直ちに親タスクの自動削除が適用されます。自動削除が行われない場合は、親タスクが明示的に削除されるまで、親タスクと後続タスクの両方がそれぞれの状態で残ります。親タスクが削除されると、後続タスクも削除されます。

いずれかの終了状態にあるタスクは、予定タスクの後続タスクでない限り、再開できます。タスクは、作動可能状態に戻されます。タスクに関連したエスカレーションは取り消され、非アクティブ状態に戻され、削除タイマーも取り消されます。

並列所有権を持つコラボレーション・タスク

次の図は、並列所有権を持つコラボレーション・タスクのライフ・サイクル中に発生する可能性がある状態遷移を示しています。



親タスクを要求したり、手動で完了させることはできません。親タスクは、実行状態になると、完了条件が true になるか、有効期限に達するまでは、実行状態にとどまります。

関連概念

86 ページの『並列所有権を持つ予定タスクおよびコラボレーション・タスク』
並列所有権を持つタスクでは、複数の潜在的所有者がタスクに対して同時に作業することができます。並列所有権の一般的な例としては、一連の潜在的所有者がビジネス・プロセスの予定タスクを承認する必要がある場合が挙げられます。並列所有権は、予定タスクとコラボレーション・タスクに指定できます。

66 ページの『実行時のタスクの有効期限時刻、削除時刻、および期限の変更』
ビジネス・シチュエーションで、タスクに元から定義されている期限、有効期限時刻、または削除時刻を変更しなければならない場合があります。それらの時刻うちで実行時にスケジュール変更、キャンセル、開始できるものはどれか、およびどのような場合にそれらのアクションを実行できるかは、タスク状態によって決まります。Business Process Choreographer Explorer を使用してこれらの時刻を変更することも、Human Task Manager API の update メソッドを使用して、適切なタスク・プロパティーを変更することもできます。

関連タスク

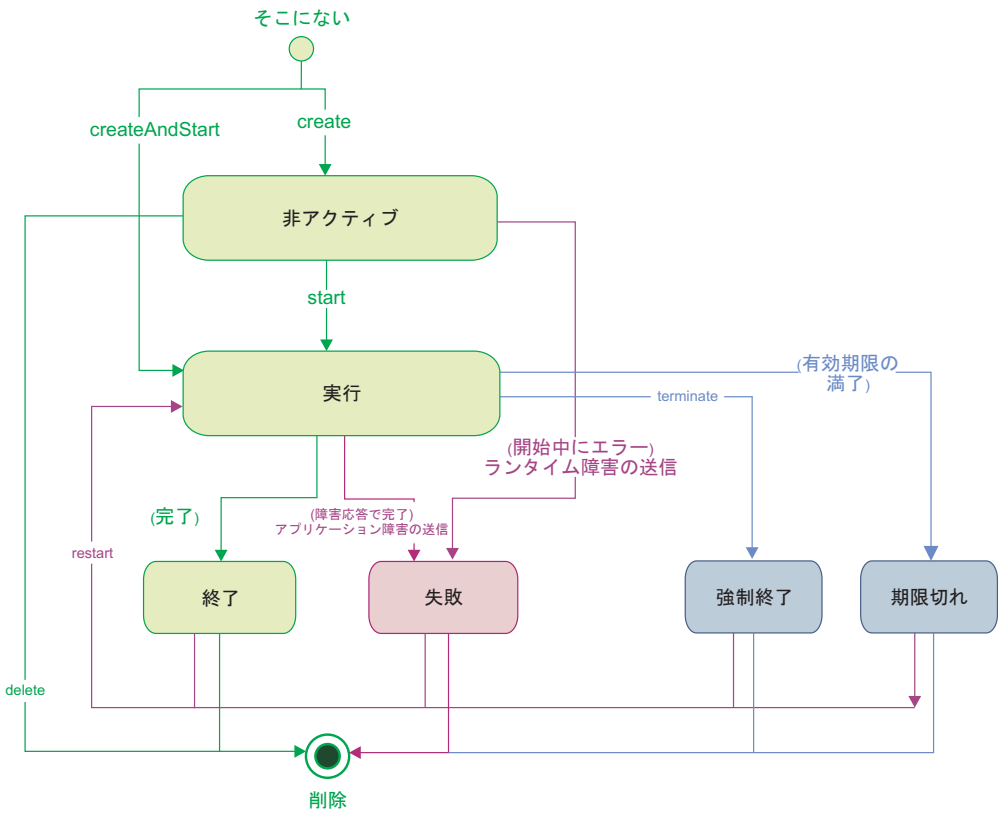
586 ページの『予定タスクまたはコラボレーション・タスクの処理』
予定タスク (API では*参加タスク* とも呼ばれる) またはコラボレーション・タスク (API ではヒューマン・タスクとも呼ばれる) は、作業項目を通じて組織内のさまざまな人に割り当てられます。プロセスがヒューマン・タスク・アクティビティーにナビゲートしたときなどに、予定タスクとそれに関連した作業項目が作成されます。

呼び出しタスクの状態遷移図

呼び出しタスクは、担当者がサービスを呼び出すときに使用できるタスクです。呼び出しタスクのライフ・サイクル内では、特定の対話は特定のタスク状態でのみ可能であり、これらの対話が今度はタスクの状態に影響を与えます。

ユーザーが呼び出しタスクを作成して開始すると、その人がタスクのオリジネーターになります。タスクは、開始されると、サービスを自動的に呼び出し、その結果を待ちます。サービスの結果が使用可能になると、呼び出しタスクはその結果を保管するので、そのタスクが存在している限り、オリジネーターはその結果を取り出すことができます。

以下の図は、呼び出しタスクのライフ・サイクルを通して発生する可能性のある状態遷移を示しています。



呼び出しタスクは、作成後に非アクティブ状態になります。この状態では、タスク・プロパティーの更新またはカスタム・プロパティーの設定を行うことができます。サービスを呼び出すには、呼び出しタスクを開始する必要があります。呼び出しタスクは、オリジネーターまたは潜在的スターターの 1 人によって開始できます。

呼び出しタスクが開始すると、そのタスクは実行状態になります。そのタスクは、この状態で、呼び出されたサービスが戻るのを待ちます。この状態では、以下の例外イベントが発生する可能性があります。

- 時間が経過してもサービスが戻らない場合は、タスクがエスカレートされる。タスクはエスカレート済み副状態になりますが、タスクの残りのライフ・サイクルの間は、この状態に留まります。
- タスクの有効期限が切れる。これはタスクを終了させる状態変更です。
- タスクが、強制終了アクションによって、手動で終了させられる。これはタスクを終了させる状態変更です。

通常のタスク・フローでは、サービスは出力メッセージまたは障害メッセージを伴って戻ります。次に呼び出しタスクは、出力メッセージが返された場合は完了状態になり、障害メッセージが返された場合は失敗状態になります。どちらの場合も、メッセージはタスクのオリジネーターおよびスターターが使用できます。

失敗、強制終了、完了、および期限切れの各状態は終了状態です。タスク・テンプレートで自動削除が指定されている場合は、削除タイマーの満了後に削除されるか、手動で削除されます。デフォルトでは、呼び出しタスクは自動的には削除されないため、呼び出されたサービスの結果を参照することができます。

いずれかの終了状態にあるタスクは再開できます。タスクは、実行状態に戻されます。このタスクに関連付けられているエスカレーションは取り消され、削除タイマーも取り消されます。

インライン呼び出しタスクに適用される追加の規則もいくつかあります。これらのタスクは、ビジネス・プロセスに不可欠の部分であり、ビジネス・プロセスはこれらのタスクのライフ・サイクルを制御できます。

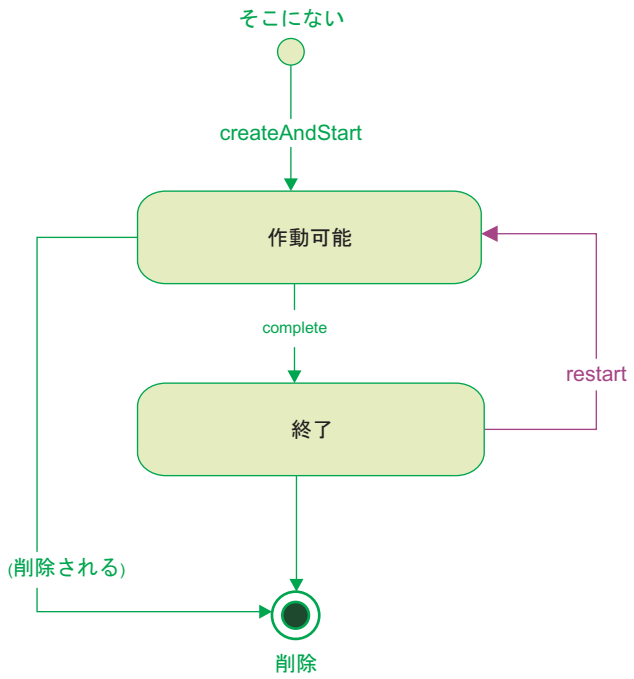
- Business Flow Manager API または SCA クライアントを使用してビジネス・プロセスが開始されると、プロセス・インスタンスを作成するアクティビティーに対応するタスクがビジネス・プロセスによって暗黙で作成され、開始されます。呼び出しタスクは、既に実行されているプロセス・インスタンスでも使用できます。この場合、それらのタスクはビジネス・プロセスによって作成され、receive アクティビティー、pick (receive choice) アクティビティー、または on-event イベント・ハンドラーに関連付けられます。
- タスクは、ビジネス・プロセスでは receive アクティビティー、pick (receive choice) アクティビティー、または on-event イベント・ハンドラーとして表されます。インラインの呼び出しタスクがアクティビティーに対して定義されている場合は、このアクティビティーに対する許可も定義されています。
- 呼び出しタスクがビジネス・プロセスによって作成され、開始される場合、呼び出しタスクのライフ・サイクルはそのビジネス・プロセスによって決まり、そのビジネス・プロセスと一緒に削除されます。呼び出しタスクが Human Task Manager API を使用して開始される場合、呼び出しタスクのライフ・サイクルは、作成された方法に関わらずプロセスに依存することはなく、その結果はプロセスが削除された後でも表示できます。
- インライン呼び出しタスクの開始方法にかかわらず、タスクの期限をスケジュール変更できます。
- インライン呼び出しタスクは、有効期限までの期間および削除されるまでの期間の両方の値を使用してモデル化できます。これらの設定は、タスクが Human Task Manager API を使用して作成される場合のみ使用可能です。これらの期間は、タスクが開始する前には更新することができ、タスクの開始後はスケジュールを変更できます。

管理タスクの状態遷移図

管理タスクは、ビジネス・プロセスとそのアクティビティーを管理するユーザーをサポートします。管理タスクのライフ・サイクル内では、特定の対話は特定のタスク状態でのみ可能であり、これらの対話が今度はタスクの状態に影響を与えます。

管理タスク・テンプレートを使用できない場合、ビジネス・プロセスでテンプレートが必要なときにはいつでも、デフォルトの管理タスクが実行時に作成されます。

以下の図は、管理タスクに対して発生する可能性のある状態遷移を示しています。



Business Flow Manager は、暗黙で管理タスクを単一のトランザクションで作成し、開始します。したがって、非アクティブ状態は外部的には不可視であり、タスクは直接作動可能状態に達します。

完了状態は終了状態です。ただし、完了状態ではそれ以上の管理アクションが禁止されているわけではありません。

管理タスクは常にインライン・タスクであり、そのライフ・サイクルはビジネス・プロセスによって制御されます。管理タスクは常にビジネス・プロセスと一緒に削除されます。

注: プロセス管理がシステム管理者に制限される場合、インスタンス・ベースの管理は使用不可になり、管理タスク・インスタンスは作成されません。つまり、プロセス、スコープ、およびアクティビティーに対する管理アクションは BPESystemAdministrator ロールのユーザーに制限されます。この管理モードについて詳しくは、 58 ページの『代替プロセス管理許可モード』を参照してください。

Business Process Choreographer でのタスク状態と Business Space でのタスク状況との関係

Business Process Choreographer には、*タスク状態* の概念が含まれています。これは、タスクのライフ・サイクル内の一段階として定義されます。例えば、タスクは非アクティブ状態または実行状態になることができます。同じ概念が、WebSphere が提供する Business Space 内のヒューマン・タスク管理ウィジェットでは*タスク状況* と呼ばれます。さらに、個々のタスク状態の名前と番号は、Business Process Choreographer と Business Space では異なります。

以下の表に、Business Process Choreographer で使用可能なタスク状態から、Business Space で示されるタスク状況へのマッピングを示します。

表 13. Business Process Choreographer でのタスク状態から Business Space でのタスク状況へのマッピング

Business Process Choreographer でのタスク状態	**タスク・タイプ**	**Business Space でのタスク状況**
非アクティブ	すべてのタスク・タイプ	アクティブでない
作動可能	コラボレーション・タスクおよび予定タスク	使用可能
実行中	並列所有権を持つ呼び出しタスク、コラボレーション・タスク、および予定タスク	進行中
終了	すべてのタスク・タイプ	正常に終了
失敗	すべてのタスク・タイプ	失敗
強制終了	すべてのタスク・タイプ	キャンセル済み
要求	単一所有権を持つコラボレーション・タスクおよび予定タスク	進行中
強制終了中	すべてのタスク・タイプ	キャンセル済み
失敗中	すべてのタスク・タイプ	失敗
期限切れ	すべてのタスク・タイプ	期限切れ
転送済み	単一所有権を持つコラボレーション・タスクおよび予定タスク	転送済み
スキップ	インライン呼び出しおよび予定タスク	スキップ
停止	すべてのタイプのインライン・タスク	停止

タスク呼び出しのシナリオ

タスクを呼び出すためのさまざまな方法については、ここで説明します。

Human Task Manager API を使用したタスク・コンポーネントの呼び出し

タスクは、Human Task Manager API を使用してインスタンス化できます。Human Task Manager API クライアントはこの API を使用してタスク・インスタンスおよび照会を作成して開始し、タスク・インスタンスを操作します。この API は、タスク呼び出しに対しては、以下の種類のタスクを作成して開始するためのメソッドを提供しています。

- スタンドアロン・タスクおよびインライン呼び出しタスク
- スタンドアロン予定タスク
- コラボレーション・タスク

この API を使用して管理タスクを呼び出すことはできません。それは、管理タスクはビジネス・プロセスのコンテキストで呼び出されるからです。

この API は、タスクに対して以下の対話スタイルをサポートしています。

- タスクおよびそれに関連付けられているサービスの同期呼び出し

  この対話スタイルでは callTask メソッドが使用されます。片方向操作の場合、タスクおよびサービス・コンポーネントの実行をトリガーした後に呼び出しが戻ります。要求/応答操作の場合は、サービスおよびタスクが完了して、呼び出しの結果が返されるまで、呼び出しは待機します。

  このスタイルの対話は、呼び出しタスクにのみ適用できます。

- タスクおよびそれに関連付けられているサービスの非同期呼び出し

  この対話スタイルでは startTask メソッドが使用されます。片方向操作および要求/応答操作の場合は、タスクおよびサービス・コンポーネントの実行をトリガーした後に呼び出しが戻ります。さらに、要求/応答操作の場合、呼び出しは、呼び出しタスクのコンテキストで出力メッセージまたは障害メッセージとして保管されている結果を非同期的に返します。呼び出し側の API クライアントは、API メソッドを使用して、結果をプログラマチックに取り出す必要があります。あるいは、応答ハンドラーを使用して、応答が使用可能になると即時に非同期応答がクライアントに戻るようにすることもできます。

  このスタイルの対話は、予定タスク、コラボレーション・タスク、および呼び出しタスクに適用できます。

Human Task Manager API は、Enterprise JavaBeans (EJB) 実装、Web サービス実装、JMS メッセージ実装、および REST 実装として提供されています。これらの API メソッドは、すべての実装に関しては類似していますが、それぞれの機能範囲は異なっています。

SCA サービス・コンポーネントとしての予定タスクの呼び出し

スタンドアロン予定タスクは、Service Component Architecture (SCA) クライアントによって非同期的に呼び出すことのできる SCA サービス・コンポーネントを表しています。SCA によって提供されるメカニズムは、SCA クライアントとスタンドアロン予定タスクの接続に使用できます。これには、以下を定義するための SCA 手段が含まれます。

- SCA クライアント参照と、予定タスクを表すコンポーネントのインターフェースとを接続するワイヤー
- 対話スタイル、トランザクションの振る舞い、対話の信頼性などの面を制御する、コンポーネント参照およびコンポーネント・インターフェースの SCA 修飾子設定

さらに、スタンドアロン予定タスクは、ビジネス・プロセスとして実装されている SCA クライアントによって呼び出すこともできます。この場合は、SCA レベルおよびプロセス・レベルの両方で接続を考慮する必要があります。SCA レベルで見ると、SCA クライアント参照が SCA サービスのインターフェースに接続しています。プロセス・レベルで見ると、invoke アクティビティーのパートナー・リンクが予定タスクに接続されています。

インライン予定タスクの呼び出し

予定タスクは、長期間にわたって実行するビジネス・プロセス内のヒューマン・タスク・アクティビティーのコンテキストで指定できます。この場合、予定タスクは、SCA レベルでの表現を取るのではなく、ビジネス・プロセスを表す SCA コンポーネントのパーツです。予定タスクはヒューマン・タスク・アクティビティーに対するサービス・プロバイダーとして動作します。プロセス・ナビゲーション中にこのアクティビティーに到達すると、予定タスクが非同期的に呼び出されます。

呼び出しタスクによる SCA サービスの呼び出し

スタンドアロン呼び出しタスクは、関連する SCA サービスへのアクセス・コンポーネントとして動作します。このサービスとの関係付けは、SCA レベルで定義されます。呼び出しタスクは、SCA サービス・コンポーネントに結合されている SCA クライアントを表します。呼び出しタスクの呼び出しには、Human Task Manager レベルと SCA レベルの両方が関係します。呼び出しタスク自体は、Human Task Manager API を介して、同期的または非同期的に呼び出されます。このタスク (SCA クライアント) は次に、そのタスクが呼び出されたのと同じ方法で、関連付けられている SCA サービス・コンポーネントを呼び出します。

タスクとサービスの間の関連のモデル化は、SCA レベルで実行されます。したがって、SCA によって提供される概念とメカニズムは、スタンドアロン呼び出しタスクと SCA サービス・コンポーネントとの接続に使用できます。これには、以下を定義するための SCA 手段が含まれます。

- SCA クライアント参照と、サービス・コンポーネントのインターフェースとを接続するワイヤー
- 対話スタイル、トランザクションの振る舞い、対話の信頼性などの面を制御する、コンポーネント参照およびコンポーネント・インターフェースの SCA 修飾子設定

さらに、スタンドアロン呼び出しタスクは、ビジネス・プロセスによって実装されている SCA コンポーネントに接続することもできます。

インライン呼び出しタスクを介したビジネス・プロセスの呼び出し

インライン呼び出しタスクは、receive アクティビティー、pick アクティビティー、またはビジネス・プロセスでのイベント・ハンドラーのコンテキストで指定できます。インライン呼び出しタスクは、SCA レベルでの表現を取るのではなく、ビジネス・プロセスを表す SCA コンポーネントのパーツです。それにもかかわらず、このインライン呼び出しタスクは、ビジネス・プロセスに対するクライアントとして動作します。このタスクは、Human Task Manager API によって呼び出されるときはいつでも、それが呼び出されたのと同じ方法でビジネス・プロセスを呼び出します。

関連概念

『スタンドアロン呼び出しタスクおよび各タスクのサービス・コンポーネントの動作に影響を与える要因』
スタンドアロン呼び出しタスクを使用すると、タスクの SCA コンポーネントに関連付けられている Service Component Architecture (SCA) サービス・コンポーネントを実行できます。呼び出しタスクとサービス・コンポーネントの関連付けは、タスク・コンポーネントの参照を、関連付けられているサービス・コンポーネントのインターフェースに結合することにより、SCA レベルでモデル化されます。呼び出しタスクおよびそれに関連付けられているサービス・コンポーネントの動作には数多くの要因が影響します。

114 ページの『シナリオ: サービスの非同期呼び出しをサポートするスタンドアロン呼び出しタスク』
このシナリオでは、タスクおよびサービスの非同期呼び出しのみを検討します。このシナリオでは、Service Component Architecture (SCA) の設定、およびこのタイプの呼び出しに対して予想されるトランザクションの動作および障害時の動作について説明します。

117 ページの『シナリオ: サービスの非同期呼び出しおよび同期呼び出しをサポートするスタンドアロン呼び出しタスク』
このシナリオでは、タスクおよびそれに関連付けられているサービスの非同期呼び出しおよび同期呼び出しを検討します。このシナリオでは、Service Component Architecture (SCA) の設定、およびこれらのタイプの呼び出しに対して予想されるトランザクションの動作および障害時の動作について説明します。

スタンドアロン呼び出しタスクおよび各タスクのサービス・コンポーネントの動作に影響を与える要因

スタンドアロン呼び出しタスクを使用すると、タスクの SCA コンポーネントに関連付けられている Service Component Architecture (SCA) サービス・コンポーネントを実行できます。呼び出しタスクとサービス・コンポーネントの関連付けは、タスク・コンポーネントの参照を、関連付けられているサービス・コンポーネントのインターフェースに結合することにより、SCA レベルでモデル化されます。呼び出しタスクおよびそれに関連付けられているサービス・コンポーネントの動作には数多くの要因が影響します。

WSDL 操作タイプ

SCA 参照と SCA インターフェースは、1 つ以上の操作を含む WSDL ポート・タイプに関連付けられています。各操作は、片方向操作または要求応答操作です。

- 片方向操作では、その完了が呼び出し側のタスクには知らされないサービス実行が暗黙指定されています。タスク・サービスの実行は、関連するサービスが正常に呼び出されて終了します。
- 要求/応答操作では、その完了が呼び出し側のタスクに知らされるサービス実行が暗黙指定されています。タスクの実行は、呼び出し側のタスクでサービスの実行の結果が使用可能になると終了します。

API 呼び出しメソッド

Human Task Manager API は、タスクの以下の対話スタイルをサポートします。

- callTask メソッドを使用した、タスクおよびそれに関連付けられているサービスの同期呼び出し
- startTask メソッドを使用した、タスクおよびそれに関連付けられているサービスの非同期呼び出し

サービス・コンポーネントの実行期間

実行期間に設定する値には、システム上の他のワークロードによる、想定オーバーヘッドを考慮する必要があります。実行期間は、Business Process Choreographer をホスティングするサーバーに設定されているトランザクションのタイムアウト値との関連で考慮する必要もあります。要求/応答インターフェースを持つサービス・コンポーネントを同期呼び出しに使用できるようにする前に、これらの値を比較しておく必要があります。そのような場合は、サービス・コンポーネントの実行時間を、サーバーに設定されているトランザクションのタイムアウト値よりも短くする必要があります。

SCA 修飾子の設定値

タスク・コンポーネント参照およびサービス・コンポーネント・インターフェースには、SCA 修飾子の特定の組み合わせのみが許可されています。

関連概念

110 ページの『タスク呼び出しのシナリオ』
タスクを呼び出すためのさまざまな方法については、ここで説明します。

シナリオ: サービスの非同期呼び出しをサポートするスタンドアロン呼び出しタスク

このシナリオでは、タスクおよびサービスの非同期呼び出しのみを検討します。このシナリオでは、Service Component Architecture (SCA) の設定、およびこのタイプの呼び出しに対して予想されるトランザクションの動作および障害時の動作について説明します。

このシナリオは、Human Task Manager API クライアント (例えば、Business Process Choreographer Explorer) が非同期呼び出しを利用する場合にのみ適用できます。このシナリオでは、タスクのモデル化時にそのタスクに関連付けられたサービスの実行期間の評価をする必要はありません。

タスク・コンポーネントの設定値

タスク・コンポーネントは、以下の設定値を取ることができます。WebSphere Integration Developer を使用してタスク・コンポーネントを定義すると、属性タイプに有効な値が自動的に生成されます。

修飾子のタイプ: 属性タイプ	値
参照属性: Multiplicity	1:1 (必須)
参照修飾子: DeliverAsyncAt	commit (必須)
実装修飾子*: Transaction	global (必須)
参照修飾子**: SuspendTransaction	適用外
実装修飾子***: ActivitySession	true (必須)
参照修飾子***: SuspendActivitySession	false (デフォルト)
参照修飾子: Reliability	assured (必須)

修飾子のタイプ: 属性タイプ	値
参照修飾子: RequestExpiration	any
参照修飾子: ResponseExpiration	any
注: • *: トランザクションの設定値を使用する場合は global を、アクティビティー・セッションの設定値を使用する場合は local を使用してください。 • **: トランザクションが global に設定されている場合は、トランザクションの設定値のみが使用されます。 • ***: トランザクションが local に設定されている場合は、アクティビティー・セッションの設定値のみが使用されます。	

サービス・コンポーネントの設定値

サービス・コンポーネントは、以下の設定値を取ることができます。WebSphere Integration Developer を使用してタスク・コンポーネントを定義すると、属性タイプに有効な値が自動的に生成されます。

修飾子のタイプ: 属性タイプ	値
インターフェース属性: PreferredInteractionStyle	無視
実装修飾子*: Transaction	local (デフォルト) global
インターフェース修飾子**: JoinTransaction	false (デフォルト) true
実装修飾子***: ActivitySession	any (デフォルト)
インターフェース修飾子***: JoinActivitySession	false (デフォルト)
注: • *: トランザクションの設定値を使用する場合は global を、アクティビティー・セッションの設定値を使用する場合は local を使用してください。 • **: トランザクションが global に設定されている場合は、トランザクションの設定値のみが使用されます。 • ***: トランザクションが local に設定されている場合は、アクティビティー・セッションの設定値のみが使用されます。	

以下のリストは、サービス **Transaction** と **JoinTransaction** 修飾子の有効な組み合わせを示しています。

- **Transaction** 修飾子が local に設定され、**JoinTransaction** が false に設定される。これらの設定値を使用すると、タスクおよびサービスの呼び出しは個別のトランザクションで実行されます。
- **Transaction** 修飾子が global に設定され、**JoinTransaction** が false に設定される。これらの設定値を使用すると、タスクおよびサービスの呼び出しは個別のトランザクションで実行されます。
- **Transaction** 修飾子が global に設定され、**JoinTransaction** が true に設定される。これらの設定値を使用すると、タスクおよびサービスの呼び出しは同じトランザクションで実行されます。

トランザクションおよび障害時の振る舞い

この非同期呼び出しシナリオでは、API 呼び出しにのみ startTask メソッドが使用されます。タスクおよびサービスの呼び出しは、それぞれ別のトランザクションで行われます。サービスの実装で処理されていない実行時例外が発生すると、以下が適用されます。このシナリオには、以下のトランザクション動作と例外処理があります。

操作のタイプ	SCA 実行時例外が発生する時期	タスクおよびサービスの動作
片方向操作	サービスの呼び出し中、ただしサービスの実行の開始前	タスクは SCA 実行時例外を受け取ります。Human Task Manager API メソッドは CoreOTaskServiceRuntimeExceptionReceivedException 例外をスローします。タスク・トランザクションはロールバックされ、タスクは非アクティブ状態に留まります。
片方向操作	サービスの実行中	呼び出しタスクには通知されません。タスクは完了状態に移行します。失敗イベントが生成されます。このイベントは、Failed event manager を使用して処理できます。
要求応答操作	サービスの呼び出し中、ただしサービスの実行の開始前	タスクは SCA 実行時例外を受け取ります。Human Task Manager API メソッドは CoreOTaskServiceRuntimeExceptionReceivedException 例外をスローします。タスク・トランザクションはロールバックされ、タスクは非アクティブ状態に留まります。
要求応答操作	サービスの実行中	タスクは SCA 実行時例外を通知され、その例外をデータベース内のタスク・コンテキストに保管します。応答ハンドラーを使用できる場合は、それを使用してクライアントに通知します。タスクは失敗状態になります。

操作定義には、実行中にサービス・コンポーネントがスローできる 1 つ以上の障害メッセージを含めることができます。

タスク・コンポーネントに、次のように障害メッセージが通知されます。

- 障害メッセージがタスクのコンテキストでデータベースに保管されます。
- タスクは失敗状態になります。
- タスクが同期的に呼び出され、応答ハンドラーが指定されていた場合は、応答ハンドラーが呼び出されて、クライアントに障害発生が返されます。
- タスクが非同期的に呼び出された場合は、障害メッセージが FaultReplyException 例外としてクライアントに返されます。

障害処理はトランザクションの動作には影響しません。トランザクションはロールバックされません。

関連概念

110 ページの『タスク呼び出しのシナリオ』
タスクを呼び出すためのさまざまな方法については、ここで説明します。

シナリオ: サービスの非同期呼び出しおよび同期呼び出しをサポートするスタンドアロン呼び出しタスク

このシナリオでは、タスクおよびそれに関連付けられているサービスの非同期呼び出しおよび同期呼び出しを検討します。このシナリオでは、Service Component Architecture (SCA) の設定、およびこれらのタイプの呼び出しに対して予想されるトランザクションの動作および障害時の動作について説明します。

このシナリオでは、Human Task Manager クライアントは非同期呼び出しと同期呼び出しの両方を使用します。このシナリオでは、サービス実行時間がサーバー・トランザクションのタイムアウトの期待値よりも短いかどうかが評価済みであるものとしています。一般に実行期間は、サーバー・トランザクションのタイムアウト値よりも十分に短くなっている必要があります。

タスク・コンポーネントの設定値

タスク・コンポーネントは、以下の設定値を取ることができます。WebSphere Integration Developer を使用してタスク・コンポーネントを定義すると、属性タイプに有効な値が自動的に生成されます。

修飾子のタイプ: 属性タイプ	値
参照属性: Multiplicity	`1:1` (必須)
参照修飾子: DeliverAsyncAt	`commit` (必須)
実装修飾子*: Transaction	`global` (必須)
参照修飾子**: SuspendTransaction	適用外
実装修飾子***: ActivitySession	`true` (必須)
参照修飾子***: SuspendActivitySession	`false` (デフォルト)
参照修飾子: Reliability	`assured` (必須)
参照修飾子: RequestExpiration	`any`
参照修飾子: ResponseExpiration	`any`

注:

- *: トランザクションの設定値を使用する場合は `global` を、アクティビティー・セッションの設定値を使用する場合は `local` を使用してください。
- **: トランザクションが `global` に設定されている場合は、トランザクションの設定値のみが使用されます。
- ***: トランザクションが `local` に設定されている場合は、アクティビティー・セッションの設定値のみが使用されます。

サービス・コンポーネントの設定値

サービス・コンポーネントは、以下の設定値を取ることができます。WebSphere Integration Developer を使用してタスク・コンポーネントを定義すると、属性タイプに有効な値が自動的に生成されます。

修飾子のタイプ: 属性タイプ	値
インターフェース属性: PreferredInteractionStyle	無視
実装修飾子*: Transaction	`local` (デフォルト) `global`
インターフェース修飾子**: JoinTransaction	`false` (デフォルト) `true`
実装修飾子***: ActivitySession	`any` (デフォルト)
インターフェース修飾子***: JoinActivitySession	`false` (デフォルト)

注:

- *: トランザクションの設定値を使用する場合は `global` を、アクティビティー・セッションの設定値を使用する場合は `local` を使用してください。
- **: トランザクションが `global` に設定されている場合は、トランザクションの設定値のみが使用されます。
- ***: トランザクションが `local` に設定されている場合は、アクティビティー・セッションの設定値のみが使用されます。

以下のリストは、サービス **Transaction** と **JoinTransaction** 修飾子の有効な組み合わせを示しています。

- **Transaction** 修飾子が `local` に設定され、**JoinTransaction** が `false` に設定される。これらの設定値を使用すると、タスクおよびサービスの呼び出しは個別のトランザクションで実行されます。
- **Transaction** 修飾子が `global` に設定され、**JoinTransaction** が `false` に設定される。これらの設定値を使用すると、タスクおよびサービスの呼び出しは個別のトランザクションで実行されます。
- **Transaction** 修飾子が `global` に設定され、**JoinTransaction** が `true` に設定される。これらの設定値を使用すると、タスクおよびサービスの呼び出しは同じトランザクションで実行されます。

トランザクションおよび障害時の振る舞い

このシナリオには、以下のトランザクション動作と例外処理があります。

API 呼び出しスタイル	操作のタイプ	SCA 実行時例外が発生する時期	タスクおよびサービスの動作
callTask	片方向操作	サービスの呼び出し中、ただしサービスの実行の開始前	タスクは SCA 実行時例外を受け取ります。Human Task Manager API メソッドは CoreOTaskServiceRuntimeExceptionReceivedException 例外をスローします。タスク・トランザクションはロールバックされ、タスクは非アクティブ状態に留まります。
callTask	片方向操作	サービスの実行中	タスクは SCA 実行時例外を受け取ります。Human Task Manager API メソッドは CoreOTaskServiceRuntimeExceptionReceivedException 例外をスローします。タスク・トランザクションはロールバックされ、タスクは非アクティブ状態に留まります。

API 呼び出しスタイル	操作のタイプ	SCA 実行時例外が発生する時期	タスクおよびサービスの動作
callTask	要求応答操作	サービスの呼び出し中、ただしサービスの実行の開始前	タスクは SCA 実行時例外を受け取ります。Human Task Manager API メソッドは CoreOTaskServiceRuntimeExceptionReceivedException 例外をスローします。タスク・トランザクションはロールバックされ、タスクは非アクティブ状態に留まります。
callTask	要求応答操作	サービスの実行中	タスクは SCA 実行時例外を受け取ります。Human Task Manager API メソッドは CoreOTaskServiceRuntimeExceptionReceivedException 例外をスローします。タスク・トランザクションはロールバックされ、タスクは非アクティブ状態に留まります。
startTask	片方向操作	サービスの呼び出し中、ただしサービスの実行の開始前	タスクは SCA 実行時例外を受け取ります。Human Task Manager API メソッドは CoreOTaskServiceRuntimeExceptionReceivedException 例外をスローします。タスク・トランザクションはロールバックされ、タスクは非アクティブ状態に留まります。
startTask	片方向操作	サービスの実行中	呼び出しタスクには通知されません。タスクは完了状態に移行します。失敗イベントが生成されます。このイベントは、Failed event manager を使用して処理できます。
startTask	要求応答操作	サービスの呼び出し中、ただしサービスの実行の開始前	タスクは SCA 実行時例外を受け取ります。Human Task Manager API メソッドは CoreOTaskServiceRuntimeExceptionReceivedException 例外をスローします。タスク・トランザクションはロールバックされ、タスクは非アクティブ状態に留まります。
startTask	要求応答操作	サービスの実行中	タスクは SCA 実行時例外を通知され、その例外をデータベース内のタスク・コンテキストに保管します。応答ハンドラーを使用できる場合は、それを使用してクライアントに通知します。タスクは失敗状態に移行します。

操作定義には、実行中にサービス・コンポーネントがスローできる 1 つ以上の障害メッセージを含めることができます。

タスク・コンポーネントに、次のように障害メッセージが通知されます。

- 障害メッセージがタスクのコンテキストでデータベースに保管されます。
- タスクは失敗状態になります。
- タスクが非同期的に呼び出され、応答ハンドラーが指定されていた場合は、応答ハンドラーが呼び出されて、クライアントに障害発生が返されます。
- タスクが同期的に呼び出された場合は、障害メッセージが FaultReplyException 例外としてクライアントに返されます。

障害処理はトランザクションの動作には影響しません。トランザクションはロールバックされません。

関連概念

110 ページの『タスク呼び出しのシナリオ』
タスクを呼び出すためのさまざまな方法については、ここで説明します。

ヒューマン・タスクのための許可および担当者割り当て

許可は、特定の担当者が、選択されているアクションをタスク・テンプレート、タスク・インスタンス、およびエスカレーションに対して実行できるようにするメカニズムです。許可のロールを使用して、特定のロールで使用可能な一連のアクションを定義します。Java EE メカニズムを使用すると担当者をシステム・レベルのロールに割り当てることができます。または、担当者割り当て基準を使用すると、担当者をタスク・インスタンス・ロールに割り当てることができます。

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、システム・レベルの Java EE 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・ベースの許可では、管理およびアプリケーション・セキュリティーがアプリケーション・サーバーに対して使用可能にされている必要があります。

関連概念

81 ページの『サブタスク』
サブタスクは、担当者が自分に割り当てられている作業の一部を別の担当者に代行してもらう必要があるが、全体の結果は管理し続けたい場合をサポートしてくれます。サブタスクは、担当者が作業対象のタスクを完遂するのに役立つサポート・サービスを呼び出す場合にも使用できます。

ヒューマン・タスクに対する Java EE の許可のロール

システム・レベルの Java EE ロールは、Human Task Manager が構成されたときにセットアップされます。これらのロールで暗黙指定される権限レベルは、すべてのタスクおよびエスカレーションに有効です。

サポートされている Java 2 Platform, Enterprise Edition (Java EE) ロールを次に示します。

- 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。
- 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 Platform, Enterprise Edition (Java EE) 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 インフォメーション・センターの『ロール・ベースの許可の System Authorization Facility』を参照してください。

ヒューマン・タスクに対するインスタンス・ベースの許可のロール

タスク・インスタンスまたはエスカレーション・インスタンスは個人には直接割り当てられておらず、その代わり、個人が割り当てられる事前定義のロールに関連付けられます。インスタンス・ベースのロールに割り当てられたユーザーは、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、個人の割り当てによるか、またはタスク・アクションの結果として決定されます。

個人は、担当者ディレクトリーに保管されたユーザーまたはユーザー・グループの情報に基づくユーザーの割り当てにより、実行時に次のロールに割り当てられます。つまり、潜在的作成者、可能なスターター、可能な所有者、読者、編集者、管理者、およびエスカレーション受信者です。次のロールは 1 人のユーザーのみに関連付けられ、タスク・アクションの結果として割り当てられます。つまり、オリジネーター、スターター、所有者です。

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・テンプレートに対して潜在的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見なされます。
オリジネーター	このロールを持つユーザーは、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。
潜在的スターター	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。スタンドアロン・タスクに潜在的スターターが指定されていない場合は、オリジネーターが潜在的スターターになります。潜在的スターターがないインライン呼び出しタスクの場合、デフォルトは全員となります。
スターター	このロールを持つユーザ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。

ロール	許可されたアクション
潜在的所有者	このロールのメンバーはタスクを要求できます。潜在的所有者が指定されている場合、すべてのユーザーがこのロールのメンバーと見なされます。このロールの担当者の解決が失敗した場合、潜在的所有者として管理者が割り当てられます。
所有者	このロールを持つユーザーがタスクで作業して完了します。
読者	このロールのメンバーは、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。

注: プロセス管理がシステム管理者に制限される場合、インスタンス・ベースの管理は使用不可になります。つまり、プロセス、スコープ、およびアクティビティーに対する管理アクションは BPESystemAdministrator ロールのユーザーに制限されます。さらに、プロセス・インスタンスまたはその一部の読み取り、表示、およびモニターを実行できるのは、BPESystemAdministrator ロールまたは BPESystemMonitor ロールのユーザーのみになります。この管理モードについて詳しくは、58 ページの『代替プロセス管理許可モード』を参照してください。

タスクの種類とインスタンス・ベースの許可のロール

インスタンス・ベースの許可のロールは、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関連付けられています。タスクの種類により、特定の許可のロールがタスクに使用できるかどうかが決まります。

ロール	予定タスク	呼び出しタスク	コラボレーション・タスク	管理タスク	コメント
潜在的インスタンス作成者	X	X	X		タスク・インスタンスを作成することを許可されるユーザー
オリジネーター	X	X	X		タスクを作成したユーザー
潜在的所有者	X		X		タスクを要求して作業することができるユーザー
所有者	X		X		タスクを要求したユーザー
可能なスターター		X			タスクの開始を許可されるユーザー
スターター		X			タスクを開始したユーザー
管理者	X	X	X	X[1]	タスクの管理を許可されるユーザー
編集者	X		X		タスク・データの編集を許可されるユーザー
読者	X	X	X	X[2]	タスク・データの表示を許可されるユーザー
エスカレーション受信者	X[3]	X[3]	X[3]	X[3]	エスカレーションを受け取るユーザー

ロール	予定タスク	呼び出しタスク	コラボレーション・タスク	管理タスク	コメント
注: 1. このロールには、管理対象のプロセス、スコープ、またはアクティビティーで管理操作を実行する権限もあります。 2. このロールには、管理対象のプロセス、スコープ、またはアクティビティーで読み取り操作を実行する権限もあります。 3. このロールには、対応するタスクで読み取り操作を実行する権限があります。					

注: プロセス管理がシステム管理者に制限される場合、インスタンス・ベースの管理は使用不可になり、管理タスク・インスタンスは作成されません。つまり、プロセス、スコープ、およびアクティビティーに対する管理アクションは BPESystemAdministrator ロールのユーザーに制限されます。この管理モードについて詳しくは、 58 ページの『代替プロセス管理許可モード』を参照してください。

タスクの許可および作業項目

すべてのタスク・ロールにより、ユーザーは関連タスクに対して一連の的確なアクションを実行できます。ユーザーの許可は作業項目を使用して管理されます。作業項目は、割り当てられたユーザーと、タスク・ロールによって示されるタスク・アクションの関係を表します。

作業項目には以下の局面があります。

- ユーザーまたはユーザー・グループの ID
- アクションの実行対象にすることができるオブジェクト (ヒューマン・タスクやビジネス・プロセスなど) の ID。
- ユーザーが関連付けられるタスク・ロール

作業項目に関連したユーザーは、以下のいずれかの方法で指定できます。

- 1 つだけのユーザー ID として指定。これはユーザー作業項目になります。
- 1 つだけのユーザー・グループ ID として指定。これはグループ作業項目になります。
- **Everybody** 担当者割り当て基準を使用するすべてのユーザーについて指定。これは Everybody 作業項目になります。

Business Process Choreographer の許可の仕組みにより、以下のいずれかの条件が保持されている場合、ユーザーは作業項目に関連付けられたアクションを実行できます。

- ユーザーはユーザー作業項目に指定されたユーザー ID に適合するユーザー ID でログインします。
- ログオン・ユーザーは、グループ作業項目に指定されたグループ ID に対応するグループのメンバーです。
- 作業項目とは、すべてのユーザーに割り当てられる作業項目です。

Human Task Manager API は、ヒューマン・タスク、エスカレーション、およびその他のオブジェクトを照会する方法を提供します。照会が実行されると、ユーザー

が作業項目を持っているデータのみを戻すことにより、ユーザーが照会済みデータを表示するための許可が確保されます。また、API を使用して、インスタンス・ベースの許可を管理することもできます。これは、作業項目を作成および削除し、担当者間で作業項目を転送することによって行われます。これらの API メソッドについて詳しくは、com.ibm.task.api パッケージ内の HumanTaskManager インターフェースに関する Javadoc を参照してください。

担当者割り当て基準

担当者割り当て基準は、インスタンス・ベースの許可のロールに割り当てることのできる担当者のセットを識別するために、タスク・モデルで使用される構成体です。担当者解決は、実行時に、担当者割り当て基準を使用してユーザー ID および他のユーザー情報を、例えば電子メールを作成するために、担当者ディレクトリーから取り出します。担当者割り当て基準は、実行時に、タスク・モデルがプログラマチックに作成される場合にも使用されます。

担当者割り当て基準定義を WebSphere Integration Developer で使用すると、タスク・ロールに対する担当者割り当てをモデル化できます。定義は照会名および照会パラメーターのセットで構成されます。タスクがデプロイされると、担当者割り当て基準が、担当者ディレクトリー (例えば、仮想メンバー・マネージャー) に固有の照会に変換されます。タスクを実行すると、潜在的所有者など、あるロールに割り当てられている担当者のセットがこれらの照会で取り出されます。

以下の例では、タスク・ロールに対する担当者割り当て基準定義の実装に関する手順を示します。

1. WebSphere Integration Developer で、モデラーが新規タスクを、例えば仮想メンバー・マネージャーの担当者ディレクトリー構成 `bpe/staff/samplevmmconfiguration` に関連付けます。

   このステップでは、担当者割り当てに使用できる担当者割り当て基準が決まります。

2. WebSphere Integration Developer で、モデラーがタスク・ロールを担当者割り当て基準に関連付けます。

   例えば、潜在的所有者ロールは、以下のパラメーターを含む担当者割り当て基準「**グループ・メンバー**」に関連付けられます。

   - 値 `cn=group1，dc=mycomp，dc=com` に設定される **GroupName**
   - 値 `true` に設定される **IncludeSubgroups**

3. タスクがデプロイされると、担当者割り当てサービスによって、使用される担当者ディレクトリー・プロバイダーが設定されます。このサービスは、担当者割り当て基準を担当者ディレクトリー・プロバイダーの照会に変換し、内部に保管します。

使用される担当者ディレクトリーに応じて、事前定義されている担当者割り当て基準のさまざまなサブセットがタスクのモデル化時に使用可能になります。

- LDAP および仮想メンバー・マネージャー担当者ディレクトリー・プロバイダーは、事前定義されているすべての定義をサポートします。

- ユーザー・レジストリーの担当者ディレクトリー・プロバイダーは、ユーザー名およびグループ名に基づく定義のみをサポートします。マネージャーまたは電子メール属性に基づく定義はサポートされません。
- システムの担当者ディレクトリー・プロバイダーは、テストのみを目的としたものです。担当者ディレクトリーへのアクセスが不要になるよう、サポートは、ハードコーディングされたユーザー ID のセットの指定に限定されています。

担当者割り当て基準定義における置換変数

置換変数を、いくつかの担当者割り当て基準定義内のパラメーター値として使用することができます。担当者解決は、コンテキストによって提供される情報に基づいて、実行時に割り当て基準を解決できます。

例えば、担当者割り当て基準定義に `%htm:input.¥name%` という置換変数がパラメーターとして含まれている場合があります。この変数は、タスクの開始時にそのタスクが受け取ったタスク入力メッセージ値の「name」エレメントを示しています。担当者解決では、変数がタスク入力メッセージ値で動的に置き換えられます。

関連概念

70 ページの『ヒューマン・タスクの置換変数』
置換変数は、実行時に解決されるエレメントの値を参照するためにヒューマン・タスクの定義で使用されます。これらの変数は、タスクに割り当てられた担当者や、タスクのカスタム・プロパティーなどの、タスクおよびプロセス関連のデータを表します。このデータは、実行時に、タスク・インスタンスのライフ・サイクルの全体または一部で使用できます。

担当者解決

担当者解決は、担当者割り当て基準と呼ばれるパラメーター化された照会式のセットに基づいて、担当者ディレクトリーからユーザー情報を取り出します。

Business Process Choreographer で使用される担当者ディレクトリー

担当者ディレクトリーには、担当者をヒューマン・タスクに割り当てる照会を解決するために使用するユーザー情報が保管されています。

担当者解決をサポートするには、担当者ディレクトリーで以下の属性がサポートされる必要があります。

- ユーザーのユーザー・プロファイルおよびログイン ID を識別する名前。
- ユーザーの管理者に関連する情報を活用するために、担当者ディレクトリーは対応する属性を提供する必要があります (デフォルトでは管理者属性)。
- エスカレーションを知らせるための E メール通知機能を活用するために、担当者ディレクトリーはユーザーの E メール・アドレスを提供する必要があります。

Business Process Choreographer では、担当者解決のために以下の担当者ディレクトリーがサポートされます。担当者割り当てのために Business Process Choreographer に用意されている機能のフルセットを活用する場合は、担当者ディレクトリーとして Virtual Member Manager を使用してください。

- フェデレーテッド・リポジトリー (Virtual Member Manager ともいわれる)

これは、WebSphere Application Server でサポートされるデフォルトの担当者ディレクトリーです。Lightweight Directory Access Protocol (LDAP) ディレクトリー、データベースとファイル・ベース・リポジトリー、およびカスタム・リポジトリーを含む、さまざまなディレクトリー・タイプへのアクセスを提供します。また、リポジトリーの統合もサポートします。

個人情報とグループ情報の両方を取得できます。サポートされる個人スキーマ (PersonAccount エンティティー・タイプ) には、ユーザーの名前、ログイン ID、管理者 ID、および E メール・アドレスのプロパティーが含まれます。担当者解決で使用できるようにするには、フェデレーテッド・リポジトリーを WebSphere Application Server のアクティブなセキュリティー・レルム定義として構成する必要があります。

- LDAP ディレクトリー

  Business Process Choreographer は、担当者解決のために、WebSphere Application Server セキュリティーを使用することなく直接 LDAP ディレクトリーにアクセスできます。担当者解決 (Business Process Choreographer で実装) とユーザー認証 (WebSphere Application Server セキュリティーで実装) との整合性を持たせるには、WebSphere Application Server セキュリティーを構成して、Business Process Choreographer で担当者解決用に指定されているのと同じ LDAP ディレクトリー・サーバーにアクセスするようにする必要があります。

  使用する LDAP 個人スキーマに応じて、個人に関連する情報には、ユーザー名、ID、管理者名、および E メール・アドレスが含まれます。担当者解決に使用できるようにするには、Business Process Choreographer 担当者ディレクトリー・プロバイダー構成が必要です。

- WebSphere Application Server のユーザー・レジストリー

  ユーザー・レジストリーは、ユーザー情報を取得するための、アプリケーション・サーバーのサブシステムです。Business Process Choreographer では、このユーザー・レジストリーを担当者ディレクトリーとして使用できます。Business Process Choreographer では、独自のユーザー・レジストリーの担当者ディレクトリー・プロバイダーを使用して、WebSphere Application Server ユーザー・レジストリーにアクセスします。

担当者ディレクトリー・プロバイダーと担当者ディレクトリー構成

Business Process Choreographer では、担当者ディレクトリーにアクセスするためのアダプターとして担当者ディレクトリー・プロバイダーを使用します。ユーザーは、Virtual Member Manager、LDAP、ユーザー・レジストリー、およびシステムの担当者ディレクトリー・プロバイダーを構成して、ユーザー情報を取得することができます。

どの担当者ディレクトリー・プロバイダーを使用するかは、担当者解決に必要なサポートによって決まります。Business Process Choreographer に用意されている担当者割り当て機能をすべて活用するには、Virtual Member Manager を使用します。

すべての担当者ディレクトリー・プロバイダーは、ノード・レベルで使用可能です。

Virtual Member Manager 担当者ディレクトリー・プロバイダー

Virtual Member Manager 担当者ディレクトリー・プロバイダーは、WebSphere Application Server のフェデレーテッド・リポジトリーにアクセスするために使用します。このプロバイダーを使用して、担当者解決の以下の側面を活用できます。

- さまざまなリポジトリーの使用を含む、フェデレーテッド・リポジトリー機能 (リポジトリーには、ファイル・リポジトリーとデータベース・リポジトリー、LDAP ディレクトリー、プロパティー拡張リポジトリー、リポジトリーの統合などがあります)
- エスカレーションを知らせる E メール通知
- 不在者の代替
- 事前定義の担当者割り当て基準のすべて

Lightweight Directory Access Protocol (LDAP) 担当者ディレクトリー・プロバイダー

LDAP 担当者ディレクトリー・プロバイダーは、WebSphere Application Server を使用せずに LDAP ディレクトリーに直接アクセスするために使用します。ほとんどの場合、WebSphere Application Server セキュリティー・レルムは「スタンドアロン LDAP レジストリー」に設定されており、LDAP 担当者ディレクトリー・プロバイダーで参照されるのと同じ LDAP ディレクトリーを指すよう構成されています。このプロバイダーを使用して、担当者解決の以下の側面を活用できます。

- エスカレーションを知らせる E メール通知
- 事前定義の担当者割り当て基準のすべて

ユーザー・レジストリーの担当者ディレクトリー・プロバイダー

ユーザー・レジストリーの担当者ディレクトリー・プロバイダーを WebSphere Application Server とともに使用して、ローカル・オペレーティング・システム、スタンドアロン LDAP レジストリー、またはスタンドアロン・カスタム・レジストリーの担当者ディレクトリーにアクセスできます。使用する担当者ディレクトリーは、アプリケーション・サーバーのセキュリティー・レルムの構成に応じて異なります。このプロバイダーを使用して、担当者解決の以下の側面を活用できます。

- Business Process Choreographer に対する担当者ディレクトリー・プロバイダーの最小構成。これは、アプリケーション・サーバーのセキュリティー・レルムによってリポジトリーが決定するためです。
- 事前定義の担当者割り当て基準の制限されたセット。ユーザー・レジストリーの担当者ディレクトリー・プロバイダーは、ユーザーおよびグループを解決できますが、従業員と管理者の関係、ユーザー・プロパティー、または E メール・アドレスは解決できません。

システムの担当者ディレクトリー・プロバイダー

システムの担当者ディレクトリー・プロバイダーでは、担当者解決のサポートが制限されています。システム・プロバイダーではハードコーディングされた照会のみがサポートされるため、テストの目的のみに適しています。

すべての担当者ディレクトリー構成では、WebSphere Application Server の管理セキュリティーとアプリケーション・セキュリティーが使用可能になっていなければなりません。

各担当者ディレクトリー・プロバイダーは、1 つ以上の担当者ディレクトリー・プロバイダー構成に関連付けることができます。LDAP 担当者ディレクトリー・プロバイダーを除くすべての構成は、すぐに使用できます。Virtual Member Manager 担当者ディレクトリー・プロバイダーの場合、WebSphere Application Server でフェデレーテッド・リポジトリー機能を構成する必要があります。LDAP プロバイダー構成の場合、必須の接続パラメーターを設定する必要があります。さらに、LDAP プロバイダー構成用に変換ファイルをカスタマイズする必要があります。

各構成は、Java Naming Directory (JNDI) 名で一意的に識別されます。JNDI 名は、タスク・テンプレート定義と担当者ディレクトリー構成との間のリンクであり、タスク・ロールへの担当者割り当ての解決に使用されます。タスク・テンプレートの構成名を指定するには、WebSphere Integration Developer を使用します。実行時にタスク作成 API を使用してタスクを定義する場合は、構成名を API に直接指定できます。さまざまなタスク・テンプレートがさまざまな担当者ディレクトリー構成を参照することができます。

タスク・テンプレートがデプロイされると、デプロイされたテンプレートの存続時間中は担当者ディレクトリー構成名が固定されます。テンプレートに関連付けられた担当者ディレクトリーを変更する必要がある場合は、WebSphere Integration Developer を使用して、タスク・テンプレート定義に対して定義されている担当者ディレクトリー構成の JNDI 名を変更し、テンプレートをもう一度デプロイしてください。

担当者割り当て基準の担当者照会への変換

アプリケーションがデプロイされると、担当者割り当て基準定義は、担当者ディレクトリー構成に固有の一連の照会に変換されます。結果の担当者照会は、タスク・テンプレートとともに Business Process Choreographer データベースに保管されます。

担当者ディレクトリーとして Virtual Member Manager を使用するときは、カスタム担当者割り当て基準を定義する場合に限り、変換 XSL ファイル内の事前定義マッピングを変換する必要があります。

変換 (XSLT) ファイルには、担当者割り当て基準を変換するための指示が含まれています。個々の担当者ディレクトリー構成は、変換ファイルに関連付けられ、特定のリポジトリーに固有の担当者照会を生成します。各照会はそれぞれの担当者ディレクトリー・プロバイダーによって実行され、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。担当者ディレクトリー・プロバイダーで使用できる定義済み照会は、プロバイダー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。

担当者ディレクトリー構成には、以下のデフォルト変換ファイルが用意されています。

- LDAP 担当者ディレクトリー・プロバイダー用の LDAPTransformation.xsl
- Virtual Member Manager 担当者ディレクトリー・プロバイダー用の VMMTransformation.xsl
- ユーザー・レジストリー担当者ディレクトリー・プロバイダー用の UserRegistryTransformation.xsl

- システム担当者ディレクトリー・プロバイダー用の SystemTransformation.xsl および EverybodyTransformation.xsl

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。

特定の担当者ディレクトリー・プロバイダーの担当者照会

担当者ディレクトリー・プロバイダーで提供される、一連のリポジトリー固有の照会は、対応する担当者ディレクトリーからユーザー情報を取得する際に使用できるメソッドに対応しています。この一連の照会を使用して、以下の例に示すようなさらに複雑な照会を形成することができます。

- 照会結果を結合して、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します。例えば、LDAP 担当者ディレクトリー・プロバイダーでは、以下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識別名 (DN):

    ```
    <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、および指定したユーザーの識別名のリストに対して、以下のように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>
    ```

- 現在の結果リストから照会結果を除去します。例えば、以下のコード断片では、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" を除去する方法を示しています。

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

- 1 つの照会から得られた照会結果を使用して、後続の照会の振る舞いに影響を及ぼします。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>
  ```

以上の結合ルールに従って構成した担当者照会は、担当者ディレクトリー・プロバイダーで実行できます。

不在者の代替

代替フィーチャーを使用すると、自分自身、または自分が管理しているグループのメンバーに対して不在設定を指定できます。代替ポリシーでは、不在ユーザーに割り当てられているタスクとエスカレーションの処置方法が定義されます。

代替ポリシーは、タスク・テンプレートのモデル化時に定義されます。1 つのタスク・テンプレートに関連付けられているすべてのタスク・ロールには、同じポリシーが適用されます。タスク・テンプレートがデプロイされると、その後にそのポリシーを変更することはできません。

ユーザーが不在の場合は、代替ポリシーが担当者解決の結果に適用され、不在ユーザーの代わりに作業項目を受け取る人が決定されます。代替ポリシーは、担当者割り当て基準を持つタスク・ロールにのみ適用されます。つまり、タスクのオリジネーター、スターター、または所有者は、代替の対象とはなりません。同様に、担当者割り当て基準が更新されると、代替も更新されます。

代替の時間を制限するには、開始点および終了点を指定します。開始点のみが指定されると、ユーザーは指定された開始点から不在であるとみなされます。

特定の代替ポリシーに応じて、以下のアクションが適用されます。

代替なし (デフォルト)
ユーザーのセットは変更されず、そのままです。

不在ユーザーを代理人で置換

- 在社しているすべてのユーザーには、そのユーザー自身が使用されます。
- 不在のすべてのユーザーには、在社している最初の代理人が使用されます。
- いずれのユーザーおよびいずれの代理人も不在の場合は、デフォルトの担当者割り当て規則が適用されます。

在社しているユーザーを優先

- 在社しているすべてのユーザーには、そのユーザーが使用されます。
- 代替は考慮されません。
- いずれのユーザーも不在の場合は、元のユーザー・セットが使用されます。つまり、ユーザーが不在であるという事実は無視されます。

代替フィーチャーには、担当者ディレクトリーとして仮想メンバー・マネージャーが必要です。代替に対して仮想メンバー・マネージャーを使用可能にするには、WebSphere Application Server でフェデレーテッド・リポジトリーをアクティブなセキュリティー・レルムとして構成する必要があります。管理コンソールで Human Task Manager に対する代替を有効にしてください。デフォルト以外の代替ポリシーを持つタスク・テンプレートを仮想メンバー・マネージャー以外の担当者ディレクトリー・プロバイダーにデプロイすると、デプロイメントは失敗します。

デフォルトの担当者割り当てと継承ルール

特定のタスク・ロールに担当者割り当て基準を定義していない場合、または担当者解決が失敗するかあるいは担当者解決が結果を返さない場合は、デフォルトの担当

者割り当てが実行されます。デフォルトの割り当ては、インライン・タスクとスタンドアロン・タスクでは異なっています。

継承ルールは、担当者が既に別のロールに割り当てられている場合に、その割り当てに基づいて担当者を特定のロールに自動的に割り当てるために適用されます。担当者割り当てによって決定されるユーザーに加え、継承ルールにより、実質的にロールにユーザーが追加されることになります。これらのルールは、インライン・タスクとスタンドアロン・タスクでは異なっています。

インライン・タスク

以下の表は、インライン・タスクのデフォルトの担当者割り当てを示しています。

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 ...**	**担当者割り当てが失敗した場合...**
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読者になります。
- 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読者になります。
- 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読者になります。
- エスカレーション受信側が、エスカレートしたタスクの読者になります。

スタンドアロン・タスク

以下の表は、スタンドアロン・タスクのデフォルトの担当者割り当てを示しています。

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 ...	担当者割り当てが失敗した場合...
タスク管理者	オリジネーターが管理者になる	この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	この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
エディター	エディターなし	エディターなし
読者	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、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読者になります。
- 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読者になります。
- エスカレーション受信側が、エスカレートしたタスクの読者になります。

Business Flow Manager API を使用してメソッドが呼び出されると、BPESystemAdministrator ロールのメンバーは管理者権限を付与され、BPESystemMonitor ロールのメンバーは読者権限を付与されます。Human Task Manager API によってメソッドが呼び出されると、TaskSystemAdministrator ロールのメンバーは管理者権限を付与され、TaskSystemMonitor ロールのメンバーは読者権限を付与されます。

担当者割り当て基準と担当者照会結果

担当者割り当て基準は、タスク許可のロールに関連付けられています。担当者割り当て基準から派生する担当者照会は、デプロイ済みのタスク・テンプレート、すなわちタスク・インスタンスの一部として保管されます。タスクの実行中は、許可のロールには、担当者をタスクに割り当てることができるように、関連付けられている担当者照会の解決が必要です。

担当者割り当て基準を変更する必要がある場合は、WebSphere Integration Developer でタスク定義を変更し、タスク・テンプレートをもう一度デプロイする必要があります。

担当者照会の結果は担当者ディレクトリーの内容によって異なり、その内容は時間の経過とともに変更されることがあります。例えば、新規メンバーが担当者グループに追加されることがあります。担当者ディレクトリーの変更を反映させるには、以下のいずれかの方法で担当者照会を更新する必要があります。

- 管理者によって明示的に更新

  管理者は管理コンソールまたは管理コマンドのいずれかを使用して、担当者照会結果を更新することができます。以下のアクションには、それを実行するためのコマンドがあります。
  - すべての担当者照会結果の一括更新
  - タスク・テンプレートに関連付けられているすべての担当者照会結果の更新
  - 現在の結果に特定のユーザー ID を含んでいる担当者照会結果の更新
- 有効期限が切れた担当者照会の計画的更新によるトリガー

  このアプローチは、以下のパラメーターに基づいています。
  - 担当者照会結果のタイムアウト値 (T_{out})。
  - 担当者照会の更新スケジュール。スケジュールの定義 (例えば、毎週月曜日の午後 1 時、または各平日の午前 0 時) には WebSphere Application Server CRON 構文が使用されます。

  以下のパラメーターによって、担当者照会の自動更新方法が決まります。
  - 照会が初めて実行されるか、または照会が更新されると、有効期限を示すタイム・スタンプが照会結果に付けられます ($t_{exp} = t_{current} + T_{out}$)。
  - 照会更新デーモンが呼び出されると、期限切れという結果になっているすべての担当者照会が再実行されます。

  タイムアウト値は、スケジュールの更新間隔を越えて設定できます。例えば、タイムアウト値を 24 時間に設定し、更新間隔を 1 時間に設定できます。このようにすることで、担当者照会に対する更新を 1 日全体にわたって分散させ、すべての担当者照会結果を一度に更新することによるオーバーヘッドを回避することができます。

担当者割り当ての共用

特定のタスク・ロールについて、タスク・テンプレートのすべてのインスタンスに同じ担当者割り当て基準が使用されます。これは、すべてのタスク・インスタンスは同じタスク・テンプレートからインスタンス化されているためです。担当者照会を再実行するのを避けるため、照会の結果はタスク・テンプレートのタスク・インスタンス全体で共用されます。

結果の共用が適用されるのは、担当者割り当て基準定義に固定パラメーター値が含まれる場合のみです。このような値 (例えば、グループ名: `cn=group1, cn=groups`) は、担当者照会が解決されるタスク・インスタンスのコンテキストに関係なく、対応する担当者照会結果が同じであることを暗黙指定します。

担当者割り当て基準定義に置換変数が含まれる場合、共用の有効範囲は同じ置換変数値を持つ担当者割り当てに狭められます。例えば、パラメーター値はタスクの入力メッセージの一部によって異なる場合があります。タスク・インスタンスが異なれば、含まれる入力メッセージも異なるため、担当者照会のパラメーター値も異なります。

担当者照会結果を後処理する場合、デフォルトではこれらの結果に共用は適用されません。後処理した結果を共用するには、管理コンソールで、次の手順を実行します。

1. 「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」をクリックします。
2. 「**追加プロパティー**」セクションで「**カスタム・プロパティー**」をクリックして、**Staff.PostProcessorPlugin.EnableResultSharing** カスタム・プロパティーの値を `true` に変更し、変更を保存します。
3. サーバーまたはクラスターを再始動して、変更を有効にします。

関連概念

70 ページの『ヒューマン・タスクの置換変数』
置換変数は、実行時に解決されるエレメントの値を参照するためにヒューマン・タスクの定義で使用されます。これらの変数は、タスクに割り当てられた担当者や、タスクのカスタム・プロパティーなどの、タスクおよびプロセス関連のデータを表します。このデータは、実行時に、タスク・インスタンスのライフ・サイクルの全体または一部で使用できます。

第 2 部 Business Process Choreographer の計画および構成

第 3 章 Business Process Choreographer の構成計画

Business Process Choreographer のセットアップと構成パラメーターを計画します。

手順

1. 『トポロジー、セットアップ、および構成パスの計画』を実行します。
2. 選択した構成パスに応じて、以下のいずれかを実行します。
   - 『基本サンプル』の場合は、 143 ページの『基本サンプル Business Process Choreographer 構成の作成の計画』を実行してください。
   - 『組織付きサンプル』の場合は、 144 ページの『サンプル組織を含むサンプル Business Process Choreographer 構成の作成の計画』を実行してください。
   - 『非実動デプロイメント環境』の場合は、 145 ページの『非実動デプロイメント環境構成の計画』を実行してください。
   - 『実動デプロイメント環境』の場合は、 147 ページの『管理コンソールのデプロイメント環境ウィザードを使用するための計画』を実行してください。
   - 『柔軟なカスタム構成』の場合は、 152 ページの『Business Process Choreographer カスタム構成の計画』を実行してください。

タスクの結果

これで、Business Process Choreographer を構成するために必要なすべての事項の計画が完了しました。

関連概念

184 ページの『Business Process Choreographer の概要』
Business Flow Manager と Human Task Manager の機能について説明します。

トポロジー、セットアップ、および構成パスの計画

選択するトポロジーとセットアップによって、使用できる Business Process Choreographer 構成パスが決まります。

このタスクについて

構成パスごとに、複雑さ、柔軟性、および各種トポロジーとデータベースのサポートが異なります。

手順

1. 次の 5 つの異なる構成パスから選択する必要がある点に注意してください。
   - 『基本サンプル』
   - 『組織付きサンプル』
   - 『非実動デプロイメント環境』
   - 『実動デプロイメント環境』
   - 『柔軟なカスタム構成』

ほとんどの構成パスでは、構成ツールを選択できます。

2. Business Process Choreographer の構成にはさまざまな構成ツールを使用できる点に注意してください。

インストーラー または プロファイル管理ツール

非実動システムを最も簡単に作成できます。また、計画しておく必要があることが最も少ないツールです。

- 『基本サンプル』構成には、次の Business Process Choreographer コンポーネントが含まれています。
  - Business Process Choreographer
  - Business Process Choreographer Explorer (レポート作成機能付き)
  - レポート作成機能のための Business Process Choreographer Event Collector
- 『組織付きサンプル』構成には、サンプル組織の 15 人のユーザーを使用して事前に構成されている担当者ディレクトリーも含まれており、代替とグループ作業項目が有効になっています。
- 『非実動デプロイメント環境』構成では、クラスターで Business Process Choreographer を容易に構成できますが、Business Process Choreographer に専用のデータベースを設定できません。代わりに、共通 WPRCSDB データベースが使用されます。

管理コンソールのデプロイメント環境ウィザード

このウィザードを使用して、デプロイメント環境パターンに基づいて『実動デプロイメント環境』の Business Process Choreographer 構成を作成できます。

管理コンソールの「Business Process Choreographer の構成」ページ

管理コンソールのこのページでは、サーバーまたはクラスター上の『柔軟なカスタム構成』の Business Process Choreographer 実動システムを構成できます。多数の構成パラメーターを設定できます。これには、詳細な計画が必要です。このページでは、Business Process Choreographer Explorer は構成されません。これは、管理コンソール内の独自の構成ページを使用するか、スクリプトを実行することによって構成できます。この構成パスは、実動システムの構築に最も適しています。

bpeconfig.jacl 構成スクリプト

このスクリプトを使用して、特定のサーバーまたはクラスター上で、『柔軟なカスタム構成』の Business Process Choreographer 実動システムと必要なすべてのリソースを構成できます。このスクリプトは対話式に実行できます。また、必要なパラメーターをすべて指定する場合は、バッチ・モードで実行して繰り返し可能な自動化を実現できます。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ます。また、オプションで Business Process Choreographer Explorer を構成することもできます。ここには、Business Process Choreographer Explorer レポート作成機能が組み込まれます。データベース・システムによっては、リモート・データベースも作成できます。この構成パスは、実動システムの構築に最も適しています。

clientconfig.jacl 構成スクリプト

このスクリプトを使用できるのは、Business Process Choreographer

Explorer を構成する場合に限られます (オプションのレポート作成機能を設定する場合も設定しない場合も対象になります)。

3. 一部の構成パスには、実動システムへの適合を限定する制約事項があります。以下に例を挙げます。
   - 1 つのサンプル構成を試験的に使用した場合、実動システムに適した構成を作成する前に、このサンプル構成を除去する必要があります。
   - Derby Embedded データベースまたは共通 WPRCSDB データベースを使用する構成を作成する場合、この構成はハイパフォーマンス・システムには適していません。別のハイパフォーマンス・データベースを使用する新規構成を作成する前に、この構成を除去する必要があります。
   - メッセージ・ストアがファイル・ストアまたは Derby 組み込みのデータ・ストアを使用する場合は、Network Deployment 環境にプロファイルを統合できません。プロファイルを統合するには、Business Process Choreographer 構成を完全に除去してから、メッセージ・ストアとしてリモート操作でアクセス可能なデータベースを使用する新規構成を作成する必要があります。
4. バージョン 6.1.2 までの Business Process Choreographer Observer を使用していた場合は、この機能が今では Business Process Choreographer Explorer に統合されていることに注意してください。 これは今では Business Process Choreographer Explorer レポート作成機能と呼ばれ、 Business Process Choreographer Explorer クライアントの「**レポート**」タブからアクセスできます。レポート作成機能が使用する URL は、Business Process Choreographer Explorer の URL と同じです。

   管理コンソールで Business Process Choreographer Explorer を構成する場合や、bpeconfig.jacl 構成スクリプトまたは clientconfig.jacl 構成スクリプトを使用する場合は、Business Process Choreographer Explorer レポート作成機能を構成することもできます。

   既存の Business Process Choreographer 構成をマイグレーションしても、Business Process Choreographer Observer の構成はマイグレーションされません。Business Process Choreographer Explorer レポート作成機能を使用するには、263 ページの『Business Process Choreographer Explorer レポート作成機能のマイグレーション後の使用可能化』の説明に従って、この機能を使用可能にする必要があります。
5. 使用する構成パスを決定する上での主な基準を確認します。 次の表で、選択できる構成パスと制約を確認してください。

表 14. 構成パスの選択基準

選択項目			制約事項		適切な構成パス
実動システムを計画しているか	**デプロイメント・ターゲットの形態**	**Business Process Choreographer 構成のタイプ**	**別の BPEDB データベースを使用できるか**	**メッセージング・エンジンでサポートされているメッセージ・ストア**	**構成パス名、ツール、およびオプション**
いいえ	スタンドアロン・サーバー	基本サンプル (サンプル組織なし)	はい。ただし Derby Embedded のみ。	Derby Embedded のみ	**『基本サンプル』** 次のいずれかを使用: • インストーラー • プロファイル管理ツール オプションを選択する。 • **スタンドアロン・サーバー・プロファイル** • **標準的** • **管理セキュリティーを有効にする**
		15 名の担当者で構成される組織を含むサンプル (担当者の代替が使用可能)。 このサンプルは、WebSphere テスト環境を組み込むときに WebSphere Integration Developer で使用可能なサンプルと同じです。		Derby Embedded、ファイル・ストア、または WPRCSDB	**『組織付きサンプル』** 次のコンポーネントを使用: • プロファイル管理ツール オプションを選択する。 • **スタンドアロン・サーバー・プロファイル** • **拡張** • **開発テンプレートからサーバーを作成する (Create server from development template)** • **管理セキュリティーを有効にする** • **サンプル Business Process Choreographer の構成**
	クラスター	選択できるデプロイメント環境パターン: • リモート・メッセージングおよびリモート・サポート • リモート・メッセージング • 単一クラスター	いいえ、WPRCSDB を共有します。WPRCSDB は、Derby Embedded および Microsoft® SQL Server 以外の任意のデータベースです。	WPRCSDB を共有します。WPRCSDB は、ファイル・ストアおよび Derby Embedded 以外のサポートされている任意のデータベースです。	**『非実動デプロイメント環境』** 次のいずれかを使用: • インストーラー • プロファイル管理ツール 「**デプロイメント環境**」を選択する。

表 14. 構成パスの選択基準 (続き)

選択項目			制約事項		適切な構成パス
実動システムを計画しているか	**デプロイメント・ターゲットの形態**	**Business Process Choreographer 構成のタイプ**	**別の BPEDB データベースを使用できるか**	**メッセージング・エンジンでサポートされているメッセージ・ストア**	**構成パス名、ツール、およびオプション**
はい	クラスター	選択できるデプロイメント環境パターン: • リモート・メッセージングおよびリモート・サポート • リモート・メッセージング • 単一クラスター • カスタム	はい。Derby Embedded を除くサポートされているすべてのデータベース	ファイル・ストアおよび Derby Embedded を除く、サポートされているすべてのデータベース	**『実動デプロイメント環境』** 次のコンポーネントを使用: • 管理コンソール 「**デプロイメント環境**」を選択する。
	クラスター	柔軟なカスタム構成	はい。サポートされている任意のデータベース	ファイル・ストアおよび Derby Embedded を除く、サポートされているすべてのデータベース	**『柔軟なカスタム構成』** 次のいずれかを使用: • bpeconfig.jacl スクリプト • 管理コンソールの「Business Process Choreographer の構成」ページ
	スタンドアロン・サーバー			サポートされている任意のデータベース、またはファイル・ストア	

注: 実動システムに適していない構成を作成するときに、実動システムの作成向けに推奨されている構成パスを使用することもできます。
次のオプションを検討します。

a. 実動システムを構成するかどうかを決定します。 一般に実動システムでは、ハイパフォーマンス、スケーラビリティー、およびセキュリティーが必要です。Business Process Choreographer の場合、実動システムに専用の BPEDB データベース (Derby 以外) が必要です。

b. Business Process Choreographer のデプロイメント・ターゲットとして、スタンドアロン・サーバーとクラスターのいずれを使用するかを決定します。

c. 実動システムを構築しない場合は、スタンドアロン・サーバーのサンプル構成が要件に対応するかどうかを確認します。 対応する場合は、担当者割り当てと担当者の代替を使用可能にするためにサンプル担当者ディレクトリー (サンプル組織が取り込まれています) をサンプルに組み込むかどうかを決定します。

注: サンプル担当者ディレクトリーでは、フェデレーテッド・リポジトリー用に構成されたデフォルトのファイル・レジストリーが使用され、すべてのサンプル担当者のパスワードが「wid」に設定されています。WebSphere 管理ユーザー ID も、プロファイル作成時に指定されたパスワードを使用してこのディレクトリーに追加されます。サンプル構成が作成された後、管理コンソールを使用して「**ユーザーおよびグループ**」をクリックしてから「**ユーザーの管理 (Manage Users)**」または「**グループの管理 (Manage Groups)**」をクリックすることで、使用可能なユーザーおよびグループを表示させることができます。

d. クラスターで Business Process Choreographer を構成する場合は、パフォーマンスの要件に基づいて、メッセージング・エンジンとサポート・アプリケーション (Business Process Choreographer Explorer、 Common Event Infrastructure など) に専用のクラスターを設定するか、それともクラスターを共有するかを決定します。標準のデプロイメント環境パターンは以下のとおりです。

リモート・メッセージングおよびリモート・サポート
3 つのクラスターが使用されます。アプリケーション、メッセージング・エンジン、およびサポート・アプリケーション用にそれぞれクラスターが 1 つずつ使用されます。

リモート・メッセージング
アプリケーションとサポート関数に 1 つのクラスターが使用されます。2 番目のクラスターはメッセージング・エンジン用に使用されます。

単一クラスター
アプリケーション、メッセージング・エンジン、およびサポート・アプリケーションに対して 1 つのクラスターが使用されます。

カスタム
柔軟性の高いセットアップです。

e. Business Process Choreographer に専用 BPEDB データベースを使用するかどうかを決定します。

f. Business Process Choreographer は、SCA が使用するメッセージ・ストアと同じタイプのメッセージ・ストアを使用します。

- SCA で FILESTORE が使用される場合、Business Process Choreographer でも FILESTORE が使用されます。
- SCA で Derby Embedded データベースが使用される場合、Business Process Choreographer ではそれ独自の Derby Embedded データベースが使用されます。
- SCA でそれら以外のデータベースが使用される場合、Business Process Choreographer では同じデータベース内でそれ独自のスキーマを使用します。

6. Business Process Choreographer Explorer レポート作成機能 (Business Process Choreographer Explorer に統合) を使用する場合は、Business Process Choreographer 構成を作成する際に同時にこれを構成することも、後で作成することもできます。また、Business Process Choreographer Explorer レポート作成機能が BPEDB データベースを使用するか、あるいは専用の OBSRVRDB データ

ベースを使用するかどうかを決定します。Business Process Choreographer Explorer レポート作成機能のコンポーネントのトポロジーについても計画します。詳細な計画を実行するには、 179 ページの『Business Process Choreographer Explorer レポート作成機能の計画』を実行します。

7. WebSphere Portal Server または任意のカスタムの WebSphere Process Server クライアントから Business Process Choreographer にアクセスする場合は、 182 ページの『リモート・クライアント・アプリケーションの計画』を実行します。
8. アプリケーション・セキュリティーを有効にしていて、リモート EJB メソッドを呼び出す長期実行のプロセスがある場合、Common Secure Interoperability Version 2 (CSIv2) のインバウンド認証を構成するときに CSIv2 の ID アサーションを有効にする必要があります。
9. ヒューマン・タスクを使用する場合は、WebSphere 管理セキュリティーとアプリケーション・セキュリティーの両方を有効にする必要があります。

タスクの結果

これでトポロジーの計画が完了し、使用する構成パスと構成ツールが決定しました。

基本サンプル Business Process Choreographer 構成の作成の計画

この基本サンプルはスタンドアロン・サーバーを対象としており、サンプル組織は含まれていません。

始める前に

137 ページの『トポロジー、セットアップ、および構成パスの計画』を行い、『基本サンプル』構成パスを選択している。

手順

1. インストーラーまたはプロファイル管理ツールのどちらを使用してサンプルを作成するかを決定します。
2. プロファイル管理ツールを使用することを決定した場合、Business Process Choreographer メッセージング・エンジンが、ファイル・ストア、組み込み Derby データベース、または共通の WPRCSDB データベースのいずれを使用するかどうかを決定します。
3. Human Task Manager でエスカレーションの通知を E メールで送信できるようにするには、以下を計画します。
   - 使用可能なローカル Simple Mail Transfer Protocol (SMTP) メール・サーバーがない場合は、後で適切なメール・サーバーを指し示すようにメール・セッションを変更することを計画します。
   - E メールの送信者アドレスの変更を計画します。変更しない場合は、ダミー送信者アドレスが使用されます。
4. このサンプル構成では、各種 Business Process Choreographer ユーザー ID として WebSphere 管理者ユーザー ID とパスワードが使用される点に注意してください。

タスクの結果

基本サンプル Business Process Choreographer 構成の作成の計画が完了しました。

サンプル組織を含むサンプル Business Process Choreographer 構成の作成の計画

このサンプルには、15 名の担当員からなるサンプル組織が含まれており、スタンドアロン・サーバーで担当者の割り当てと代替を試験的に使用する場合に適しています。このサンプルは、WebSphere テスト環境を組み込むときに WebSphere Integration Developer で使用可能なサンプルと同じです。

始める前に

137 ページの『トポロジー、セットアップ、および構成パスの計画』を行い、『組織付きサンプル』構成パスを選択している。

このタスクについて

このサンプル Business Process Choreographer 構成に必要な計画は最小限の計画です。

手順

1. Business Process Choreographer メッセージング・エンジンが、ファイル・ストア、組み込み Derby データベース、または共通の WPRCSDB データベースのいずれを使用するかどうかを決定します。
2. このサンプルは、プロファイル管理ツールでのみ作成できる点に注意してください。 このサンプルを取得するには、以下のオプションを選択する必要があります。
   - **スタンドアロン・サーバー・プロファイル**
   - **拡張**
   - **開発テンプレートからサーバーを作成する (Create server from development template)**
   - **管理セキュリティーを有効にする**
   - **サンプル Business Process Choreographer の構成**

   例えば、管理セキュリティーを使用可能にしていない場合、サンプル Business Process Choreographer 構成は作成されません。

   注: サンプル担当者ディレクトリーでは、フェデレーテッド・リポジトリー用に構成されたデフォルトのファイル・レジストリーが使用され、すべてのサンプル担当者のパスワードが「wid」に設定されています。WebSphere 管理ユーザー ID も、プロファイル作成時に指定されたパスワードを使用してこのディレクトリーに追加されます。サンプル構成が作成された後、管理コンソールを使用して「**ユーザーおよびグループ**」をクリックしてから「**ユーザーの管理 (Manage Users)**」または「**グループの管理 (Manage Groups)**」をクリックすることで、使用可能なユーザーおよびグループを表示させることができます。
3. Human Task Manager でエスカレーションの通知を E メールで送信できるようにするには、以下を計画します。

- 使用可能なローカル Simple Mail Transfer Protocol (SMTP) メール・サーバーがない場合は、後で適切なメール・サーバーを指し示すようにメール・セッションを変更することを計画します。
- E メールの送信者アドレスの変更を計画します。変更しない場合は、ダミー送信者アドレスが使用されます。

4. このサンプル構成では、各種 Business Process Choreographer ユーザー ID として WebSphere 管理者ユーザー ID とパスワードが使用される点に注意してください。

タスクの結果

サンプル組織を含むサンプル Business Process Choreographer 構成の作成の計画が完了しました。

非実動デプロイメント環境構成の計画

インストーラーまたはプロファイル管理ツールを使用して、デプロイメント環境パターンに基づいて Business Process Choreographer 構成を作成することを計画します。

始める前に

137 ページの『トポロジー、セットアップ、および構成パスの計画』を行い、『非実動デプロイメント環境』構成パスを選択している。

このタスクについて

デプロイメント環境ウィザードを使用するときには、デプロイメント環境パターンを選択する必要があります。パターンの選択後に、WBI_BPC コンポーネントのデフォルトのデータベース・パラメーターと認証別名を変更し、Business Process Choreographer の他のパラメーターを入力できます。

手順

1. 使用するデプロイメント環境パターンを決定します。
   - **リモート・メッセージングおよびリモート・サポート**
   - **リモート・メッセージング**
   - **単一クラスター**
2. セキュリティー・ステップで入力する Business Process Choreographer JMS 認証別名のユーザー名を計画します。
3. 「**Business Process Choreographer Explorer コンテキスト・ルート**」を計画します。これによって、ブラウザーで Business Process Choreographer Explorer を開く場合に必要な URL の一部が定義されます。
4. Business Process Choreographer ステップのセキュリティー・パラメーターを計画します。以下に示すユーザー ID とグループは、Business Flow Manager と Human Task Manager に使用されます。

   管理者ユーザーおよび管理者グループ

   ビジネス管理者ロールがマップされるユーザー ID リストまたはグループのリスト (あるいはこの両方) を計画します。

モニター・ユーザーおよびモニター・グループ
ビジネス・モニター・ロールがマップされるユーザー ID リストまたはグループのリスト (あるいはこの両方) を計画します。

JMS API 認証ユーザーおよびパスワード
Business Flow Manager メッセージ駆動型 Bean の run-as ユーザー ID。

エスカレーション・ユーザーの認証ユーザーおよびパスワード
Human Task Manager メッセージ駆動型 Bean の run-as ユーザー ID。

クリーンアップ・ユーザー認証のユーザーとパスワード
Business Flow Manager と Human Task Manager のクリーンアップ・サービスの run-as ユーザー ID。そのユーザーは、ビジネス管理者ロールのメンバーでなければなりません。

5. Human Task Manager エスカレーションの E メール・セッションを構成する場合は、Business Process Choreographer ステップの次のパラメーターを計画します。

   メール・トランスポートのホスト
   Simple Mail Transfer Protocol (SMTP) E メール・サービスが設置されている場所のホスト名または IP アドレス。

   メール・トランスポート・ユーザーおよびメール・トランスポート・パスワード
   メール・サーバーで認証が不要な場合は、これらのフィールドを空白のままにすることができます。

   Business Process Choreographer Explorer の URL
   この URL は、生成される E メールにリンクを設定するために使用されます。これにより、E メール通知を受け取るビジネス管理者は、そのリンクをクリックして、関連するビジネス・プロセスまたはヒューマン・タスクを Web ブラウザーで表示することができます。

6. Business Process Choreographer Explorer、Business Space、あるいは Representational State Transfer (REST) API または JAX Web サービス API を使用するクライアントを使用する場合は、REST API および JAX Web サービス API のコンテキスト・ルートを決定してください。
   - Business Flow Manager のデフォルトは `/rest/bpm/bfm` および `/BFMJAXWSAPI` です。
   - Human Task Manager のデフォルトは `/rest/bpm/htm` および `/HTMJAXWSAPI` です。
   - サーバーまたは単一クラスター、あるいは異なる Web サーバーにマップされた複数のクラスターに構成されている場合は、デフォルト値を使用できます。
   - 同じ Web サーバーにマップされた、Network Deployment 環境内の複数のデプロイメント・ターゲットに構成されている場合は、デフォルト値を使用できません。各 Business Process Choreographer 構成のコンテキスト・ルートは、ホスト名とポートの組み合わせごとに固有のものでなければなりません。これらの値は、Business Process Choreographer の構成後に、管理コンソールを使用して手動で設定する必要があります。
7. 担当者割り当てを使用する場合は、175 ページの『担当者ディレクトリー・プロバイダーの計画』を行います。

タスクの結果

非実動デプロイメント環境構成の作成の計画が完了しました。

管理コンソールのデプロイメント環境ウィザードを使用するための計画

実動システムの場合は、Business Process Choreographer のすべての構成パラメーター (個別のデータベースを含む) を計画します。非実動システムの場合は、共用データベースを使用できます。

始める前に

137 ページの『トポロジー、セットアップ、および構成パスの計画』を行い、『実動デプロイメント環境』構成パスを選択している。

このタスクについて

デプロイメント環境ウィザードを使用するときには、デプロイメント環境パターンを選択する必要があります。パターンの選択後に、WBI_BPC コンポーネントのデフォルトのデータベース・パラメーターと認証別名を変更し、Business Process Choreographer の他のパラメーターを入力できます。

手順

1. 構成全体を作成するための十分な情報または権限がない場合は、システムの他の部分に対する責任を持つ担当者に相談して計画します。 以下に例を挙げます。
   - 場合によっては、組織の LDAP サーバーに関する情報を要求する必要があります。また、このサーバーで認証が使用される場合には、ユーザー ID と許可が必要になります。
   - ユーザーにデータベースを作成する権限がない場合は、データベースの計画にデータベース管理者 (DBA) が参加する必要があります。DBA には、カスタマイズして実行するデータベース・スクリプトのコピーが必要です。
2. 153 ページの『セキュリティー、ユーザー ID、および許可の計画』を実行します。
3. 使用するデプロイメント環境パターンを決定します。
   - **リモート・メッセージングおよびリモート・サポート**
   - **リモート・メッセージング**
   - **単一クラスター**
   - **カスタム**
4. 「**カスタム**」デプロイメント環境パターンを選択した場合:
   a. Business Process Choreographer Explorer をインストールするかどうかを決定します。 インストールする場合は、以下について計画します。
      - デプロイする場所。
      - Business Process Choreographer Explorer レポート作成機能を使用する場合は、 Business Process Choreographer Event Collector のデプロイ場所も計画します。
   b. SCA バインディングのコンテキスト・ルートを計画します。

c. 状態オブザーバーと監査ログ記録を有効または無効のいずれにするかを計画します。

5. 以下の専用データベースを使用する場合:
   - Business Process Choreographer 用の BPEDB データベース。ウィザードの、コンポーネント WBI_BPC 用のテーブル行で変更できます。
   - Business Process Choreographer メッセージング・エンジン用の BPEME データベース。ウィザードの、コンポーネント WBI_BPC_ME 用のテーブル行で変更できます。
   - Business Process Choreographer Explorer レポート作成機能用の OBSRVRDB データベース。ウィザードの、コンポーネント WBI_BPCEventCollector 用のテーブル行で変更できます。

   データベースごとに次のパラメーターを計画します。これらのパラメーターは、ウィザードのデータベース・ページで入力します。

   データベース名
   デフォルト値 WPRCSDB の代わりに使用するデータベースの名前 (BPEDB、BPEME、OBSRVRDB など)。デフォルト値を使用すると、共通データベースを共用することになります。デフォルト値は、低パフォーマンス・セットアップにのみ適しています。

   スキーマ
   各データベースで使用するスキーマ修飾子。

   テーブルの作成
   このオプションを選択した場合は、初めてデータベースにアクセスしたときに、表が自動的に作成されます。このオプションが機能するためには、データベースが既に存在している必要があり、データ・ソースを作成するために指定するユーザー名には、データベースに表と索引を作成するための権限が必要です。このオプションを選択しなかった場合、表は自動的に作成されないため、スクリプトを実行して、表を手動で作成する必要があります。実動システムの場合は、このオプションをクリアし、提供される SQL スクリプトを使用してデータベースをセットアップすることを計画します。

   ユーザー名とパスワード
   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。このユーザー ID の権限で、データベース内に表と索引を作成できる場合は、表を自動的に作成するオプションを使用できます。また、必要に応じて、サービス・パックまたはフィックスパックの適用後、データベース・スキーマを自動的に更新できます。

   サーバー
   データベース・サーバーのアドレス。ホスト名または IP アドレスを指定します。

   プロバイダー
   JDBC プロバイダー。

   また、データベース固有の設定も計画します。この設定は、JDBC プロバイダーの「**編集**」ボタンを使用して設定できます。

表 15. データベース固有の設定

データベース / JDBC ドライバー・タイプ	データベース固有の設定
DB2® UDB - Universal ドライバー	• ユーザー名 • パスワード • データベース名 • スキーマ名 • サーバー名 • サーバー・ポート番号 • ドライバー・タイプ • 説明 • テーブルの作成
DB2 for i5/OS® - ツールボックス・ドライバー	• ユーザー名 • パスワード • データベース名 • 集合名 • サーバー名 • 説明 • テーブルの作成
DB2 for z/OS® V8 および V9	• 実装タイプ - 接続プール・データ・ソースまたは XA データ・ソース • ユーザー名 • パスワード • データベース名 • スキーマ名 • サーバー名 • サーバー・ポート番号 • ストレージ・グループ • 説明
Derby Network Server または Derby Network Server 40	• ユーザー名 • パスワード • 説明 • テーブルの作成 • サーバー名 • サーバー・ポート番号
Derby Embedded または Derby Embedded 40	• 説明 • テーブルの作成

表 15. データベース固有の設定 (続き)

データベース / JDBC ドライバー・タイプ	データベース固有の設定
Microsoft SQL Server - Datadirect、および Microsoft ドライバー	• ユーザー名 • パスワード • データベース名 • サーバー名 • サーバー・ポート番号 • 説明 • テーブルの作成
Informix® Dynamic Server - Universal および DataServer ドライバー	• ユーザー名 • パスワード • サーバー名 • サーバー・ポート番号 • 説明 • テーブルの作成
Oracle - oci ドライバー	• ユーザー名 • パスワード • データベース名 • スキーマ名 • ドライバー・タイプ - oci • 説明 • テーブルの作成
Oracle - thin ドライバー	• ユーザー名 • パスワード • データベース名 • スキーマ名 • サーバー名 • サーバー・ポート番号 • ドライバー・タイプ - thin • 説明 • テーブルの作成

データベースの計画について詳しくは、161 ページの『Business Process Choreographer のデータベースの計画』を参照してください。

6. セキュリティー・ステップで入力する Business Process Choreographer JMS 認証別名のユーザー名を計画します。
7. 「**Business Process Choreographer Explorer コンテキスト・ルート**」を計画します。これによって、ブラウザーで Business Process Choreographer Explorer を開く場合に必要な URL の一部が定義されます。
8. Business Process Choreographer ステップのセキュリティー・パラメーターを計画します。以下に示すユーザー ID とグループは、Business Flow Manager と Human Task Manager に使用されます。

管理者ユーザーおよび管理者グループ
ビジネス管理者ロールがマップされるユーザー ID リストまたはグループのリスト (あるいはこの両方) を計画します。

モニター・ユーザーおよびモニター・グループ
ビジネス・モニター・ロールがマップされるユーザー ID リストまたはグループのリスト (あるいはこの両方) を計画します。

JMS API 認証ユーザーおよびパスワード
Business Flow Manager メッセージ駆動型 Bean の run-as ユーザー ID。

エスカレーション・ユーザーの認証ユーザーおよびパスワード
Human Task Manager メッセージ駆動型 Bean の run-as ユーザー ID。

クリーンアップ・ユーザー認証のユーザーとパスワード
Business Flow Manager と Human Task Manager のクリーンアップ・サービスの run-as ユーザー ID。そのユーザーは、ビジネス管理者ロールのメンバーでなければなりません。

9. Human Task Manager エスカレーションの E メール・セッションを構成する場合は、Business Process Choreographer ステップの次のパラメーターを計画します。

メール・トランスポートのホスト
Simple Mail Transfer Protocol (SMTP) E メール・サービスが設置されている場所のホスト名または IP アドレス。

メール・トランスポート・ユーザーおよびメール・トランスポート・パスワード
メール・サーバーで認証が不要な場合は、これらのフィールドを空白のままにすることができます。

Business Process Choreographer Explorer の URL
この URL は、生成される E メールにリンクを設定するために使用されます。これにより、E メール通知を受け取るビジネス管理者は、そのリンクをクリックして、関連するビジネス・プロセスまたはヒューマン・タスクを Web ブラウザーで表示することができます。

10. Business Process Choreographer Explorer、Business Space、あるいは Representational State Transfer (REST) API または JAX Web サービス API を使用するクライアントを使用する場合は、REST API および JAX Web サービス API のコンテキスト・ルートを決定してください。
    - Business Flow Manager のデフォルトは `/rest/bpm/bfm` および `/BFMJAXWSAPI` です。
    - Human Task Manager のデフォルトは `/rest/bpm/htm` および `/HTMJAXWSAPI` です。
    - サーバーまたは単一クラスター、あるいは異なる Web サーバーにマップされた複数のクラスターに構成されている場合は、デフォルト値を使用できます。
    - 同じ Web サーバーにマップされた、Network Deployment 環境内の複数のデプロイメント・ターゲットに構成されている場合は、デフォルト値を使用できません。各 Business Process Choreographer 構成のコンテキスト・ルートは、ホスト名とポートの組み合わせごとに固有のものでなければなりませ

ん。これらの値は、Business Process Choreographer の構成後に、管理コンソールを使用して手動で設定する必要があります。

11. 担当者割り当てを使用する場合は、 175 ページの『担当者ディレクトリー・プロバイダーの計画』を行います。

タスクの結果

管理コンソールのデプロイメント環境ウィザードを使用するための計画が完了しました。

Business Process Choreographer カスタム構成の計画

管理コンソールの「Business Process Choreographer の構成」ページまたは bpeconfig.jacl 構成スクリプトを使用してカスタム構成を作成するための構成パラメーターおよびオプションを計画します。

始める前に

137 ページの『トポロジー、セットアップ、および構成パスの計画』を行い、『柔軟なカスタム構成』構成パスを選択している。

手順

1. Business Process Choreographer を構成するときに次のどちらを使用するかを確認します。
   - 管理コンソールの「Business Process Choreographer の構成」ページ
   - bpeconfig.jacl 構成スクリプト
2. 構成全体を作成するための十分な情報または権限がない場合は、システムの他の部分に対する責任を持つ担当者に相談して計画します。 以下に例を挙げます。
   - 場合によっては、組織の LDAP サーバーに関する情報を要求する必要があります。また、このサーバーで認証が使用される場合には、ユーザー ID と許可が必要になります。
   - ユーザーにデータベースを作成する権限がない場合は、データベースの計画にデータベース管理者 (DBA) が参加する必要があります。DBA には、カスタマイズして実行するデータベース・スクリプトのコピーが必要です。
3. 153 ページの『セキュリティー、ユーザー ID、および許可の計画』
4. 161 ページの『Business Process Choreographer のデータベースの計画』
5. 174 ページの『Business Flow Manager および Human Task Manager の計画』
6. 175 ページの『担当者ディレクトリー・プロバイダーの計画』
7. 177 ページの『Business Process Choreographer Explorer の計画』
8. 管理コンソールの「Business Process Choreographer の構成」ページを使用する場合は、構成ページに入力するすべての値の計画が完了していることを確認します。
9. bpeconfig.jacl 構成スクリプトを使用する場合:
   a. コマンド行またはバッチ・ファイルに指定する必要があるすべてのオプションとパラメーター値の計画が完了していることを確認します。オプションとパラメーターの要約については 195 ページの『bpeconfig.jacl スクリプトを使

用した Business Process Choreographer の構成』、詳細については 201 ページの『bpeconfig.jacl スクリプト』をそれぞれ参照してください。

b. バッチ・ファイルを使用して bpeconfig.jacl 構成スクリプトを実行する場合は、バッチ・ファイルまたはシェル・スクリプトを作成します。

タスクの結果

これで、Business Process Choreographer のカスタム構成を作成するために必要なすべての事項の計画が完了しました。

次のタスク

191 ページの『第 4 章 Business Process Choreographer の構成』を実行します。

セキュリティー、ユーザー ID、および許可の計画

Business Process Choreographer を構成する場合のユーザー ID と許可を計画します。

このタスクについて

構成時には、さまざまなユーザー ID を使用する必要があり、実行時に使用される他のユーザー ID を指定する必要があります。Business Process Choreographer の構成を始める前に、必ずすべてのユーザー ID を計画および作成してください。

サンプル Business Process Choreographer 構成の場合:

必要な権限は、新規プロファイルを作成する権限のみです。プロファイル管理ツールで標準プロファイル作成オプションを使用し、管理セキュリティーを有効にすると、Business Process Choreographer サンプルも構成されます。他の計画やユーザー ID は不要なので、このタスクはスキップできます。

高セキュリティー構成の場合:

このタスクでの説明に従って、すべてのユーザー ID を詳細に計画する必要があります。

低セキュリティー構成の場合:

非実動システムなど、完全なセキュリティーが不要な場合は、使用されるユーザー ID の数を減らすことができます。すべてのユーザー ID を詳細に計画する必要がありますが、特定のユーザー ID を複数の目的で使用できます。例えば、データベース・スキーマを作成するために使用するデータベース・ユーザー ID は、実行時にデータベースに接続するためのデータ・ソース・ユーザー名としても使用できます。

bpeconfig.jacl スクリプトを使用して Business Process Choreographer を構成する場合:

bpeconfig.jacl スクリプトを実行するために使用されるユーザー ID には、スクリプトが実行する構成アクションで必要とされる権限が付与されている必要があります。そうでない場合は、必要な権限を持つユーザー ID をスクリプトのパラメーターとして指定する必要があり、そのためにすべてのユーザー ID を詳細に計画する必要があります。bpeconfig.jacl スクリプトに対するパラメーターとして指定可能なユーザー ID の場合は、パラメーター名がテーブルに取り込まれます。プロファイルが既に作成されている必要があ

ります。WebSphere 管理セキュリティーが有効になっている場合は、wsadmin ツールを起動するために使用できる configurator ロールの WebSphere 管理者ユーザー ID が必要です。

ヒューマン・タスクを使用する場合:

WebSphere 管理セキュリティーとアプリケーション・セキュリティーの両方を有効にする必要があります。

手順

1. このページのハードコピーを印刷し、計画した値を最後の列に書き込みます。Business Process Choreographer を構成するときの参照用に保管し、将来的に参照できるように記録を残すことができます。
2. Business Process Choreographer を構成するために、WebSphere Process Server で使用するユーザー ID を計画します。

表 16. WebSphere Process Server 用のユーザー ID の計画

ユーザー ID またはロール	ユーザー ID を使用する場合	ユーザー ID の使用目的	ユーザー ID に必要な権限	計画されるユーザー ID
Business Process Choreographer を構成するユーザー	構成	管理コンソールにログオンして管理スクリプトを実行する。	WebSphere 管理セキュリティーが有効になっている場合は、WebSphere 管理者またはコンフィギュレーターのロール。	
		bpeconfig.jacl スクリプトを実行して Business Process Choreographer を構成する場合。	スクリプトを実行する場合は、選択するオプションの必須項目としてユーザー ID も入力する必要があります。詳しくは、201 ページの『bpeconfig.jacl スクリプト』を参照してください。	

3. *install_root* のサブディレクトリーにアクセスする必要があるユーザーを計画します。セキュリティー・ポリシーにより、これらのユーザーにこの種のアクセス権限を付与することが許可されない場合は、ディレクトリー内のファイルのコピーを与える必要が生じます。

表 17. install_root のサブディレクトリーへのアクセスの計画

ユーザー ID またはロール	ユーザー ID を使用する場合	ユーザー ID の使用目的	ユーザー ID に必要な権限	計画されるユーザー ID
データベース管理者	構成	次のデータベースをセットアップするスクリプトの実行 **BPEDB**: Business Process Choreographer 用データベースのデフォルト名。 **OBSRVDB**: Business Process Choreographer Explorer レポート作成機能用データベースのデフォルト名。	bpeconfig.jacl スクリプトを使用して Business Process Choreographer を構成する場合: bpeconfig.jacl により次のディレクトリーのサブディレクトリーに生成される createSchema.sql スクリプト (またはそのコピー) への読み取りアクセス権限: *profile_root*/dbscripts/ProcessChoreographer/	
			データベース・スクリプト・ファイルを確認する場合: ディレクトリー内に作成されたデータベース・スクリプト (またはそのディレクトリー内のファイルのコピー) への読み取りアクセス。ディレクトリーは以下のとおりです。 *install_root*/dbscripts/ProcessChoreographer/*database_type*ここで、*database_type* は以下のいずれかになります。 • DB2zOSV8 • DB2zOSV9	
Integration Developer	カスタマイズ	Lightweight Directory Access Protocol (LDAP) または Virtual Member Manager (VMM) 担当者ディレクトリー・プロバイダーで担当者割り当てを使用するには、サンプル XSL 変換ファイルのコピーをカスタマイズする必要があります。	Staff ディレクトリーまたは次のディレクトリー内のファイルのコピーへの読み取りアクセス。*install_root*/ProcessChoreographer/StaffIntegration Developer では、カスタマイズされた XSL 変換ファイルをサーバーで使用可能にするために、適切なディレクトリーへの書き込みアクセスも必要になります。	

4. Business Process Choreographer が使用するデータベースの作成、構成、およびアクセスに使用するユーザー ID を計画します。

表 18. BPEDB データベースのユーザー ID の計画

ユーザー ID またはロール	ユーザー ID を使用する場合	ユーザー ID の使用目的	ユーザー ID に必要な権限	計画されるユーザー ID
データベース管理者	構成前	BPEDB データベースを作成します。Oracle の場合は、BPEDB データベースを作成します。	データベースを作成します。	

表 18. BPEDB データベースのユーザー ID の計画 (続き)

ユーザー ID またはロール	ユーザー ID を使用する場合	ユーザー ID の使用目的	ユーザー ID に必要な権限	計画されるユーザー ID
bpeconfig.jacl スクリプトを実行するデータベース管理者または管理者	構成	ユーザーまたはデータベース管理者は、組み込み Derby データベースを使用するのでない限り、Business Process Choreographer データベースのスクリプトを実行する必要があります。	BPEDB データベースの場合: テーブル変更、接続、テーブル挿入、および索引、スキーマ、テーブル、テーブル・スペース、およびビューの作成を行います。	
データ・ソースのユーザー名 bpeconfig.jacl スクリプトを使用する場合、これは -dbUser パラメーターです。	構成	「**テーブルの作成**」オプションを選択する場合は、このユーザー ID を使用してデータベース表が作成されます。	「**テーブルの作成**」構成オプションを使用するには、このユーザー ID に、次に示す BPEDB データベースに対する操作の実行権限も付与する必要があります: テーブル変更、接続、テーブル挿入、および索引、テーブル、ビューの作成。	
	実行時	Business Flow Manager と Human Task Manager は、このユーザー ID を使用して BPEDB データベースに接続します。	このユーザー ID に、次に示す BPEDB データベースに対する操作の実行権限も付与する必要があります: 接続、テーブル削除、テーブル挿入、テーブルとビューの選択、テーブルの更新。	
	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・スキーマは、必要に応じてサービスの適用後に自動的に更新されます。この仕組みは、このユーザー ID に必要なデータベース権限が付与されている場合にのみ機能します。権限が付与されていない場合は、スキーマの更新を手動で実行する必要があります。	このユーザー ID に、BPEDB データベースに対する次に示す操作の実行権限も付与する必要があります。テーブルの変更、作成、挿入、および選択、データベースへの接続、索引とビューの作成および除去。	

5. Business Process Choreographer Explorer レポート作成機能を構成する場合は、レポート・データベースの作成、構成、およびアクセスに使用するユーザー ID を計画します。

表 19. レポート・データベースのユーザー ID の計画

ユーザー ID またはロール	ユーザー ID を使用する場合	ユーザー ID の使用目的	ユーザー ID に必要な権限	計画されるユーザー ID
データベース管理者	構成前	レポート・データベースを作成します。 Oracle の場合は、レポート・データベースを作成します。	データベースを作成します。	

表 19. レポート・データベースのユーザー ID の計画 (続き)

ユーザー ID またはロール	ユーザー ID を使用する場合	ユーザー ID の使用目的	ユーザー ID に必要な権限	計画されるユーザー ID
データベース管理者または管理者	構成	setupEventCollector ツール、またはスキーマを作成する SQL スクリプトを実行します。	レポート・データベースの場合: テーブル変更、接続、プロシージャー作成、テーブル挿入、およびテーブル、テーブル・スペース、およびビューの作成。 ユーザー定義関数の Java 実装を使用する場合、ユーザー ID には、JAR ファイルをインストールする権限も必要です。	
Event Collector データ・ソースのユーザー名	実行時	レポート・データベースに接続します。レポート・データベースを使用しており、そのデータベースで BPEDB データベースを使用する場合は、Business Process Choreographer データ・ソースの場合と同じユーザー名を使用します。	データベースに接続します。	

6. Business Process Choreographer のメッセージング・エンジンのメッセージ・ストア用に別のデータベース (Derby Embedded およびファイル・ストア以外) を使用する場合は、データベースへのアクセスに使用するユーザー ID を計画してください。

表 20. 事前構成 BPEME メッセージング・エンジン・データベースのユーザー ID の計画

ユーザー ID	ユーザー ID を使用する場合	ユーザー ID の使用目的	ユーザー ID に必要な権限	計画されるユーザー ID
バス・データ・ソースのユーザー名 bpeconfig.jacl スクリプトを使用する場合、これは -medbUser パラメーターです。	構成および実行時	このユーザー名は、BPEME データベースへの接続と、必要なテーブルおよび索引の作成に使用されます。	このユーザー ID に、次に示す BPEME データベースに対する操作の実行権限も付与する必要があります: 接続、テーブル削除、テーブル挿入、テーブルとビューの選択、テーブルの更新。	

7. Java Message Service (JMS) で使用する Business Process Choreographer のユーザー ID を計画します。

表 21. JMS のユーザー ID の計画

ユーザー ID	ユーザー ID を使用する場合	ユーザー ID の使用目的	ユーザー ID に必要な権限	計画されるユーザー ID
JMS 認証ユーザー	実行時	システム統合バスの認証別名。Business Process Choreographer の構成時に指定する必要があります。 bpeconfig.jacl スクリプトを使用する場合、このユーザー ID とそのパスワードは、-mqUser パラメーターと -mqPwd パラメーターです。	WebSphere ユーザー・レジストリーに存在するユーザー名でなければなりません。Business Process Choreographer バスのバス・コネクター・ロールに自動的に追加されます。	
JMS API 認証ユーザー	実行時	Business Flow Manager JMS API 要求はすべて、このユーザー ID を使用して処理されます。 bpeconfig.jacl スクリプトを使用する場合、このユーザー ID とそのパスワードは、-jmsBFMRunAsUser パラメーターと -jmsBFMRunAsPwd パラメーターです。	このユーザー名は、WebSphere ユーザー・レジストリーに存在するユーザー名でなければなりません。	
エスカレーションの認証ユーザー	実行時	Human Task Manager のエスカレーションはすべて、このユーザー ID を使用して処理されます。 bpeconfig.jacl スクリプトを使用する場合、このユーザー ID とそのパスワードは、-jmsHTMRunAsUser パラメーターと -jmsHTMRunAsPwd パラメーターです。	このユーザー名は、WebSphere ユーザー・レジストリーに存在するユーザー名でなければなりません。	

8. Business Flow Manager および Human Task Manager の Java EE ロールのマップ対象グループまたはユーザー ID を計画します。

表 22. Business Flow Manager および Human Task Manager のセキュリティーのロールの計画

ユーザー ID またはロール	ユーザー ID を使用する場合	ユーザー ID の使用目的	ユーザー ID、グループ、またはこの両方の計画済みリスト
管理者ユーザー	実行時	Business Flow Manager と Human Task Manager の両方のセキュリティー・ロールであるシステム管理者またはシステム・モニターはそれぞれ、ユーザー ID のリスト、グループのリスト、またはこの両方のリストにマップされます。ここで定義される値により、必要とするアクセス権限をこのロールのユーザーに与えるマッピングが作成されます。 bpeconfig.jacl スクリプトを使用する場合、これらのユーザーおよびグループは次のパラメーターに対応します。 • -adminUsers • -adminGroups • -monitorUsers • -monitorGroups	
管理者グループ	実行時		
モニター・ユーザー	実行時		
モニター・グループ	実行時		

9. Business Flow Manager および Human Task Manager のクリーンアップ・サービス、およびプロセス・インスタンス・マイグレーション・ツールなどの管理ジョブで Java EE run-as ロールとして使用するユーザー ID を計画します。このユーザー ID は、表 22で計画した管理者ロールのユーザーまたはグループのメンバーでなければなりません。

表 23. 管理ジョブを実行するためのユーザー ID の計画

ユーザー ID	ユーザー ID を使用する場合	ユーザー ID の使用目的	計画されるユーザー ID
管理ジョブ・ユーザー ID	ランタイム管理	このユーザー ID を使用して、管理ジョブを実行します。 bpeconfig.jacl スクリプトを使用する場合は、このユーザー ID とパスワードが -adminJobUser パラメーターと -adminJobPwd パラメーターに対応します。	

10. ヒューマン・タスクのエスカレーションで特定のビジネス・イベントについての通知 E メールを送信し、使用する Simple Mail Transfer Protocol (SMTP) サーバーで認証を必要とする場合は、E メール・サーバーに接続するために使用するユーザー ID を決定します。

表 24. E メール・サーバーのユーザー ID の計画

ユーザー ID またはロール	ユーザー ID を使用する場合	ユーザー ID の使用目的	ユーザー ID に必要な権限	計画されるユーザー ID
メール・トランスポートのユーザー	実行時	Human Task Manager は、エスカレーション E メールを送信する構成済みメール・サーバーに対する認証でこのユーザー ID を使用します。 bpeconfig.jacl スクリプトを使用する場合、これは -mailUser パラメーターです。パスワードは -mailPwd パラメーターです。	E メールを送信する。	

11. ヒューマン・タスクで担当者割り当てを使用し、簡易認証を使用する Lightweight Directory Access Protocol (LDAP) 担当者ディレクトリー・プロバイダーを使用する場合は、LDAP サーバーへの接続に使用する Java 認証・承認サービス (JAAS) 別名および関連付けられたユーザー ID を計画します。LDAP サーバーで匿名認証を使用する場合、この別名およびユーザー ID は不要です。

表 25. LDAP サーバーの別名およびユーザー ID の計画

ユーザー ID またはロール	ユーザー ID を使用する場合	別名およびユーザー ID の使用目的	ユーザー ID に必要な権限	計画される別名およびユーザー ID
LDAP プラグイン・プロパティー: 認証別名	実行時	この別名は、LDAP サーバーに接続するときに使用するユーザー ID を検索する目的で使用します。この別名 ID は、LDAP プラグインのプロパティーをカスタマイズするときに指定します。例えば、`mycomputer/My LDAP Alias` と指定します。	JAAS 別名は、LDAP ユーザー ID と関連付ける必要があります。	
LDAP ユーザー ID	実行時	このユーザー ID は、LDAP サーバーに接続するときに使用します。	LDAP サーバーで簡易認証を使用する場合は、このユーザー ID で LDAP サーバーに接続できるようにする必要があります。このユーザー ID は、ショート・ネームまたは識別名 (DN) です。LDAP サーバーに識別名が必要な場合は、ショート・ネームを使用することはできません。	

12. 必要な権限を付与して、計画したユーザー ID を作成します。 すべてを自分で作成する権限を持っていない場合は、適切な管理者に要求を実行依頼し、管理者によってユーザー用に作成されるユーザー ID の名前をこのテーブルに入力します。

タスクの結果

Business Process Choreographer の構成時に必要とされるユーザー ID が分かります。

Business Process Choreographer のデータベースの計画

Business Process Choreographer のデータベースを計画します。セットアップによっては、最大 3 つのデータベースを作成するか、またはデータベースをまったく作成しないかを計画しておく必要があります。

このタスクについて

Business Process Choreographer は、他のプロセス・サーバー・コンポーネントまたは他の Business Process Choreographer 構成とデータベースを共用できます。BPEDB データベースは、Business Flow Manager と Human Task Manager で使用されます。実動システムでは、Business Process Choreographer が構成されているデプロイメント・ターゲットごとに専用データベースを設定することを計画します。

Business Process Choreographer の構成が複数ある場合は、それぞれの構成で独自のデータベースまたはデータベース・スキーマが必要になります。Business Process Choreographer の複数の構成で Business Process Choreographer データベース表を共用することはできません。

制約事項: Informix を使用している場合、複数の Business Process Choreographer 構成で同じデータベースを共用することはできません。

Business Process Choreographer Explorer レポート作成機能 (バージョン 6.1.2 までの Business Process Choreographer Observer) を使用する場合は、同じ BPEDB データベースを使用できますが、追加データベースを使用するとパフォーマンスが改善されます。レポート・データベースをセットアップするためのスクリプトの一部には、推奨される名前 OBSRVDB が既に含まれていますが、別の名前を選択することもできます。

Business Process Choreographer メッセージング・エンジンは、SCA メッセージング・エンジンとデータベースを共用するか、または専用の BPEMEDB データベースを使用できます。選択した構成パスでサポートされているデータベースの詳細については、 140 ページの表 14 を参照してください。

手順

1. 実動システムの場合:
   a. パフォーマンスの重要性が高い場合は、 163 ページの『BPEDB データベースの計画』の説明に従い、Business Process Choreographer に個別のデータベースを使用することを計画します。パフォーマンスの重要性が低い場合は、WPRCSDB 共通データベースを使用することを計画します。
   b. Business Process Choreographer Explorer レポート作成機能を使用する場合:
      - ビジネス・プロセスのパフォーマンスに及ぶ照会の影響を最小限に抑えるには、 168 ページの『レポート・データベースの計画』の説明に従い、個別のデータベース (OBSRVDB) を使用するように計画します。
      - それ以外の場合は、BPEDB データベースを使用するように構成することを計画します。
   c. メッセージング・レートが非常に高い大規模クラスターなど、高負荷セットアップの場合、Business Process Choreographer メッセージング・エンジンに個別のデータベースを使用することでパフォーマンスを向上させることを検

討してください。 これによって、データベース・ロギングが並列化され、ボトルネックになることを防ぐことができます。

- 管理コンソールを使用して Business Process Choreographer を構成するときに、Business Process Choreographer のメッセージング・エンジンで別個のデータベースを使用する場合は、172 ページの『メッセージング・エンジン・データベースの計画』を実行します。それ以外の場合は、Service Component Architecture (SCA) が使用するデフォルトのデータベースを使用するように計画します。
- bpeconfig.jacl 構成スクリプトを使用して Business Process Choreographer を構成する場合、Business Process Choreographer は SCA によって使用されるものと同じ種類のメッセージ・ストアを使用します。

d. オプション: データベース設計ツールを使用して、データベース設計ファイルおよび SQL スクリプト・ファイルを対話式で作成します。データベース管理者は、作成されたファイルを使用して、これまでのステップで計画した 3 つのすべてのデータベースを作成することができます。 このツールを使用することには、以下のような大きなメリットがあります。

- ツールを必要に応じて随時実行して、データベース設計パラメーターを調整することができます。提供される SQL テンプレート・ファイルを手動で編集する方法とは異なり、データベース設計パラメーターを破損させるというリスクを回避できます。
- データベース設計ファイルを使用している場合は、次回、新しいバージョンの WebSphere Process Server にマイグレーションするときに、スキーマ更新 SQL スクリプトを生成することができます。
- テスト構成用のデータベース設計ファイルを作成する場合は、設計ファイルのコピーを作成してそれに若干の変更を加え、実動システムのデータベースに使用できるので便利です。
- また、このツールを使用して、3 つのすべてのデータベースのデータ・ソースを定義することができます。ただし、レポート・データベース用のデータ・ソースは手動で構成する必要があります。

重要: データベース設計ツールを使用してデプロイメント環境を構築すると、共通データベースを構成した後に Business Process Choreographer が「完全」なものとして表示されます。これは、有効なデフォルトが存在し、その結果、Business Process Choreographer のテーブルが共通データベースに作成されるためです。ただし、このデフォルトは実動システムには適していません。実動システムでは、必ず、Business Process Choreographer が構成されるデプロイメント・ターゲットごとに専用データベースを構成してください。

2. パフォーマンスよりもセットアップの単純さの重要性の方が高い非実動システムの場合、選択するオプションは、選択した構成パスによって決まります。

- インストーラーまたはプロファイル管理ツールを使用して『基本サンプル』または『組織付きサンプル』の Business Process Choreographer 構成を作成する場合は、別の Derby Embedded BPEDB データベースが作成されます。このデータベースは、Business Process Choreographer Explorer レポート作成機能でも使用されます。Business Process Choreographer のメッセージング・エンジンの場合は、デフォルトで別個の Derby Embedded データベース (BPEME) が作

成されます。プロファイル管理ツールを使用する場合、「**ファイル・ストア**」を使用するか、または WPRCSDB データベースを共用するかを選択することもできます。

- インストーラーまたはプロファイル管理ツールを使用して、Business Process Choreographer 構成が含まれているデプロイメント環境を作成する場合、Business Process Choreographer、Business Process Choreographer Explorer レポート作成機能、および Business Process Choreographer メッセージング・エンジンはすべて WPRCSDB データベースを使用します。したがって、Business Process Choreographer のデータベースに関する計画は不要です。

タスクの結果

これで、Business Process Choreographer 構成のすべてのデータベースの計画が完了しました。

BPEDB データベースの計画

Business Process Choreographer のデータベースを計画します。

このタスクについて

Business Process Choreographer にはデータベースが必要です。サポートされているすべてのデータベース・システムを対象とした、データベース・スキーマを作成および管理するための SQL スクリプトが提供されています。データベースが配備されたら、Business Process Choreographer のデータベースへの JDBC アクセスを構成する必要があります。ご使用のデータベース・システム、トポロジー、インストールの目的、および使用する管理ツールによっては、データベース作成と JDBC アクセス構成の一部またはすべての操作を自動化できます。実動システムでは、Business Process Choreographer に専用のデータベースを確保しておく必要がありますが、パフォーマンスが重要ではない場合は、他の WebSphere Process Server コンポーネントとデータベースを共有するように Business Process Choreographer を構成することもできます。

手順

1. 選択した BPEDB データベースと構成パスに互換性のあることを確認します。以下のデータベースがサポートされています。
   - Derby

     注: Derby 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

   Business Process Choreographer の構成方法を既に決定している場合は、選択した構成パスが、データベースの作成方法に影響します。Business Process Choreographer の構成に使用する構成パスを決定していない場合は、データベースの要件を確認することで、ニーズに対応しない構成パスを除外できます。各構成パスでサポートされているデータベースの詳細については、140 ページの表 14を参照してください。
2. Business Process Choreographer は Derby データベースを使用します。応答ファイルを使用する場合は、DB2 for z/OS を使用します。

3. 実動システムに通常必要なパフォーマンス、スケーラビリティー、セキュリティーを必要としない場合は、WebSphere Process Server に対してローカルなデータベース・サーバー上の単一のテーブル・スペースにデータベース・オブジェクトを作成できます。このようにすると、データベース作成に必要な計画と労力を最小限に抑えることができますが、データベースへのアクセスに使用するユーザー ID にもデータベース管理権限が必要になります。 計画する必要があるオプションは、選択する構成パスにより異なります。

   a. **インストーラー**または**プロファイル管理ツール**を使用してサンプル Business Process Choreographer 構成を取得する場合は、Business Process Choreographer に単独の Derby BPEDB データベースが作成されます。この場合、それ以上の計画は不要です。

   b. 管理コンソールの**デプロイメント環境ウィザード**で Business Process Choreographer を構成する際に、1 つのテーブル・スペースにデフォルト・スキーマが作成されると十分である場合は、提供されている SQL スクリプトのコピーを使用して BPEDB データベースを作成するように計画してください。

   c. **bpeconfig.jacl** ツールを使用して Business Process Choreographer を構成する場合は、以下のいずれか、該当する項目を計画します。

      - bpeconfig.jacl スクリプトを対話モードで実行する場合は、既存のデータベースにテーブルを作成できます。
      - データベース・オブジェクトを作成できる権限が付与されているユーザー ID では、-createDB yes オプションを使用できます。このオプションを指定すると、bpeconfig.jacl スクリプトで、デフォルトのテーブル・スペースにデータベース・オブジェクトを作成する SQL ファイルが生成、実行されます。この場合も、サーバーを停止してから wsadmin ユーティリティーに `-conntype NONE` オプションを使用するように計画してください。
        - Derby データベースをご使用の場合、bepconfig.jacl によりデータベース・インスタンスが作成されます。 DB2 for z/OS データベースをご使用の場合、データベース・インスタンスが既に作成されている必要があります。
        - データベースまたはオブジェクトの作成中にエラーが発生した場合は、-createDB no オプションを使用する場合と同様の方法で生成された SQL スクリプトを使用できます。
      - データベース・オブジェクトを作成できる権限が付与されていないユーザー ID では、-createDB no オプションを使用する必要があります。このオプションを指定すると、bpeconfig.jacl スクリプトで、デフォルトのテーブル・スペースにデータベース・オブジェクトを作成する SQL ファイルが生成されますが、スクリプトの実行は行われません。この場合は、データベース管理者に連絡し、スクリプトのカスタマイズと実行を依頼してください。

      このツールとその他のデータベース・パラメーターの詳細については、 201 ページの『bpeconfig.jacl スクリプト』を参照してください。

   d. 管理コンソールの「**Business Process Choreographer の構成**」**ページ**を使用すると、次の操作を実行できます。

- Business Process Choreographer データベース・オブジェクトを共通データベース WPRCSDB 内に作成するには、Business Process Choreographer データ・ソースのターゲットとしてデフォルトのデータベースを使用するように計画します。
- 既存のデータベースを再利用するには、Business Process Choreographer データ・ソースのターゲットとして既存のデータベースを指定するように計画します。
- 「テーブルの作成」オプションを選択すると、Business Process Choreographer が初めてデータベースを使用するときに、必要なデータベース・オブジェクトをデフォルトのテーブル・スペース内に作成します。このオプションは、DB2 on z/OS データベースおよびリモート Oracle データベースには使用できません。DB2 UDB データベースでこのオプションを使用する場合は、データベースで AUTOMATIC STORAGE YES が有効になっていなければなりません。
- スクリプトを使用してデータベースを作成するには、「テーブルの作成」オプションを使用しないように計画します。

e. ステップ 14 (167 ページ) に進みます。

4. Business Process Choreographer に対して、次の特性を備えた**ハイパフォーマンス**・データベース・セットアップが必要な場合は、以下のステップをすべて実行してください。
   - データベースは Business Process Choreographer 専用で使用されます。
   - データベースは専用サーバーにインストールされていることが理想的ですが、WebSphere Process Server システムのローカルにあってもかまいません。
   - パフォーマンスを向上するためにテーブル・スペースのディスクへの割り振りをカスタマイズできます。
   - データベース管理用のユーザー ID とは異なるユーザー ID を使用してデータベースにアクセスできます。
5. データベースのユーザー ID についてまだ計画していない場合は、 155 ページの表 18 の内容を実行します。
6. ディスクとテーブル・スペースの割り振りを計画します。 データベース・ホストに対し高速ストレージ・サブシステム (Network Attached Storage やストレージ・エリア・ネットワークなど) を使用することが理想的です。実動システムでは、開発およびシステム・テストにおける経験を考慮します。データベースのサイズは、さまざまな要因によって決まります。 Microflow として実行されるプロセスは、ほとんどスペースを使用しません。ただし、各プロセス・テンプレートに数十キロバイトまたは数百キロバイトが必要です。

   個別のディスクを使用する予定であり、ご使用のデータベース・システムでデータベース表を複数のディスクに割り振ることができる場合は、使用するディスクの数とディスクの割り振り方法を計画します。一般に、ハードウェア補助ディスク・アレイは単一ディスクよりもパフォーマンスが優れています。

   DB2 for z/OS の場合は、表ごとにテーブル・スペースが作成され、LOB 列についても追加のラージ・オブジェクト (LOB) テーブル・スペースが作成されます。 1 つのハイパフォーマンス RAID アレイにすべてのテーブル・スペース

を配置できますが、並列アクセスを可能にするには、各テーブル・スペースを異なるファイルに作成する必要があります。特定の数のディスクでは、テーブル・スペースを個別のディスクに割り振るよりも、RAID 構成を使用する方がパフォーマンスが高くなります。 例えば、N 基のプロセッサーを搭載した専用サーバーで稼働する DB2 データベースの場合、次のガイドラインを検討してください。

- テーブル・スペースには、2*N 個のプライマリー・ディスク、2*N 個のミラー・ディスクを備え、ストライプ・サイズが 256 KB の RAID-1 アレイを使用します。
- データベース・トランザクション・ログには、1.5*N 個のプライマリー・ディスク、1.5*N 個のミラー・ディスクを備え、ストライプ・サイズが 64 KB の RAID-1 アレイを使用します。

7. データベース・オブジェクトを作成する SQL スクリプトを実行する前に、ユーザー自身またはデータベース管理者がこのスクリプトをカスタマイズするように計画します。

   - **bpeconfig.jacl** ツールを使用して Business Process Choreographer を構成する場合は、-createDB no オプションを使用します。これにより、このツールで生成される SQL スクリプトが実行されません。生成される SQL ファイルは、ご使用のデータベース用に提供されるオリジナルの SQL ファイルをベースとしていますが、bpeconfig.jacl ツールに対して指定される構成パラメーターのすべてがこの SQL ファイルに事前に入力されるため、必要となるカスタマイズ作業が最小限に抑えられます。
   - 管理コンソールの「**Business Process Choreographer の構成**」**ページ**または**デプロイメント環境ウィザード**で Business Process Choreographer を構成する場合は、「**テーブルの作成**」オプションのチェックを外し、デフォルト・スキーマが作成されないようにしてください。生成される SQL ファイルは、ご使用のデータベース用に提供されるオリジナルの SQL ファイルをベースとしていますが、管理コンソールで入力した構成パラメーターのすべてが、生成された SQL ファイルに事前に入力されるため、必要となるカスタマイズ作業が最小限に抑えられます。

   生成された SQL スクリプトの使用について詳しくは、217 ページの『生成済みの SQL スクリプトを使用した Business Process Choreographer 用のデータベース・スキーマの作成』を参照してください。実施するカスタマイズについて計画できるように、データベースのオリジナルの SQL ファイルのプレビューを表示するには、ご使用のデータベースの SQL スクリプト `createSchema.sql` を見つけて表示します。ただし、このスクリプトは変更しないでください。オリジナルの SQL ファイルは以下のディレクトリーにあります。

   install_root/dbscripts/ProcessChoreographer/*database_type*ここで、*database_type* は以下のいずれかになります。
   - DB2zOSV8
   - DB2zOSV9

8. データベース・サーバーが WebSphere Process Server システムに対しリモートである場合は、Java Database Connectivity (JDBC) ドライバーまたはデータベース・クライアントを WebSphere Process Server システムにインストールするように計画します。

- タイプ 2 JDBC ドライバー: インストールするデータベース・クライアントとインストール先の場所を決定します。
- タイプ 4 JDBC ドライバー: 製品インストールの一部として提供されるドライバーの JAR ファイルを見つけ、インストール先の場所を決定します。

9. データベース・サーバーがプロセス・サーバーに対しローカルである場合は、データベースにアクセスするために必要な JDBC JAR ファイルがデータベース・システムと共にインストールされます。 これらの JAR ファイルを見つけ、その場所を書き留めておきます。
10. DB2 for z/OS を使用する場合は、使用するサブシステムを決定します。 スクリプト・ファイル `createTablespace.sql` および `createSchema.sql` 内のストレージ・グループ名、(サブシステム名ではなく) データベース名、およびスキーマ修飾子を置き換える値を計画します。
11. 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。
12. 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。
13. データベースに対して指定する必要がある次の構成パラメーターの値を決定します。
    - Java Database Connectivity (JDBC) プロバイダーは、タイプ 2 またはタイプ 4 のいずれかです。
    - サブシステム名。
    - ストレージ・グループ名。
    - データベース名。
    - スキーマ修飾子。

      制約事項: Informix データベースを使用する場合は、スキーマ修飾子を使用できるようにするため、ANSI モードでデータベースを作成する必要があります。現時点では、1 つのスキーマのみを使用できます。
    - スキーマを作成するユーザーのユーザー名。
    - データベース・サーバーの名前または IP アドレス。
    - データベース・サーバーが使用するポート番号。これは、タイプ 4 JDBC ドライバーを使用する場合にのみ必要です。
    - 認証別名のユーザー ID とパスワード。これは、jdbc/BPEDB データ・ソースが実行時にデータベースへアクセスするときに使用するユーザー ID です。これらは、bpeconfig.jacl の -dbUser および -dbPwd パラメーターです。
14. 十分な数の並列 JDBC 接続をサポートするように計画します。
    a. 必要な Business Process Choreographer BPEDB データベースへの並列 JDBC 接続の最大数を推定します。これは、ビジネス・プロセスの特性とユーザー数によって異なります。 適切な推測値は、Business Process Choreographer API を介して同時接続可能なクライアントの最大数、JMS アクティベーション・スペック BPEInternalActivationSpec および HTMInternalActivationSpec で定義されている並行エンドポイントの数、および過負荷状態を考慮した 10% の安全マージンを加算した値です。

b. ご使用のデータベース・システムで必要な数の並列 JDBC 接続をサポートできることを確認します。

c. ご使用のデータベース・システムで推測される数の並列 JDBC 接続をサポートするためのベスト・プラクティスに基づき、適切な設定を計画します。

15. 実動システムでは、次の管理作業について計画します。
    - データベースに標準的な実動データを取り込んだ後で、このデータベースを調整する。
    - 完了したプロセス・インスタンスおよびタスク・インスタンスをデータベースから定期的に削除する。使用可能なツールやスクリプトの概要については、『Business Process Choreographer のクリーンアップ手順』を参照してください。

タスクの結果

これで、Business Process Choreographer のデータベースの計画が完了しました。

レポート・データベースの計画

Business Process Choreographer Explorer レポート作成機能のデータベースを計画します。

このタスクについて

Business Process Choreographer Explorer レポート作成機能では同じデータベースを使用できますが、データベースを別途使用すると、パフォーマンスが向上します。BPEDB データベースを再利用しない場合は、以下を実行します。

手順

1. 複数の Event Collector インスタンスを使用する予定であり、これらのインスタンスが同一データベースを使用する場合は、Event Collector ごとに固有のスキーマ名を使用するよう計画します。パフォーマンスを改善するには、Event Collector ごとに個別のデータベースを使用することを計画します。
2. データベースに使用するデータベース・システムを決定します。

   注: Business Process Choreographer Explorer レポート作成機能では、Derby または DB2 のいずれかのデータベースを必要とします。 Derby 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。
3. データベースをホストするサーバーを決定します。
4. データベースのユーザー ID についてまだ計画していない場合は、 156 ページの表 19の内容を実行します。
5. レポート・データベースに Derby データベースを**使用しない**場合は、SQL ベースのユーザー定義関数 (UDF) を使用するか Java ベースの UDF を使用するかを決めます。
    - Java UDF の方が高精度ですが、データベースに JAR ファイルがインストールされていなければ使用できません。

- DB2 for z/OS データベースを使用する予定であり、SQL ベースの UDF ではなく Java ベースの UDF を使用してデータベースを作成したい場合は、メニュー方式の管理ツール setupEventCollector を使用する以外に方法はありません。
- Derby データベースを使用する場合、組み込み Derby データベースでは SQL UDF がサポートされていないため、Java ベースの UDF が使用されます。

UDF の詳細については、255 ページの『Business Process Choreographer Explorer レポート作成機能のユーザー定義関数』を参照してください。

6. Business Process Choreographer Explorer レポート作成機能と Event Collector が BPEDB データベースを使用するように構成する際に `bpeconfig.jacl` スクリプトを使用しない場合は、レポート・データベースの作成方法を決定します。

メニュー方式の管理ツール setupEventCollector の使用

このツールでは、データベースを対話モードで作成し、実行時環境に突き合わせて入力を検証できます。このツールを使用する場合は、このツールで SQL ファイルを作成するがこの SQL ファイルを実行しないようにするかどうかを決定します。実行前に SQL をカスタマイズする場合、またはデータベース管理者に SQL のカスタマイズと実行を依頼する場合は、このオプション (SQL ファイルを作成するがこの SQL ファイルを実行しない) を使用します。このツールの詳細については、272 ページの『setupEventCollector ツール』を参照してください。

このツールでは、データベースを作成する他の方法とは異なり、Java ベースのユーザー定義関数 (UDF) または SQL ベースの UDF を作成することができます。また、この 2 つのオプションの切り替えや、UDF をサポートするために必要な JAR ファイルのインストールと削除を行うこともできます。DB2 for z/OS データベースの場合、このツールでは、Java ベース UDF または SQL ベース UDF のいずれかを使用してデータベースを作成できます。 Derby データベースの場合、データベースの作成には Java ベース UDF のみが使用されます。

SQL スクリプトの実行

このツールを使用してデータベースへアクセスする操作が許可されていない場合は、SQL スクリプトを使用する必要があります。バッチ・モードで bpeconfig.jacl スクリプトを使用するか、または管理コンソールを使用して Business Process Choreographer を構成した場合、SQL スクリプトが生成され、必要なパラメーターがすべて置換されています。そうでない場合は、データベース設計ツールを使用して、SQL スクリプトを対話式で生成できます。

DB2 for z/OS データベースの場合、スクリプトでは SQL ベースの UDF が使用されるため、Java に対して有効なワークロード・マネージャー (WLM) 環境をセットアップする必要はありません。 Derby データベースの場合、データベースの作成には Java ベース UDF のみが使用されます。

初回使用時にテーブルを自動的に作成する

デフォルトのデータベース・スキーマを簡単に作成する方法として、管理コンソールの Business Process Choreographer Event Collector の構成

ページで「**テーブルの作成**」オプションを選択する方法があります。このオプションは、ハイパフォーマンス・システムには適していません。このオプションは、DB2 for z/OS データベースには使用できません。Derby データベースの場合、データベースの作成には Java ベース UDF のみが使用されます。

注: Derby ネットワーク・サーバーのデータ・ソースを使用する場合は、Derby ネットワーク・サーバーをディレクトリー *install_root*/derby/bin/networkServer から始動する必要があります。そのようにしないと、「CWWBO4013E: bpcodbutil.jar ファイルが Derby ネットワーク・サーバーで見つかりませんでした。」というエラーが出てテーブル作成に失敗します。

7. DB2 for z/OS データベースを使用する場合は、以下を計画します。
   - サブシステムのロケーション名 (ネットワーク名)。
   - ストレージ・グループ名。
   - サブシステムが認識するデータベース名。デフォルト値は OBSRVRDB です。
   - データベースへの接続に使用するユーザー ID。このユーザー ID のパスワードも必要です。
   - データベース・オブジェクトが作成されるデータベース・スキーマの名前 (SQLID)。
   - テーブル・スペースが作成されるストレージ・グループを計画します。
     - OBSVR01、OBSVR02、OBSVR03、OBSVR04、OBSVR05、OBSVR06、OBSVR07、および OBSVR08 の場合は通常のテーブル・スペース
     - OS26201、OS26202、OS26203、および OS26204 の場合は LOB テーブル・スペース。
   - デフォルトの SQL ユーザー定義関数 (UDF) ではなく、Java ベースのユーザー定義関数を使用する場合は、関数を実行する WLM 環境の名前を決定します。
   - setupEventCollector ツールを使用してデータベースをセットアップする場合は、以下についても計画します。
     - 使用する JDBC ドライバーのタイプを決定します。
       - タイプ 4 (JDBC を介して直接接続)。この場合は、以下の点も確認してください。
         - データベース・サーバーのホスト名または IP アドレス。デフォルトは localhost です。
         - データベースに使用するポート番号。デフォルトは 446 です。
         - JDBC ドライバー JAR ファイル db2jcc.jar と db2jcc_license_cisuz.jar のディレクトリー。
       - タイプ 2 (ネイティブ・データベース・クライアントを使用して接続)。この場合は、ローカル・カタログに含めるデータベース別名も計画します。
   - 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。

8. Derby データベースを使用する場合は、以下を計画します。
   - データベース名。これは、サーバーのファイル・システムの完全修飾パスでなければなりません。デフォルト値は *install_root*/databases/BPEDB です。
   - データベース・オブジェクトが作成されるデータベース・スキーマの名前。デフォルト値は APP です。
   - setupEventCollector ツールを使用してデータベースをセットアップする場合は、以下についても計画します。
     - Derby Network JDBC ドライバーを使用する場合は、データベースへの接続に使用するユーザー ID を計画します。このユーザー ID のパスワードも必要です。
     - 使用する JDBC ドライバーのタイプを決定します。
       - 組み込み JDBC ドライバーまたは組み込み 40 JDBC ドライバー。この場合、JDBC ドライバー JAR ファイル derby.jar のディレクトリーも計画します。デフォルト・ロケーションは *install_root*/derby/lib です。
       - ネットワーク JDBC ドライバーまたはネットワーク 40 JDBC ドライバー。この場合は、以下の点も確認してください。
         - JDBC ドライバー JAR ファイル derbyclient.jar のディレクトリー。デフォルト・ロケーションは *install_root*/derby/lib です。
         - Derby Network サーバーを使用する場合は、Derby Network サーバー上の UDF JAR ファイル bpcodbutil.jar の位置を決定します。デフォルト・ロケーションは *install_root*/derby/lib です。
         - データベース・サーバーのホスト名。デフォルトは localhost です。
         - データベースに使用するポート番号。デフォルトは 1527 です。
9. バッチ・モードで -createEventCollector yes オプションを指定して **bpeconfig.jacl** ツールを使用する場合は、次のいずれかを計画します。
   - -createDB yes オプションを指定すると、bpeconfig.jacl により生成された SQL スクリプトが実行されます。-dbSchema パラメーターを使用して、BPEDB データベースのスキーマ修飾子を指定できます。-reportSchemaName パラメーターと -reportDataSource パラメーターを使用すれば、Business Process Choreographer Explorer レポート作成機能で BPEDB データベースではなく別のデータベースを使用できます。
   - -createDB no オプションを指定すると、ツールで生成される SQL スクリプトが実行されません。生成される SQL ファイルは、ご使用のデータベースに対応した標準 SQL ファイルに基づいていますが、bpeconfig.jacl ツールに対して指定される構成パラメーターがすべてこの SQL ファイルに事前に入力されるため、必要となるカスタマイズ作業が最小限に抑えられます。生成された SQL スクリプト (データベース・オブジェクト作成スクリプト) を実行する前に、ユーザー自身またはデータベース管理者がこのスクリプトをカスタマイズするように計画します。このツールとその他のデータベース・パラメーターの詳細については、195 ページの『bpeconfig.jacl スクリプトを使用した Business Process Choreographer の構成』を参照してください。

10. 管理コンソールの「**Business Process Choreographer Event Collector**」ページでデータベース表を作成する場合は、次のいずれかを計画します。
    - Derby データベースの場合は、「テーブルの作成」オプションを使用して、指定されているデータベースに Business Process Choreographer が初めてアクセスするときにデフォルト・スキーマをデータベース内に作成するように設定できます。
    - SQL スクリプトを実行してデータベース表を作成する場合は、「テーブルの作成」オプションを使用しないでください。ユーザー自身またはデータベース管理者がデータベース・オブジェクト作成 SQL スクリプトのコピーをカスタマイズした後で、このスクリプトを実行するように計画します。このオプションは、実動システムに最も適しています。
11. 実施するカスタマイズについて計画できるようにするためにデータベース用 SQL スクリプトのプレビューを表示するには、以下のようにします。 データベースの `createSchema_Observer.sql` ファイルを見つけて表示します。ただし、このファイルは変更しないでください。SQL ファイルは次の場所にあります。

    install_root`/dbscripts/ProcessChoreographer/`*database_type*

    ここで、*database_type* は以下のいずれかになります。
    - DB2zOSV8
    - DB2zOSV9
    - Derby

    注: `bpeconfig.jacl` ツールを使用して Business Process Choreographer を設定する場合、このツールにより生成される SQL スクリプトを使用することを計画します。このスクリプトは、構成パラメーターのプレースホルダーを値に置き換えるために編集する必要はありません。生成されるスクリプトは、ツールの実行後にのみ使用できますが、上記に示されている場所にあるスクリプトに基づいてます。テーブル・スペース割り振りをカスタマイズする場合は、生成されたスクリプト・ファイルを編集する必要があります。あるいは、データベース設計ツールを使用して、SQL スクリプトを生成できます。

タスクの結果

レポート・データベースの計画が完了しました。

メッセージング・エンジン・データベースの計画

データベース・ロギングがボトルネックになる可能性がある高負荷設定の場合、Business Process Choreographer バスのメッセージング・エンジンに別のデータベースを使用することで、パフォーマンスを改善できます。

このタスクについて

Service Component Architecture (SCA) システム・バスの各メッセージング・エンジン、SCA アプリケーション・バスの各メッセージング・エンジン、Common Event Infrastructure バスの各メッセージング・エンジン、および Business Process Choreographer バスの各メッセージング・エンジンに対して同じメッセージング・データベースを使用できます。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。パフォーマンスの重要性が高い場合は、SCA バスおよびアプリケーションに使用されるデフォルト MEDB を使用する代わりに、Business Process Choreographer メッセージング・エンジン専用のデータベースを使用するように計画します。

手順

1. **インストーラー**または**プロファイル管理ツール**を使用していずれかのサンプル Business Process Choreographer 構成を取得する場合は、Business Process Choreographer メッセージング・エンジンに Derby Embedded、ファイル・ストア、または WPRCSDB データベースを使用するかどうかを決定します。
2. Java Database Connectivity (JDBC) プロバイダー。Network Deployment 環境ではファイル・ストア・データベースと組み込み Derby データベースは使用できない点に注意してください。
3. WebSphere MQ を使用するには、bpeconfig.jacl 構成スクリプトを使用して Business Process Choreographer を構成する必要があります。 WebSphere MQ を使用すべきではありません。
4. bpeconfig.jacl 構成スクリプトを使用して Business Process Choreographer を構成する場合、Business Process Choreographer は SCA によって使用されるものと同じ種類のメッセージ・ストアを使用します。
   - SCA で FILESTORE が使用される場合、Business Process Choreographer でも FILESTORE が使用されます。
   - SCA で Derby Embedded データベースが使用される場合、Business Process Choreographer ではそれ独自の Derby Embedded データベースが使用されます。
   - SCA でそれら以外のデータベースが使用される場合、Business Process Choreographer では同じデータベース内でそれ独自のスキーマを使用します。
5. 管理コンソールの Business Process Choreographer 構成ページを使用するとき、SCA メッセージ・ストア設定に基づくデフォルトの構成を使用する必要がある場合は「**デフォルト構成の使用**」チェック・ボックスを選択し、それ以外の場合は以下の構成パラメーターを計画します。
   - ローカル・バス・メンバーまたはリモート・バス・メンバーの場所。
   - データベースの名前。デフォルトは `BPEME` です。
   - スキーマ名。デフォルトは `MEDBPM00` です。
6. ファイル・ストアまたは組み込み Derby JDBC プロバイダーを使用している場合は、メッセージ・ストアが自動的に作成されます。
7. ファイル・ストアまたは組み込み Derby JDBC プロバイダーを使用していない場合は、次の構成パラメーターを計画します。
   a. Business Process Choreographer の開始前にデータベースを作成しておくように計画します。
   b. データベース・サーバーのホスト名または IP アドレス、およびこのデータベース・サーバーが使用するポート番号。
   c. データベースへの接続とスキーマの作成に使用するユーザー名。これは、157 ページの表 20 で計画したユーザー ID です。

タスクの結果

これで、Business Process Choreographer メッセージング・エンジンのデータベースの計画が完了しました。

Business Flow Manager および Human Task Manager の計画

Business Flow Manager と Human Task Manager は、Business Process Choreographer 構成の中核を形成します。これらの構成パラメーターを計画する必要があります。

手順

1. Business Flow Manager メッセージ駆動型 Bean で run-as ユーザー ID として使用される、Java Message Service (JMS) プロバイダーのユーザー ID を確認してください。 管理コンソールおよび 158 ページの表 21 では、「**JMS API 認証ユーザー**」とされています。
2. Human Task Manager メッセージ駆動型 Bean で run-as ユーザー ID として使用される Java Message Service (JMS) プロバイダーのユーザー ID を確認してください。 管理コンソールおよび 158 ページの表 21 では、「**エスカレーション・ユーザーの認証ユーザー**」とされています。
3. 管理者とモニターのセキュリティー・ロールがマップされるグループまたはユーザー ID が判明していることを確認します。 詳しくは、 159 ページの表 22 を参照してください。
4. Human Task Manager からエスカレーション・イベントの通知を E メールで送信する場合は、Simple Mail Transfer Protocol (SMTP) E メール・サーバーが設置されている場所のホスト名または IP アドレスを確認します。 通知 E メールの送信者アドレスを計画します。E メール・サービスで認証を必要とする場合は、サービスに接続するために使用するユーザー ID とパスワードを確認してください。
5. API の Web サービス・バインディングのコンテキスト・ルートを決定します。
   - サーバーでの構成時:
     - Business Flow Manager のデフォルトは `/BFMIF_nodeName_serverName` です。
     - Human Task Manager のデフォルトは `/HTMIF_nodeName_serverName` です。
   - クラスターでの構成時:
     - Business Flow Manager のデフォルトは `/BFMIF_clusterName` です。
     - Human Task Manager のデフォルトは `/HTMIF_clusterName` です。
6. Business Process Choreographer Explorer、Business Space、あるいは Representational State Transfer (REST) API または JAX Web サービス API を使用するクライアントを使用する場合は、REST API および JAX Web サービス API のコンテキスト・ルートを決定してください。
   - Business Flow Manager のデフォルトは `/rest/bpm/bfm` および `/BFMJAXWSAPI` です。
   - Human Task Manager のデフォルトは `/rest/bpm/htm` および `/HTMJAXWSAPI` です。
   - サーバーまたは単一クラスター、あるいは異なる Web サーバーにマップされた複数のクラスターに構成されている場合は、デフォルト値を使用できます。

- 同じ Web サーバーにマップされた、Network Deployment 環境内の複数のデプロイメント・ターゲットに構成されている場合は、デフォルト値を使用できません。各 Business Process Choreographer 構成のコンテキスト・ルートは、ホスト名とポートの組み合わせごとに固有のものでなければなりません。これらの値は、Business Process Choreographer の構成後に、管理コンソールを使用して手動で設定する必要があります。

7. 初期設定で Business Flow Manager または Human Task Manager (あるいはこの両方) の監査ロギングを使用可能にするかどうかを決定します。
8. Business Process Choreographer Explorer レポート作成機能を使用する場合は、Common Event Infrastructure ロギング・イベントを生成するように Business Flow Manager を初期設定するかどうかを決定します。

タスクの結果

これで、Business Flow Manager および Human Task Manager のすべての初期構成パラメーターの計画が完了しました。これらの設定は、管理コンソールを使用して後でいつでも変更することができます。

担当者ディレクトリー・プロバイダーの計画

Business Process Choreographer の担当者ディレクトリー・プロバイダー、担当者の代替、virtual member manager、および Lightweight Directory Access Protocol (LDAP) の設定を計画します。

手順

1. ヒューマン・タスクを使用する場合は、使用する担当者ディレクトリー・プロバイダーを決定します。

   Virtual Member Manager (VMM) 担当者ディレクトリー・プロバイダー
   VMM 担当者ディレクトリー・プロバイダーは、すぐに利用できるフェデレーテッド・リポジトリー (Virtual Member Manager とも呼ばれる) であり、ファイル・リポジトリーを使用して WebSphere セキュリティー向けに事前構成されています。フェデレーテッド・リポジトリーと別のユーザー・リポジトリーを組み合わせて使用する場合は、フェデレーテッド・リポジトリーを再構成する必要があります。 VMM 担当者ディレクトリー・プロバイダーでは、代替を含む Business Process Choreographer の担当者割り当て機能がすべてサポートされています。これは、フェデレーテッド・リポジトリーの機能 (LDAP、データベース、ファイル・ベース、およびプロパティー拡張リポジトリーなど各種リポジトリー・タイプのサポートなど) を使用します。

   VMM 担当者ディレクトリー・プロバイダーを使用するには、WebSphere Application Server セキュリティーに対応してフェデレーテッド・リポジトリーを構成している必要があります。フェデレーテッド・リポジトリーは、ファイル、LDAP、またはデータベースに基づいて 1 つ以上のユーザー・リポジトリーに関連付けることができます。詳しくは、『フェデレーテッド・リポジトリー構成におけるレルムの管理』を参照してください。フェデレーテッド・リポジトリーの使用について詳しくは、『IBM® WebSphere Developer Technical Journal』を参照してください。

Lightweight Directory Access Protocol (LDAP) 担当者ディレクトリー・プロバイダー この担当者ディレクトリー・プロバイダーは、使用前に構成しておく必要があります。ステップ 2 で計画を行います。

システム担当者ディレクトリー・プロバイダー
この担当者ディレクトリー・プロバイダーは、構成せずに使用できます。実動システムではこのプロバイダーを使用しないでください。これは、アプリケーション開発テストでのみ使用するプロバイダーとして提供されています。

ユーザー・レジストリーの担当者ディレクトリー・プロバイダー
この担当者ディレクトリー・プロバイダーは、構成せずに使用できます。WebSphere セキュリティー・レルム定義に応じて、ユーザー・レジストリーは以下のリポジトリーのいずれかを使用します。

- フェデレーテッド・リポジトリー - 以下を使用できます。
  - ファイル・レジストリー
  - 1 つ以上の LDAP
  - 1 つ以上のデータベース
- スタンドアロン LDAP
- スタンドアロン・カスタム
- ローカル・オペレーティング・システム

2. Lightweight Directory Access Protocol (LDAP) を使用する場合、以下を計画します。
   a. 場合によっては、使用する `LDAPTransformation.xsl` ファイルをカスタマイズする必要があります。このファイルの場所と、カスタマイズする必要があるプロパティーのリストについては、225 ページの『LDAP 担当者ディレクトリー・プロバイダーの構成』を参照してください。
   b. 以下の LDAP カスタム・プロパティーを計画します。

LDAP プラグイン・プロパティー	必須またはオプション	説明
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**」 → 「**セキュア管理、アプリケーション、およびインフラストラクチャー**」 → 「**Java 認証・承認サービス**」 → 「**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。別名が設定されていない場合、または AuthenticationType が `simple` に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	このプロパティーを `simple` (単純認証) に設定する場合は、AuthenticationAlias パラメーターを指定する必要があります。このパラメーターを設定しない場合は、匿名認証が使用されます。
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)。ディレクトリー・ルートを指定するには、単一引用符を 2 個使用して空ストリング `''` を指定します。
Casesentiveness ForObjectclasses	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。

LDAP プラグイン・プロパティー	必須またはオプション	説明
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。SSL 接続の場合は LDAP の URL を使用します。 ミラーリングされたデータを 2 台以上の LDAP サーバーで保持する高可用性構成の場合、それぞれの LDAP サーバーの URL を指定するように計画し、個々の URL はスペース文字で分離してください。
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。
additionalParameter Name1-5 および additionalParameter Value1-5	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。

3. virtual member manager を使用する場合は、以下を計画します。
   a. 場合によっては、使用する VMMTransformation.xsl ファイルをカスタマイズする必要があります。 このファイルの場所と、カスタマイズする必要があるプロパティーのリストについては、 224 ページの『Virtual Member Manager 担当者ディレクトリー・プロバイダーの構成』を参照してください。
4. 担当者の代替を使用する場合は、以下の点を考慮に入れてください。
   - VMM 担当者ディレクトリー・プロバイダーを使用する必要があります。LDAP、システム、およびユーザー・レジストリーの担当者ディレクトリー・プロバイダーは、担当者の代替をサポートしていません。
   - 実稼働環境で担当者の代替を使用する場合、代替情報の格納先として VMM プロパティー拡張リポジトリーを使用するように計画します。プロパティー拡張リポジトリーは、セル全体で固有であり、セル内からアクセス可能である必要があります。暗黙的には、選択されたデータベースも同様です。BPEDB データベースは必ずしもセル内で固有ではないため、BPEDB は使用できません。共通データベース WPSRCDB を使用してプロパティー拡張リポジトリーをホストできます。ただし実稼働環境では、他の WebSphere Process Server データベースから独立したデータベースを使用することをお勧めします。
   - 単一サーバーのテスト環境で担当者の代替を使用する場合は、フェデレーテッド・リポジトリー用に構成した内部ファイル・レジストリーに担当者代替情報を格納できます。

タスクの結果

これで、担当者ディレクトリー・プロバイダーと担当者割り当てオプションの計画が完了しました。

Business Process Choreographer Explorer の計画

Business Process Choreographer Explorer の構成オプションおよび構成パラメーターを計画します。

このタスクについて

Business Process Choreographer Explorer を使用する場合は、その構成を Business Process Choreographer の構成と同時に行うことも、後で行うこともできます。Business Process Choreographer Explorer レポート作成機能はオプションです。

手順

1. 構成が必要な Business Process Choreographer Explorer の数を決定します。最初のインスタンスは Business Process Choreographer の構成中に簡単に作成できます。 以下の理由および考慮事項が考えられます。
   - Business Process Choreographer Explorer の各インスタンスが接続できる Business Process Choreographer 構成は 1 つだけであるため、ご使用の環境に複数の Business Process Choreographer 構成がある場合、それぞれの構成に対して Business Process Choreographer Explorer インスタンスをセットアップすることが妥当です。
   - 異なる複数のカスタマイズ・バージョンの Business Process Choreographer Explorer を同じ Business Process Choreographer 構成に接続することが必要な場合もあります。それぞれのバージョンを独立してカスタマイズすることができます。カスタマイズできる内容については、368 ページの『Business Process Choreographer Explorer のカスタマイズ』を参照してください。
   - 各サーバーまたはクラスター上に複数の Business Process Choreographer Explorer インスタンスを構成できます。
   - Business Process Choreographer または Business Process Choreographer Event Collector の構成が存在する場所に関係なく、インスタンスはどのデプロイメント・ターゲットにも作成できます。
   - それぞれの Business Process Choreographer Explorer インスタンスのレポート作成機能が接続できる Business Process Choreographer Event Collector は 1 つだけであるため、レポート作成機能を持つ Business Process Choreographer Explorer インスタンスの数が、Business Process Choreographer イベント・コレクターと同じになるように構成することを計画します。
2. 必要な Business Process Choreographer Explorer インスタンスごとに、以下を計画します。
   a. Business Process Choreographer Explorer のコンテキスト・ルート。これはセル内で固有でなければなりません。デフォルトは `/bpc` です。
   b. エスカレーション E メールに挿入される Business Process Choreographer Explorer の URL。
   c. Business Flow Manager と Human Task Manager の Representational State Transfer (REST) API エンドポイントの URL。REST API で計画したコンテキスト・ルートの値と一致する必要があります。例えば、 Human Task Manager Web サービスのコンテキスト・ルートが `/rest/bpm/htm` である場合、 Human Task Manager REST API エンドポイントのエンドポイント URL は、`http://`*`hostname`*`:`*`port`*`/rest/bpm/htm` になります。
   d. 照会に対して返される結果の最大数。デフォルトは 10000 です。
   e. この Business Process Choreographer Explorer が管理する Business Process Choreographer インスタンスのデプロイメント・ターゲット (サーバーまたはクラスター)。

f. Business Process Choreographer Explorer レポート作成機能を使用する場合は、『Business Process Choreographer Explorer レポート作成機能の計画』を実行します。計画と構成は後で行うこともできます。

タスクの結果

これで、Business Process Choreographer Explorer の構成オプションの計画が完了しました。

Business Process Choreographer Explorer レポート作成機能の計画

Business Process Choreographer Explorer レポート作成機能と Event Collector の構成を計画します。

このタスクについて

Business Process Choreographer Explorer レポート作成機能を使用する場合は、その構成を Business Process Choreographer Explorer の構成時に行うことも、後で行うこともできます。

手順

1. セキュリティー・ロールは、Business Process Choreographer Explorer レポート作成機能へのアクセスを制限するために使用されないため、Business Process Choreographer Explorer のすべてのユーザーがレポート作成機能にアクセスすることを望まない場合は、レポート作成機能用に別の Business Process Choreographer Explorer インスタンスを構成して、通常のユーザーからアクセス不能にするように計画します。
2. Business Process Choreographer Explorer レポート作成機能の各種トポロジー・エレメントの目的とエレメント間の関係を理解します。

   Business Process Choreographer Explorer レポート作成機能
   バージョン 6.2 より前は、この機能は Business Process Choreographer Observer として利用可能でした。バージョン 6.2 以降、この機能は Business Process Choreographer Explorer に統合され、「**レポート**」タブで使用できるようになりました。この機能を使用するには、まず Business Process Choreographer Explorer レポート作成機能を構成する必要があります。

   Event Collector アプリケーション
   このアプリケーションは、Common Event Infrastructure (CEI) サーバーが構成されているクラスターまたはサーバー上にデプロイする必要があります。各 CEI デプロイメント・ターゲットについて、Event Collector を 1 つだけ持つことができます。Business Process Choreographer が構成されているターゲット上にデプロイする必要はありません。CEI からビジネス・プロセス・イベントを受信して変換し、変換したイベントをレポート・データベースに書き込みます。

   レポート・データベース
   Event Collector と Business Process Choreographer Explorer レポート作成機能は、同じデータベースを使用して通信します。非実動システムの場合、他のコンポーネントとデータベースを共有できます。

選択肢は、セットアップされている Business Process Choreographer のトポロジーから独立しています。可能な構成について詳しくは、 186 ページの『Business Process Choreographer Explorer レポート作成機能の概要』を参照してください。

3. セットアップの目的、システムの要件、およびトポロジーの影響を確認します。

   単純なセットアップ
   構成と管理が単純であるが、パフォーマンスが低いセットアップでは、Business Process Choreographer Explorer および CEI が構成されているデプロイメント・ターゲットに Event Collector アプリケーションをデプロイし、ローカル・データベース・システムを使用します。

   負荷が高い実動システム: Network Deployment
   複数ノードで構成されるセルと複数のクラスターを使用します。Business Process Choreographer Explorer のインスタンスを、セル内の任意のデプロイメント・ターゲットにインストールします。Event Collector アプリケーションを、Common Event Infrastructure (CEI) を構成したクラスターにインストールします。個別のデータベース・サーバーを使用します。

4. Business Process Choreographer Explorer レポート作成機能のデータベースについてまだ計画していない場合は、 168 ページの『レポート・データベースの計画』を行います。

5. 構成する Event Collector インスタンスごとに、以下を計画します。

   a. インスタンスをインストールするかどうかを決定します。デプロイメント・ターゲットごとに 1 つの Event Collector インスタンスのみをインストールできます。また、デプロイメント・ターゲットでは CEI が構成されている必要があります。

   b. この Event Collector インスタンスの構成方法を決定します。

      - 管理コンソールのページを使用する。このオプションについて詳しくは、262 ページの『管理コンソールを使用した Business Process Choreographer Event Collector の構成』を参照してください。
      - 対話式 setupEventCollector ツールを使用する。このオプションについて詳しくは、 260 ページの『setupEventCollector ツールを使用した Business Process Choreographer Event Collector の構成』を参照してください。
      - Business Process Choreographer 構成の作成時に、`bpeconfig.jacl` スクリプトを使用する。`-createEventCollector` オプションにはデフォルト値 `yes` が設定されています。

        注: ハイパフォーマンス・システムについては、`bpeconfig.jacl` を使用して Business Process Choreographer Explorer レポート作成機能 を構成しないでください。この理由は、`bpeconfig.jacl` は Event Collector および Business Process Choreographer Explorer レポート作成機能 アプリケーションを、Business Process Choreographer 構成と同じデプロイメント・ターゲットに構成するためです。このオプションについて詳しくは、 195 ページの『bpeconfig.jacl スクリプトを使用した Business Process Choreographer の構成』を参照してください。
        Event Collector を対話モードで構成する場合は、`bpeconfig.jacl` は使用できません。

   c. データ・ソースを計画します。

- Business Process Choreographer Explorer レポート作成機能が Business Process Choreographer と同じ物理データベースを共有している場合は、レポート・データベースに別のデータ・ソースを使用することを計画し、その JNDI 名を計画します。
- データベースに使用する認証別名を計画します。
- セルの有効範囲を使用してデータ・ソースを作成するように計画します。

d. Event Collector の構成時に必要な構成パラメーターを計画します。
- レポート・データベースの JNDI データ・ソース名。
- データベース・オブジェクトに使用するスキーマ。デフォルトは、データベースへの接続に使用するユーザー ID です。
- データベースへの接続に使用するユーザー ID。デフォルトはデータベースによって異なります。 DB2 の場合、デフォルトは db2admin、Oracle の場合、デフォルトは system、その他のデータベースの場合、デフォルトはログオン・ユーザーのユーザー ID です。
- ユーザー ID のパスワード。
- タイプ 4 JDBC 接続を使用する場合は、データベース・サーバーのホスト名または IP アドレスと、データベース・サーバーが使用するポート番号も収集します。
- Event Collector をデプロイするかどうかを決定します。デプロイメント・ターゲットでは CEI が構成されている必要があります。したがって、CEI のための別のクラスターがある場合は、Event Collector をそのクラスターにデプロイするよう計画してください。
- Network Deployment 環境に Event Collector をデプロイする場合は、CEI バスのメッセージング・エンジンが構成されているデプロイメント・ターゲットを確認しておいてください。
- CEI バスのセキュリティーが有効な場合、CEI バスでの認証に使用する JMS ユーザー ID を計画します。
- Event Collector の構成時に CEI イベントによるビジネス・イベントのログ記録を有効にするか、または後で管理コンソールを使用するかスクリプトを実行してログ記録を有効にするかを決定します。

e. ランタイム構成値を計画します。場合によっては、Event Collector の構成後にニーズに合わせてランタイム構成値をカスタマイズする必要があります。
- BpcEventTransformerEventCount
- BpcEventTransformerMaxWaitTime
- BpcEventTransformerToleranceTime
- ObserverCreateTables
- 認証別名ユーザー ID がデータベース・スキーマを所有していない場合は、ObserverSchemaName を計画します。

これらの値について詳しくは、266 ページの『Business Process Choreographer Explorer レポート作成機能の構成パラメーターの変更』を参照してください。

6. 構成する Business Process Choreographer Explorer レポート作成機能ごとに、以下を計画します。

- このインスタンスの構成方法を決定します。
  - Business Process Choreographer Explorer の作成時に、 Business Process Choreographer Explorer の管理コンソール・ページを使用する。このオプションについて詳しくは、 265 ページの『管理コンソールを使用した Business Process Choreographer Explorer レポート作成機能の構成』を参照してください。
  - Business Process Choreographer Explorer の作成時に、`clientconfig.jacl` スクリプトを使用する。
  - Business Process Choreographer 構成の作成時に、`bpeconfig.jacl` スクリプトを使用する。

    注: ハイパフォーマンス・システムについては、`bpeconfig.jacl` を使用して Business Process Choreographer Explorer レポート作成機能 を構成しないでください。この理由は、`bpeconfig.jacl` は Event Collector および Business Process Choreographer Explorer レポート作成機能 アプリケーションを、Business Process Choreographer 構成と同じデプロイメント・ターゲットに構成するためです。このオプションについて詳しくは、 195 ページの『bpeconfig.jacl スクリプトを使用した Business Process Choreographer の構成』を参照してください。
- レポート・データベースのスキーマ名。
- レポート・データベースに接続するために Business Process Choreographer Explorer によって使用されるデータ・ソースの JNDI 名。

7. `bpeconfig.jacl` スクリプトを使用して Business Process Choreographer を構成する場合:
   - スクリプトをバッチ・モードで実行する場合、デフォルトでは、 Event Collector アプリケーションと Business Process Choreographer Explorer アプリケーションも、 Business Process Choreographer 構成と同じデプロイメント・ターゲットで構成されます。
   - Event Collector および Business Process Choreographer Explorer レポート作成機能、あるいはそのいずれかを `bpeconfig.jacl` で構成したくない場合は、`bpeconfig.jacl` のオプション `-createEventCollector no` および `-reportFunction no`、あるいはそのいずれかを使用するように計画して、`bpeconfig.jacl` でこれらが構成されないようにします。

タスクの結果

これで、Business Process Choreographer Explorer レポート作成機能と Event Collector の構成オプションの計画が完了しました。

リモート・クライアント・アプリケーションの計画

Business Process Choreographer API を使用し、WebSphere Process Server クライアント・インストールで稼働するリモート Business Process Choreographer クライアント・アプリケーションを計画します。

このタスクについて

アプリケーションで Business Process Choreographer API を使用する場合は、WebSphere Process Server クライアントのインストールを使用して完全な WebSphere Process Server サーバー・インストールに対してリモート側からアプリケーションを実行できます。クライアントは、完全な WebSphere Process Server インストールよりも構成および管理が容易です。

WebSphere Process Server クライアント・インストールには WebSphere Process Server プロファイル・テンプレートは含まれていませんが、基盤となる WebSphere Application Server プロファイル (SDO 2.1.1 付きの Feature Pack for SCA バージョン 1.0 適用) を拡張する必要があります。 つまり、フェデレートされたプロファイルを持つ既存の WebSphere Application Server インストールの上に WebSphere Application Server クライアントをインストールでき、フェデレートされたこれらの WebSphere Application Server プロファイルでただちに WebSphere Process Server クライアント機能を利用できます。完全な WebSphere Process Server サーバーではこのシナリオは実現できません。WebSphere Process Server は、フェデレート済みプロファイルの拡張をサポートしていないためです。

手順

1. WebSphere Process Server クライアントのインストールを計画します。
   - WebSphere Portal Server から Business Process Choreographer にアクセスする場合は、互換性のある WebSphere Process Server クライアントがインストールされていなければなりません。

表 26. WebSphere Portal Server が Business Process Choreographer にアクセスするのに使用できる WebSphere Process Server クライアントのバージョン

	WebSphere Process Server クライアント・バージョン			
WebSphere Portal Server バージョン	**6.1.0.1**	**6.1.2**	**6.2**	**7.0**
6.1.0.1	はい	はい	いいえ	いいえ
6.1.0.2	はい	はい	はい	いいえ

   - クライアント・インストールは基本プロファイルを拡張しないため、フェデレート済みプロファイルを含む既存のプロファイルは、いずれも WebSphere Process Server クライアントをただちに使用できます。
   - 既存の WebSphere Application Server インストールがない場合は、WebSphere Application Server Network Deployment インストールが作成されます。
2. 以下のどのタイプの Business Process Choreographer クライアント・アプリケーションを使用するかを決定します。
   - カスタム・クライアント・アプリケーション
   - Business Process Choreographer Explorer

     注: カスタマイズした JavaServer Pages (JSP) を使用する場合は、663 ページの『第 15 章 タスクおよびプロセス・メッセージ用の JSP ページの開発』で説明しているように、その場所を調べておきます。

3. Business Process Choreographer を使用するカスタム・クライアント・アプリケーションを開発する場合は、アプリケーションで使用するインターフェースを計画します。 プロセスおよびタスクは、以下のいずれかを使用して処理できます。
   - Web サービス API、Java Messaging Service (JMS) API、または Representational State Transfer (REST) API: これらの API に基づくリモート・クライアント・アプリケーションでは、WebSphere Process Server のインストールは必要ありません。
   - JavaServer Faces (JSF) コンポーネント
   - Enterprise JavaBeans™ (EJB) API

   注: Business Process Choreographer の EJB API を使用するクライアント・アプリケーションを開発する場合は、 548 ページの『セッション Bean のリモート・インターフェースにアクセスする』で説明している方法に従ってパッケージ化する必要があります。
4. WebSphere Process Server クライアントをインストールするセルのタイプを決定するか明確化します。
   a. Business Process Choreographer を構成した管理対象サーバーまたはクラスターが存在するセルでは、リモート成果物ローダー (RAL) のデフォルト構成によってクライアントとサーバーの間で成果物の非セキュア伝送を実行できます。これを「単一セル」のシナリオと呼びます。
   b. Business Process Choreographer が構成されている管理対象サーバーまたはクラスターを持たないセルには、さまざまなデプロイメント・マネージャーが存在します。これを「クロス・セル」のシナリオと呼びます。クライアント・アプリケーションが EJB API を使用する場合は、名前空間のバインディングを定義し、Business Process Choreographer が構成されているサーバーまたはクラスターをクライアント・アプリケーションが検出できるようにする必要があります。

タスクの結果

リモート Business Process Choreographer クライアント・アプリケーションの計画が完了しました。

Business Process Choreographer の概要

Business Flow Manager と Human Task Manager の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。これらの構成体は、サービスのオーケストレーションと、ビジネス・プロセスの担当者が関与するアクティビティーの統合に使用できます。 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、担当者が関与するアクティビティー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、サービス指向アーキテクチャー (SOA) または Service Component Architecture (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 アプリケーションで、ビジネス・プロセスとヒューマン・タスクを管理することができます。また、これにはオプションで Business Process Choreographer Explorer レポート作成機能 (これまでは Business Process Choreographer Observer と呼ばれていました) が組み込まれ、この機能によって実行中のプロセスの状態を監視することができます。
- Business Space の一部としてのヒューマン・ワークフロー・ウィジェット。これらのウィジェットにより、作業の管理、他のユーザーのためのタスクの作成、およびサービスとプロセスの開始が可能です。

関連タスク

Business Process Choreographer の構成計画
Business Process Choreographer のセットアップと構成パラメーターを計画します。

Business Process Choreographer Explorer の概要

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実装する Web アプリケーションです。

また、オプションのレポート作成機能が組み込まれていますが、この機能はこれまでは Business Process Choreographer Observer と呼ばれていました。

1 つ以上の Business Process Choreographer Explorer インスタンスをサーバーまたはクラスター上で構成できます。 WebSphere Process Server プロファイルを持つ WebSphere Process Server のインストール済み環境、または WebSphere Process Server クライアントのインストール済み環境があれば十分です。Business Process Choreographer を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。クライアントを WebSphere Process Server に接続する必要がある唯一のインフラストラクチャーは WebSphere Process Server クライアント・インストールですが、これには Business Process Choreographer Explorer は含まれません。 WebSphere Process Server クライアント・インストールの場合でも、デプロイメント・マネージャーを使用して Business Process Choreographer Explorer をサーバーにインストールしてください。

単一の Business Process Choreographer Explorer のみ 1 つの Business Process Choreographer 構成に接続できます。とはいえ、ローカル構成に接続する必要はありません。ただし、Business Process Choreographer Explorer の複数のインスタンスを同じサーバーまたはクラスター上に構成したり、各インスタンスを別個の Business Process Choreographer 構成に接続したりすることができます。

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、ビジネス・プロセス管理者であれば、配置されたビジネス・プロセス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 レポート作成機能の概要

Business Process Choreographer Explorer レポート作成機能について。

Business Process Choreographer Explorer レポート作成機能を使用すると、完了しているプロセスに関するレポートを作成できます。また、これを使用して、実行中のプロセスの状況を表示することもできます。ここでは、アーキテクチャーと可能な構成パスについて説明します。

Business Process Choreographer Explorer レポート作成機能では、Common Event Infrastructure (CEI) を使用して、WebSphere Process Server から出力されるイベントを収集します。数多くの定義済みレポートを使用するか、または独自のレポートを定義して、多くのプロセス、アクティビティー、または他の集約データの概要を把握することができます。また、特定のプロセスまたはアクティビティーについての情報を得ることも可能です。

Business Process Choreographer Explorer レポート作成機能は、次の図に示す 2 つの Java EE エンタープライズ・アプリケーションが基本になっています。

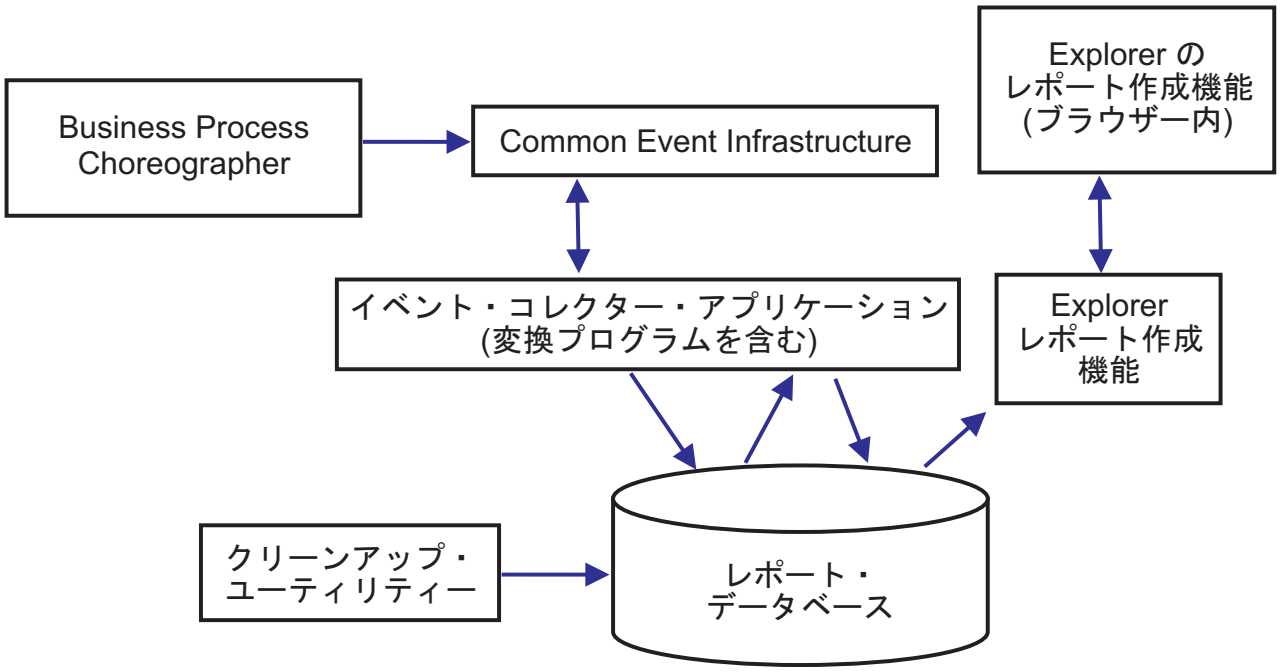


図 1. アーキテクチャー

- Event Collector アプリケーションは、CEI バスからイベント情報を読み取り、その情報をレポート・データベースの Event Collector 表に格納します。
- レポート・データベースは、イベント・データを格納する一連のデータベース表です。
- イベント変換プログラムが定期的に起動し、未加工のイベント・データを、Business Process Choreographer Explorer レポート作成機能からの照会に適した形式に変換します。
- Business Process Choreographer Explorer レポート作成機能は、レポートを生成し、ユーザーがグラフィカル・ユーザー・インターフェース (GUI) を使用して開始できるほかのアクションを実行します。
- GUI を使用して独自のレポートを生成することができます。また、自分で定義したレポートを保管および検索することも可能です。
- クリーンアップ・ユーティリティーは、データベースからレコードを除去するときに使用することができ、これはパフォーマンスの向上に役立ちます。

簡易構成

簡易構成では、パフォーマンスは重要な考慮事項ではありません。簡易構成を次の図に示します。

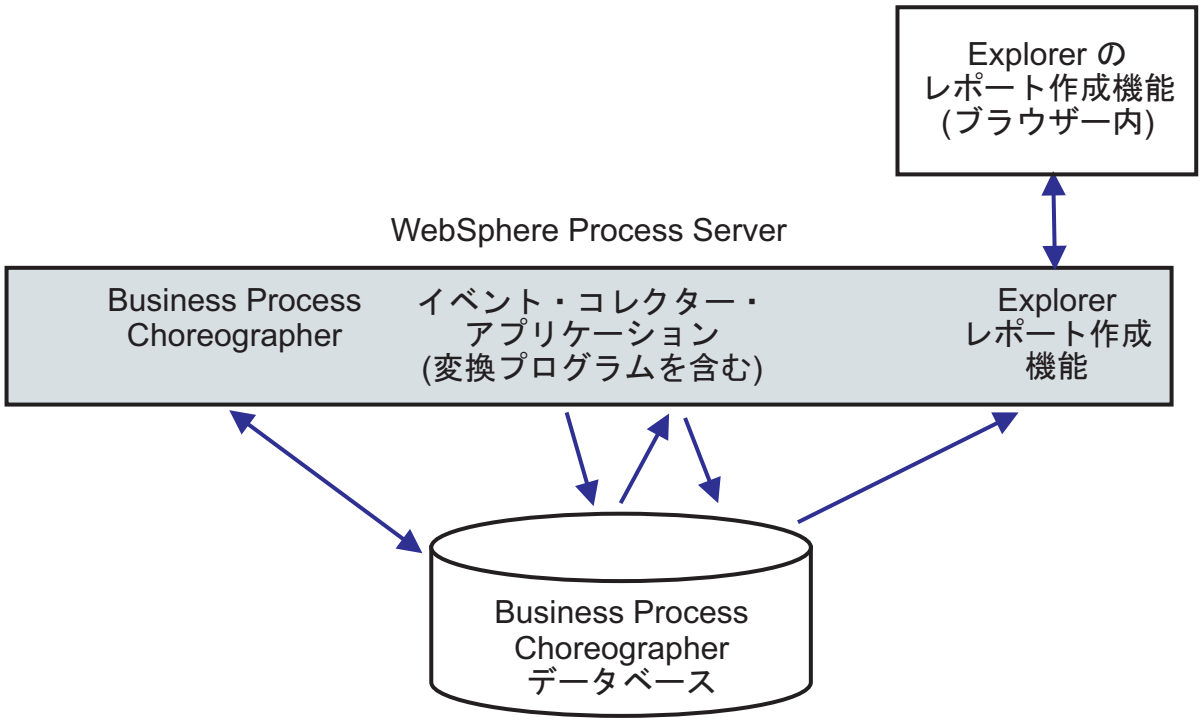


図 2. スタンドアロン・セットアップ

すべてが 1 台のシステムにインストールされており、Business Process Choreographer と Business Process Choreographer Explorer レポート作成機能が同じデータベースを使用します。

サンプル Business Process Choreographer 構成を作成する場合、このような簡易構成が作成されます。また、 bpeconfig.jacl ツールでは、デフォルトで、Business Process Choreographer 構成と同じデプロイメント・ターゲットでこの種のセットアップが構成されます。Common Event Infrastructure (CEI) ロギングが使用可能になり、必要なデータベース・スキーマが、Business Process Choreographer の Derby データベース BPEDB に作成されます。この構成パスは、パフォーマンスが重要な考慮事項ではない場合に最適です。

ハイパフォーマンス構成

Business Process Choreographer Explorer レポート作成機能アーキテクチャーの潜在能力を最大限に活用できるようにするための対話式の構成ツールが用意されています。例えば、パフォーマンスの点で理想的な構成は、Business Process Choreographer 構成、CEI イベント・サーバー、および Business Process Choreographer Explorer (レポート作成機能を持つもの) がそれぞれ別個のシステム稼働し、Business Process Choreographer および Business Process Choreographer Explorer レポート作成機能に専用のデータベースを用意した構成です。

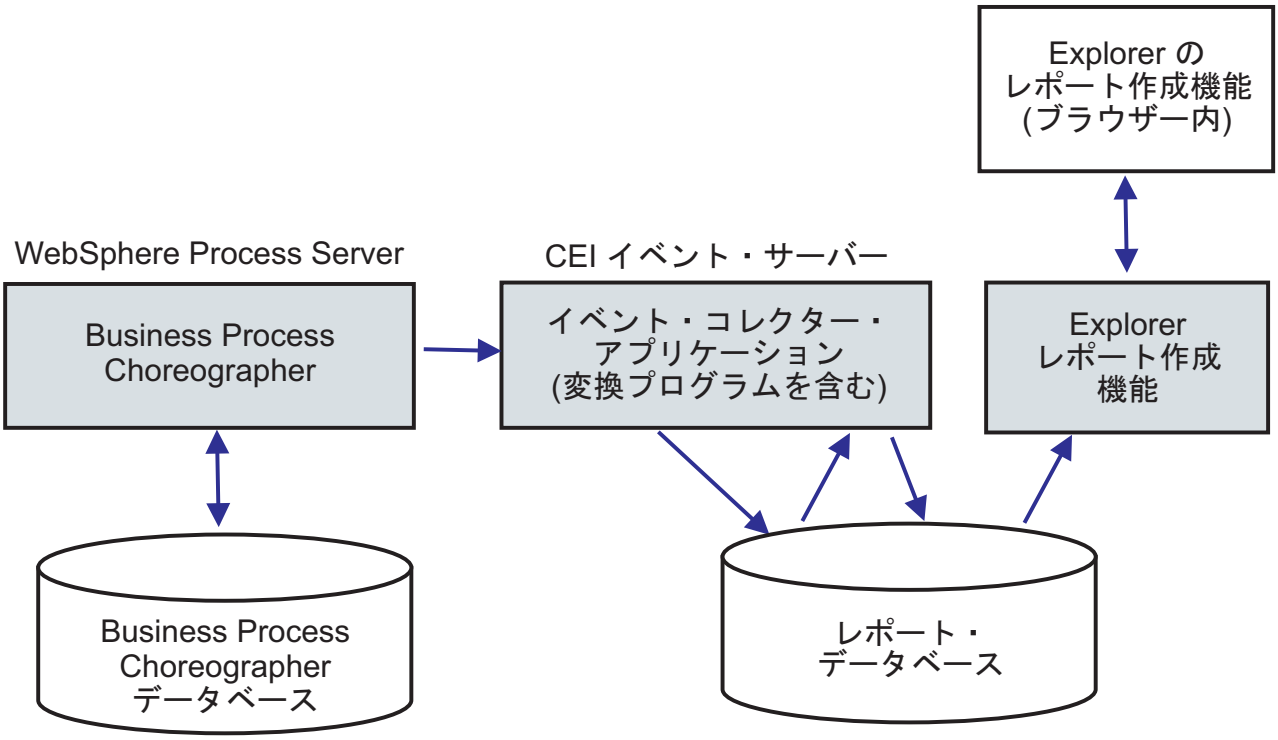


図 3. 実動パフォーマンスを考慮した Business Process Choreographer Explorer レポート作成のセットアップ

Business Process Choreographer Explorer レポート作成機能用に別個のデータベースを使用する場合、クラスター構成のセットアップにおいて Business Process Choreographer Explorer レポート作成機能を既存の Business Process Choreographer 構成に追加する場合、あるいは、より複雑なデータベース・オプションを使用する場合は、 241 ページの『Business Process Choreographer Explorer レポート作成機能および Event Collector の構成』を実行します。

Network Deployment 環境

Business Process Choreographer Explorer レポート作成機能を Network Deployment 環境で構成する場合は、以下の制約事項が適用されます。

- CEI は、使用するセル内で構成する必要があります。
- 前の図に示されるとおり、Business Process Choreographer の Event Collector は、CEI イベント・サーバーが構成されているデプロイメント・ターゲットで構成する必要があります。CEI イベント・サーバーを Business Process Choreographer とは別のクラスターで構成する場合は、CEI イベント・サーバーが構成されているデプロイメント・ターゲットに Business Process Choreographer の Event Collector を構成する必要があります。Business Process Choreographer Explorer レポート作成機能アプリケーションを Event Collector と同じシステム上にインストールする必要はありません。

第 4 章 Business Process Choreographer の構成

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任意のエンタープライズ・アプリケーションをインストールする前に、サーバーまたはクラスター上で Business Process Choreographer を構成する必要があります。

始める前に

137 ページの『第 3 章 Business Process Choreographer の構成計画』が完了していることは重要です。

このタスクについて

選択した構成ツールに応じて、Business Process Choreographer を構成する各サーバーまたはクラスターで以下のいずれかを実行します。

手順

Business Process Choreographer を構成するには、以下の 2 つのツールのいずれかを使用します。

- 管理コンソールの Business Process Choreographer 構成ページ
- bpeconfig.jacl スクリプト

タスクの結果

Business Process Choreographer が構成されます。

次のタスク

- 他のサーバーまたはクラスター上で、Business Process Choreographer 構成をさらに作成するには、このタスクを繰り返します。
- Common Event Infrastructure をまだ構成していない場合は、doc/tcei_configuration.ditaを実行してください。

管理コンソールの「Business Process Choreographer の構成」ページの使用

管理コンソールの「Business Process Choreographer の構成」ページで、特定のサーバーまたはクラスター上に構成を作成する方法を説明します。

始める前に

始める前に、以下のタスクを完了しておく必要があります。

- サーバーまたはクラスターの SCA サポートの構成

このタスクについて

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 Business Process Choreographer アプリケーションをインストールする必要があります。

手順

1. Business Process Choreographer の構成先の各ノードで、JDBC ドライバーの環境変数が設定されていることを確認します。クラスターでは、クラスター・メンバーをホストするすべてのノードに対してこれを実行する必要があります。
   a. 「**環境**」 → 「**WebSphere 変数**」をクリックし、「**有効範囲**」として、Business Process Choreographer が構成されているノードを選択します。
   b. JDBC プロバイダーの環境変数を選択します。
      - DB2 on z/OS で Universal ドライバーを使用している場合は、DB2UNIVERSAL_JDBC_DRIVER_PATH を選択します。
   c. JDBC ドライバーの JAR ファイルの場所を指すように環境変数を設定します。
2. 管理コンソールで、Business Process Choreographer を構成するサーバーまたはクラスターを選択します。 「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」 または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」をクリックします。
3. 「Business Process Choreographer の構成」ページに進みます。 「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Business Process Choreographer Container**」をクリックします。
4. Business Process Choreographer が構成されていないことを確認します。Business Flow Manager が既にインストールされているかどうかを示すメッセージが表示されます。

   Business Flow Manager と Human Task Manager が既にインストールされている場合は、次のステップに進む前に、285 ページの『第 5 章 Business Process Choreographer 構成の除去』を行います。
5. 値を入力し、このサーバーまたはクラスターの Business Process Choreographer 構成で使用するよう計画したオプションを選択します。
6. 「**適用**」をクリックします。 Business Process Choreographer のデプロイと構成の進捗を示す情報が表示されます。
7. インストールが正常に完了した場合、「**変更の保管**」をクリックします。 それ以外の場合は、変更を破棄し、管理コンソールと、デプロイメント・マネージャーまたはサーバーの `SystemOut.log` ファイルで、問題の訂正に役立つエラー・メッセージを確認してから、再試行します。
8. Business Process Choreographer をアクティブにします。 280 ページの『Business Process Choreographer の活動化』を実行します。
9. オプション: Business Process Choreographer 基本構成が機能することを確認します。 281 ページの『Business Process Choreographer の作動確認』を実行します。
10. オプション: Human Task Manager の設定を変更します。

- 送信者アドレスまたは Business Process Choreographer Explorer についての URL 接頭部など、エスカレーション E メールについてのいずれかの Human Task Manager 設定値を変更する必要がある場合、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」をクリックして変更を行います。
- E メール・サーバーのアドレス、ポート番号、ユーザー ID、またはパスワードを変更する必要がある場合、「**リソース**」 → 「**メール**」 → 「**メール・セッション**」をクリックして、「**セル**」スコープを選択して、「**HTM mail session_*suffix***」をクリックします。ここで、*suffix* は Business Process Choreographer が構成されている場所に応じて、*node_name_server_name* または *cluster_name* のいずれかです。変更を行います。

11. 担当者の割り当てに使用する担当者ディレクトリー・プロバイダーの種類によっては、以下のようにプロバイダーを構成する必要がある場合があります。
    - システムとユーザー・レジストリーの担当者ディレクトリー・プロバイダーは、構成せずに使用できます。
    - Lightweight Directory Access Protocol (LDAP) を使用している場合は、 225 ページの『LDAP 担当者ディレクトリー・プロバイダーの構成』を実行します。
    - Virtual Member Manager (VMM) を使用している場合は、 224 ページの『Virtual Member Manager 担当者ディレクトリー・プロバイダーの構成』を実行します。
12. オプション: VMM を構成しており、担当者の代替を使用する場合は、 231 ページの『担当者の代替の構成』を行います。
13. オプション: グループ作業項目を使用する場合は、管理コンソールでグループ作業項目を有効にします。 「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」をクリックしてから、「**グループ作業項目の使用可能化**」を選択します。
14. WebSphere アプリケーション・セキュリティーを有効にしており、リモート EJB メソッドを呼び出す長期実行のプロセスがある場合、Common Secure Interoperability Version 2 (CSIv2) のインバウンド認証構成で CSIv2 の ID アサーションが有効になっていることを確認してください。 これについて詳しくは、『Common Secure Interoperability バージョン 2 インバウンド通信の構成』を参照してください。
15. Business Process Choreographer を Network Deployment 環境で構成している場合、以下のようにします。
    a. ロード・バランシングおよびフェイルオーバーを実現するために、BPEContainer および TaskContainer アプリケーション用の Web モジュールを Web サーバーにマップします。 REST API および JAX Web サービス

API のためのデフォルトのコンテキスト・ルートを変更して、ホスト名およびポートの各組み合わせに対してコンテキスト・ルートが固有となるようにすることが必要な場合があります。コンテキスト・ルートを設定するには、以下を実行します。

1) Business Flow Manager に対してコンテキスト・ルートを設定するには、「**アプリケーション**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere エンタープライズ・アプリケーション**」をクリックし、次に「**BPEContainer_*suffix***」 → 「**Web モジュールのコンテキスト・ルート**」をクリックします。ここで、*suffix* は、Business Process Choreographer が構成されている *node_name_server_name* または *cluster_name* です。次に、Web モジュールの BFMRESTAPI および BFMJAXWSAPI のコンテキスト・ルートが正しく固有であることを確認します。

2) Human Task Manager に対してコンテキスト・ルートを設定するには、「**アプリケーション**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere エンタープライズ・アプリケーション**」をクリックし、次に「**TaskContainer_*suffix***」 → 「**Web モジュールのコンテキスト・ルート**」をクリックします。ここで、*suffix* は、Business Process Choreographer が構成されている *node_name_server_name* または *cluster_name* です。次に、Web モジュールの HTMRESTAPI および HTMJAXWSAPI のコンテキスト・ルートが正しく固有であることを確認します。

b. REST API のコンテキスト・ルートのいずれかを変更した場合は、対応するエンドポイントも変更する必要があります。

1) Business Process Choreographer Explorer を使用する場合: REST エンドポイントを変更して新しいコンテキスト・ルートに一致させるには、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Business Process Choreographer Explorer**」をクリックしてから新しい値を設定します。例えば、Business Flow Manager REST API のコンテキスト・ルートが `/rest/bpm/bfm` の場合、絶対 URL は `http://localhost:9080/rest/bpm/bfm` のようになります。

2) Business Space を使用する場合: 新規コンテキスト・ルートに一致するように REST エンドポイントを変更します。これを行うには、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」をクリックして、「**Business Flow Manager**」または「**Human Task Manager**」のいずれかをクリックして、「**追加プロパティー**」の下で「**REST サービスのエンドポイント**」をクリックして、新しい値を設定します。

16. 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。 235 ページの『Business Process Choreographer Explorer の構成』を実行します。

タスクの結果

Business Process Choreographer が構成されます。

次のタスク

Common Event Infrastructure をまだ構成していない場合は、doc/tcei_configuration.dita を実行してください。

bpeconfig.jacl スクリプトを使用した Business Process Choreographer の構成

bpeconfig.jacl スクリプトを使用して、特定のサーバーまたはクラスターで、Business Process Choreographer と必要なすべてのリソースを構成する方法を説明します。

手順

1. 使用するオプションとパラメーターが判明していることを確認します。 137 ページの『第 3 章 Business Process Choreographer の構成計画』で計画した値を参照してください。このスクリプトをバッチ・モードで実行する場合は、すべての必須パラメーターが含まれていなければなりません。このスクリプトを対話式に実行する場合は、コマンド行で指定されていない必須パラメーターがあると、プロンプトが表示されます。スクリプト、例、スクリプトのオプションおよびパラメーターについての詳細は、 201 ページの『bpeconfig.jacl スクリプト』を参照してください。

オプション	説明
サーバー (Network Deployment 環境ではデプロイメント・マネージャー) が稼働していない場合	次のオプションを使用します。 `-conntype NONE` サーバー (またはデプロイメント・マネージャー) が稼働している場合は、このオプションを使用しないでください。
管理セキュリティーが有効な場合	次のパラメーターを指定します。 `-user` *userName* `-password` *userPassword*
デフォルト・プロファイルを使用しない場合	次のパラメーターを指定します。 `-profileName` *profileName*
デフォルト・サーバーで Business Process Choreographer を構成しない場合	次のいずれかのパラメーターを指定します。 `-cluster` *clusterName* または次の両方のパラメーター: `-node` *nodeName* `-server` *serverName*

オプション	説明
スクリプトで常に Business Process Choreographer 構成が作成される	Business Flow Manager および Human Task Manager に必要なパラメーターを指定します。 {-adminUsers *userList* | -adminGroups *groupList*} [-adminJobUser *userID* -adminJobPwd *password*] {-monitorUsers *userList* | -monitorGroups *groupList*} -jmsBFMRunAsUser *userID* -jmsBFMRunAsPwd *password* -jmsHTMRunAsUser *userID* -jmsHTMRunAsPwd *password* -contextRootBFMWS *contextRootBFMWS* -contextRootBFMREST *contextRootBFMREST* -contextRootHTMWS *contextRootHTMWS* -contextRootHTMREST *contextRootHTMREST* *Users* と *Groups* で終わるパラメーターのペアの場合、いずれか 1 つまたは両方のパラメーターを指定する必要があります。*contextRoot* で始まる 2 つのパラメーターはオプションです。
エスカレーション E メール送信用に Simple Mail Transfer Protocol (SMTP) サーバーを有効にする場合	次のパラメーターを指定します。 -mailServerName *mailServerName* メール・サーバーで認証が必要な場合は、次のパラメーターも指定します。 -mailUser *mailUserID* -mailPwd *mailPassword*
スクリプト・ファイルによりデータベースを作成するか、または SQL スクリプトを生成してスクリプトを実行せずにおくことができる	次のオプションを使用します。 -createDB { yes | no } 構成が完了した後で、別個にデータベースを作成するため、no を選択します。yes を選択すると、スクリプトは DB2 for z/OS データベースを作成できないため失敗します。

オプション	説明

すべての Business Process Choreographer 構成でデータベースへのアクセスが必要である	データベース設計ツールを使用してデータベース設計ファイルを作成した場合は、次のパラメーターを含めます。
	`-bpcdbDesign databaseDesignFile`
	
	データベース設計ファイルで指定されている値は、コマンド行に含まれている次のパラメーターのあらゆるオカレンスに優先します。-dbJava、-dbName、-dbPwd、-dbSchema、-dbServerName、-dbServerPort、-dbTablespaceDir、-dbType、-dbUser、-driverTypeデータベース設計ファイルを指定しない場合は、次のパラメーターを含めます。
	`-dbType databaseType`
	
	また、使用しているデータベース・タイプに必要なパラメーターも指定します。 201 ページの『bpeconfig.jacl スクリプト』を参照してください。
	`-dbVersion version`
	`-dbHome databaseInstallPath`
	`-dbJava JDBCDriverPath`
	`-dbName databaseName`
	`-dbUser databaseUser`
	`-dbPwd databasePassword`
	`-dbAdmin databaseAdministratorUserID`
	`-driverType JDBCDriverType`
	`-dbTablespaceDir databaseTablespacePath`
	`-dbServerName databaseServerName`
	`-dbServerPort databaseServerPort`
	`-dbStorageGroup DB2zOSStorageGroup`
	`-dbConnectionTarget DB2zOSSubSystem`
	`-dbSchema schemaQualifier`
	
	クラスターで、このスクリプトをバッチ・モードで実行するときに、ご使用のデータベースが -dbJava パラメーターを必要としている場合は、クラスター・メンバーをホストするノードごとに、このパラメーターを次のように指定します。
	`-dbJava.nodeName JDBCDriverPath _on_nodeName`

すべての Business Process Choreographer 構成で JMS プロバイダーが使用されている	次のパラメーターを指定します。
	`-mqType { WPM | MQSeries }`
	
	また、使用している JMS プロバイダーに必要なパラメーターも指定します (詳しくは 201 ページの『bpeconfig.jacl スクリプト』を参照)。
	`-createQM { yes | no }`
	`-qmNameGet getQueueManagerName`
	`-mqClusterName mqClusterName`
	`-qmNamePut putQueueManagerName`
	`-mqHome MQInstallationDirectory`
	`-mqUser JMSProviderUserID`
	`-mqPwd JMSProviderPassword`
	
	注: MQSeries® オプションは使用すべきではありません。

オプション	説明
-mqType WPM オプションを使用するときに、SCA がメッセージ・ストアとしてデータベースを使用していれば、Business Process Choreographer のメッセージング・エンジン・ストア設定を指定する	次のパラメーターを指定します。 -mqCreateTables { true | false } -mqSchemaName *mqSchemaName* -medbUser *meDatabaseUser* -medbPwd *meDatabasePassword*
スクリプトでは常に Business Process Choreographer Explorer が構成される	次のいずれかのパラメーターを指定します。 -contextRootExplorer *explorerContextRoot* -explorerHost *explorerURL* -hostName *explorerVirtualHostname* -maxListEntries *maximum* -remoteCluster *clusterName* -remoteNode *nodeName* -remoteServer *serverName* -restAPIBFM *restAPIURL** -restAPIHTM *restAPIURL** Business Process Choreographer Explorer レポート作成機能と Event Collector アプリケーションを構成する場合は、以下のオプションを使用します。 -createEventCollector { yes | no } -reportFunction { yes | no } -reportAtSnapshotRange *number* -reportCreateTables { true | false } -reportDataSource *jndiName* -reportSchemaName *schemaName* デフォルト値を含むこれらのパラメーターについて詳しくは、201 ページの『bpeconfig.jacl スクリプト』を参照してください。 **注:** * Network Deployment 環境では、-restAPIBFM と -restAPIHTM は必須です。 **制約事項:** -createEventCollector yes オプションは、バッチ・モードでスクリプトを実行する場合にのみサポートされます。

2. bpeconfig.jacl スクリプト・ファイルを、バッチ・モードで計画したオプションと構成パラメーターを指定して呼び出すか、または対話モードで呼び出します。 スクリプト・ファイルの詳細については、201 ページの『bpeconfig.jacl スクリプト』を参照してください。
3. WebSphere MQ Java Message Service (JMS) プロバイダーを使用するときに、-createQM no オプションを指定して、スクリプトがキュー・マネージャーおよびキューを作成しないようにした場合は、213 ページの『Business Process Choreographer 用のキュー・マネージャーとキューの作成』をここで実行することにより、キュー・マネージャーおよびキューを作成します。
4. Business Process Choreographer をアクティブにします。 280 ページの『Business Process Choreographer の活動化』を実行します。
5. オプション: Business Process Choreographer 基本構成が機能することを確認します。 281 ページの『Business Process Choreographer の作動確認』を実行します。
6. オプション: JMS 認証ユーザー ID、run-as ユーザー ID、またはユーザーとグループへのロールのマッピングを変更する場合は、「**セキュリティー**」 → 「**ビジネス・インテグレーション・セキュリティー**」をクリックし、セキュリティー設定を変更します。

7. オプション: Human Task Manager の設定を変更します。
   - 送信者アドレスまたは Business Process Choreographer Explorer についての URL 接頭部など、エスカレーション E メールについてのいずれかの Human Task Manager 設定値を変更する必要がある場合、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」をクリックして変更を行います。
   - E メール・サーバーのアドレス、ポート番号、ユーザー ID、またはパスワードを変更する必要がある場合、「**リソース**」 → 「**メール**」 → 「**メール・セッション**」をクリックして、「**セル**」スコープを選択して、「**HTM mail session_*suffix***」をクリックします。ここで、*suffix* は Business Process Choreographer が構成されている場所に応じて、*node_name_server_name* または *cluster_name* のいずれかです。変更を行います。
8. 担当者の割り当てに使用する担当者ディレクトリー・プロバイダーの種類によっては、以下のようにプロバイダーを構成する必要がある場合があります。
   - システムとユーザー・レジストリーの担当者ディレクトリー・プロバイダーは、構成せずに使用できます。
   - Lightweight Directory Access Protocol (LDAP) を使用している場合は、 225 ページの『LDAP 担当者ディレクトリー・プロバイダーの構成』を実行します。
   - Virtual Member Manager (VMM) を使用している場合は、 224 ページの『Virtual Member Manager 担当者ディレクトリー・プロバイダーの構成』を実行します。
9. オプション: VMM を構成しており、担当者の代替を使用する場合は、 231 ページの『担当者の代替の構成』を行います。
10. オプション: グループ作業項目を使用する場合は、管理コンソールでグループ作業項目を有効にします。 「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」をクリックしてから、「**グループ作業項目の使用可能化**」を選択します。
11. WebSphere アプリケーション・セキュリティーを有効にしており、リモート EJB メソッドを呼び出す長期実行のプロセスがある場合、Common Secure Interoperability Version 2 (CSIv2) のインバウンド認証構成で CSIv2 の ID アサーションが有効になっていることを確認してください。 これについて詳しくは、『Common Secure Interoperability バージョン 2 インバウンド通信の構成』を参照してください。
12. 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。 235 ページの『Business Process Choreographer Explorer の構成』を実行します。

13. Business Process Choreographer を Network Deployment 環境で構成している場合、以下のようにします。
    a. ロード・バランシングおよびフェイルオーバーを実現するために、BPEContainer および TaskContainer アプリケーション用の Web モジュールを Web サーバーにマップします。 REST API および JAX Web サービス API のためのデフォルトのコンテキスト・ルートを変更して、ホスト名およびポートの各組み合わせに対してコンテキスト・ルートが固有となるようにすることが必要な場合があります。コンテキスト・ルートを設定するには、以下を実行します。
        1) Business Flow Manager に対してコンテキスト・ルートを設定するには、**「アプリケーション」** → **「アプリケーション・タイプ」** → **「WebSphere エンタープライズ・アプリケーション」**をクリックし、次に**「BPEContainer_*suffix*」** → **「Web モジュールのコンテキスト・ルート」**をクリックします。ここで、*suffix* は、Business Process Choreographer が構成されている *node_name_server_name* または *cluster_name* です。次に、Web モジュールの BFMRESTAPI および BFMJAXWSAPI のコンテキスト・ルートが正しく固有であることを確認します。
        2) Human Task Manager に対してコンテキスト・ルートを設定するには、**「アプリケーション」** → **「アプリケーション・タイプ」** → **「WebSphere エンタープライズ・アプリケーション」**をクリックし、次に**「TaskContainer_*suffix*」** → **「Web モジュールのコンテキスト・ルート」**をクリックします。ここで、*suffix* は、Business Process Choreographer が構成されている *node_name_server_name* または *cluster_name* です。次に、Web モジュールの HTMRESTAPI および HTMJAXWSAPI のコンテキスト・ルートが正しく固有であることを確認します。
    b. REST API のコンテキスト・ルートのいずれかを変更した場合は、対応するエンドポイントも変更する必要があります。
        1) Business Process Choreographer Explorer を使用する場合: REST エンドポイントを変更して新しいコンテキスト・ルートに一致させるには、**「サーバー」** → **「クラスター」** → **「WebSphere Application Server クラスター」** → **「クラスター名」**または**「サーバー」** → **「サーバー・タイプ」** → **「WebSphere Application Server」** → **「サーバー名」**のいずれかをクリック後、**「構成」**タブの**「ビジネス・インテグレーション」**セクションで、**「Business Process Choreographer」**を展開し、**「Business Process Choreographer Explorer」**をクリックしてから新しい値を設定します。例えば、Business Flow Manager REST API のコンテキスト・ルートが `/rest/bpm/bfm` の場合、絶対 URL は `http://localhost:9080/rest/bpm/bfm` のようになります。
        2) Business Space を使用する場合: 新規コンテキスト・ルートに一致するように REST エンドポイントを変更します。これを行うには、**「サーバー」** → **「クラスター」** → **「WebSphere Application Server クラスター」** → **「クラスター名」**または**「サーバー」** → **「サーバー・タイプ」** → **「WebSphere Application Server」** → **「サーバー名」**のいずれかをクリック後、**「構成」**タブの**「ビジネス・インテグレーション」**セクションで、**「Business Process Choreographer」**をクリックして、**「Business**

Flow Manager」または「**Human Task Manager**」のいずれかをクリックして、「**追加プロパティー**」の下で「**REST サービスのエンドポイント**」をクリックして、新しい値を設定します。

タスクの結果

Business Process Choreographer が構成されます。

bpeconfig.jacl スクリプト

このスクリプト・ファイルは、Business Process Choreographer と、サーバーまたはクラスター上の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。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ます。また、オプションで Business Process Choreographer Explorer と Business Process Choreographer Explorer レポート作成機能を構成できます。

場所

bpeconfig.jacl スクリプト・ファイルは、以下の Business Process Choreographer config 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。

DB2 for z/OS データベースの場合

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。

Network Deployment 環境でのスクリプトの実行

構成スクリプトは、 wsadmin コマンドで実行されます。 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。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にコンフィギュレーター権限も管理者権限もない場合は、wsadmin -user および -password のオプションを付けて、コンフィギュレーターまたは管理者の権限を持つユーザー ID を指定します。

ビジネス・プロセス・コンテナー、Business Process Choreographer Explorer、および Business Process Choreographer Explorer レポート作成機能の非対話式の構成

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、以下のコマンドのいずれかを入力します。

現行ディレクトリーが *install_root* の場合、次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。

```
-adminGroups groupList
-adminUsers userList
-adminJobPwd password
-adminJobUser userID
-bpcdbDesign databaseDesignFile
-cluster clusterName
-conntype NONE
-contextRootBFMWS contextRootBFMWS
-contextRootBFMREST contextRootBFMREST
-contextRootExplorer explorerContextRoot
-contextRootHTMWS contextRootHTMWS
-contextRootHTMREST contextRootHTMREST
-createDB { yes | no }
-createEventCollector { yes | no }
-createQM { yes | no }
-dbConnectionTarget DB2zOSSubSystem


-dbJava JDBCDriverPath

-dbPwd databasePassword
-dbSchema schemaQualifier
-dbServerName databaseServerName
-dbServerPort databaseServerPort
-dbStorageGroup DB2zOSStorageGroup
-dbTablespaceDir databaseTableSpacePath
-dbType databaseType
-dbUser databaseUser
-dbVersion version
-driverType JDBCDriverType
-explorerHost explorerURL
-hostName VirtualHostname
-jmsBFMRunAsPwd password
-jmsBFMRunAsUser userID
-jmsHTMRunAsPwd password
-jmsHTMRunAsUser userID
-mailPwd mailPassword
-mailServerName mailServerName
-mailUser mailUserID
-maxListEntries max
-medbPwd meDatabasePassword
-medbUser meDatabaseUser
-monitorGroups groupList
-monitorUsers userList
-mqClusterName mqClusterName
-mqCreateTables  { true | false }
-mqHome MQInstallationDirectory
-mqPwd JMSProviderPassword
-mqSchemaName mqSchemaName
-mqType JMSProviderType
-mqUser JMSProviderUserID
```

```
-node nodeName
-precompileJSPs { yes | no }
-qmNameGet getQueueManagerName
-qmNamePut putQueueManagerName
-remoteCluster clusterName
-remoteNode nodeName
-remoteServer serverName
-reportAtSnapshotRange number
-reportCreateTables { true | false }
-reportDataSource jndiName
-reportFunction { yes | no }
-reportSchemaName schemaName
-restAPIBFM restAPIURL
-restAPIHTM restAPIURL
-server serverName
```

注: これら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、および、パラメーターがオプションであるのか必須であるのかを決定する条件については、以下の各パラメーターの説明で示します。コマンド行で指定されていない必須パラメーターがある場合、対話式にプロンプトが表示されます。同じパラメーターが複数指定された場合、最後に指定された値が使用されます。

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。

-adminGroups *groupList*

groupList は、ユーザー・レジストリーに登録されているグループの名前のリストです。Java EE ロール BPESystemAdministrator と TaskSystemAdministrator をそれらのグループにマップします。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。adminUsers オプションと adminGroups 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。

-adminUsers *userList*

userList は、ユーザー・レジストリーに登録されているユーザーの名前のリストです。Java EE ロール BPESystemAdministrator と TaskSystemAdministrator をそれらのグループにマップします。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。adminUsers オプションと adminGroups 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。

-bpcdbDesign *databaseDesignFile*

データベース設計ツールを使用してデータベース設計ファイルを作成した場合は、このオプションを使用して、データベース設計ファイル *databaseDesignFile* を指定します。

データベース設計ファイルで指定されている値は、コマンド行に含まれている次のパラメーターのあらゆるオカレンスに優先します。-dbJava、-dbName、-dbPwd、-dbSchema、-dbServerName、-dbServerPort、-dbTablespaceDir、-dbType、-dbUser、-driverType

-adminJobPwd *password*

password は、管理ジョブ・ユーザー ID のパスワードです。

-adminJobUser *user ID*

このユーザー ID は、Business Flow Manager および Human Task Manager のクリーンアップ・サービスやプロセス・インスタンス・マイグレーション・ツールなどの管理ジョブを実行するために使用します。これらのサービスを使用しない場合、ユーザー ID を指定する必要はありません。このユーザー ID は任意指定です。このユーザー ID を指定する場合は、2 つの Java EE ロールである BPESystemAdminstrator および TaskSystemAdministrator の両方のメンバーであるユーザー ID を指定しなければなりません。

-cluster *clusterName*

clusterName は、Business Process Choreographer を構成するクラスターの名前です。このパラメーターはオプションです。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。

-conntype NONE

この指定によって、管理接続が使用不可になります。デプロイメント・マネージャー (Network Deployment の場合) が稼働していない場合にのみ、このオプションを指定してください。 これは wsadmin パラメーターですが、指定していない場合でも、指定するようにプロンプトは表示されません。

-contextRootBFMREST *contextRootBFMREST*

ここで *contextRootBFMREST* は、REST API 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、サーバーまたはクラスター上のデフォルト・コンテキスト・ルートは `/rest/bpm/bfm` です。

-contextRootBFMWS *contextRootBFMWS*

ここで *contextRootBFMWS* 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。

-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/bpc` のデフォルト URL になります。コンテキスト・ルートは、ホスト名とポートの組み合わせごとに固有のものでなければなりません。

-contextRootHTMREST *contextRootHTMREST*

ここで *contextRootHTMREST* は、REST API 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、サーバーまたはクラスター上のデフォルト・コンテキスト・ルートは `/rest/bpm/htm` です。

-contextRootHTMWS *contextRootHTMWS*

ここで *contextRootHTMWS* 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。

-createDB { yes | no }

この値を `no` に設定します。このスクリプトでは、DB2 for z/OS データベースは作成できません。

-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 Explorer レポート作成機能では必要な設定です。-createEventCollector の値を yes にした場合は、-report* パラメーターを使用して、Business Process Choreographer Explorer のレポート作成機能 (旧称 Business Process Choreographer Observer) のオプションを指定できます。例えば、Business Process Choreographer の BPEDB データベースを共用するのではなく、別個のレポート・データベースを指定する場合は、-reportDataSource を使用できます。Business Process Choreographer Event Collector アプリケーションを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。

-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が、この値は使用すべきではありません。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。

-dbConnectionTarget *DB2zOSSubSystem*
DB2zOSSubSystem は、Business Process Choreographer データベース表およびデータ・ソースを作成するときに使用される DB2 接続ターゲット・ロケーションです。デフォルト値は `BPEDB` です。

-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。

- DB2 for z/OS とタイプ 4 ドライバー。デフォルト値は *databaseInstallPath*`/java` です。

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。

このスクリプトをバッチ・モードで実行してクラスターを構成するときに、ご使用のデータベースで `-dbJava` パラメーターが必要とされる場合は、クラスター・メンバーをホストするノードごとに、このパラメーターを次のように指定します。

`-dbJava.`*nodeName JDBCDriverPath*`_on_`*nodeName*

JDBCDriverPath は JDBC ドライバーのパス、*nodeName* はノードの名前です。

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

-dbSchema *schemaQualifier*
schemaQualifier は、Business Process Choreographer データベース表およびデータ・ソースを作成するときに使用されるスキーマ修飾子です。デフォルト値は空です。この場合、使用するデータベースのタイプによって決まる暗黙的なスキーマ修飾子が使用されます。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。

-dbServerName *databaseServerName*

ここで *databaseServerName* は、Business Process Choreographer 用のデータベースをホストするネーム・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。

- DB2 の場合、デフォルト値は空です。

-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 `446` です。

-dbStorageGroup *DB2zOSStorageGroup*

ここで *DB2zOSStorageGroup* は、Business Process Choreographer データベース表を作成するために使用されるストレージ・グループです。デフォルト値はなく、空にすることはできません。

-dbTablespaceDir *databaseTableSpacePath*

ここで *databaseTableSpacePath* は、データベース表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表を作成するために使用されます。

- 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。

-dbType *databaseType*

ここで *databaseType* は、データベース・タイプです。このパラメーターは、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表の作成、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。`zOS-DB2` を使用します。

-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・ソースを作成するために使用されます。デフォルト値は、「db2inst1」です。

-dbVersion *version*

ここで *version* はデータベースのバージョン番号です。デフォルト値はありません。

- DB2 for z/OS の場合、*version* の値は `8`、または `9` のいずれかでなければなりません。

-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 for Linux®、UNIX®、Windows®、および z/OS プラットフォームの場合、指定可能な値は `Universal` または `DataServer` です。 これは、データベース表を作成する場合にも使用されます。

-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、このエクスプローラー・インスタンスにリンクするために Human Task Manager により使用されます。

-hostName *VirtualHostname*
ここで、*VirtualHostname* は、Business Process Choreographer Explorer、Business Flow Manager API および Human Task Manager API の Web サービス・バインディング、Business Flow Manager API および Human Task Manager API の REST バインディングが実行される仮想ホストです。デフォルト値は `default_host` です。

-jmsBFMRunAsPwd *password*
ここで *password* は、jmsBFMRunAsUser ユーザー ID のパスワードです。このプロパティーは、ビジネス・プロセス・コンテナーを構成する際に必要です。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。

-jmsBFMRunAsUser *user ID*
ここで *user ID* は Java EE ロール JMSAPIUser 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。このプロパティーは、ビジネス・プロセス・コンテナーを構成する際に必要です。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、jmsHTMRunAsUser ユーザー ID のパスワードです。このプロパティーは、ヒューマン・タスク・コンテナーを構成する際に必要です。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。

-jmsHTMRunAsUser *user ID*
ここで *user ID* は Java EE ロール EscalationUser 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。このプロパティーは、ヒューマン・タスク・コンテナーを構成する際に必要です。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。

-mailPwd *mailPassword*
mailPassword は、ユーザー ID *mailUserID* のパスワードです。このパラメーターは、メール・サーバーで認証が必要な場合にのみ必要です。それ以外の場合は省略できます。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ときに、このパラメーターが必要です。

-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。このパラメーターに空の値が設定されている場合は、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。このパラメーターは、メール・サーバーで認証が必要な場合にのみ必要です。それ以外の場合は省略できます。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ときに、このパラメーターが必要です。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。

-maxListEntries *maximum*

maximum は、照会に対し Business Process Choreographer Explorer から戻される結果の最大数です。デフォルトは 10000 です。

-medbPwd *MEDBPassword*

MEDBPassword は、medbUser パラメーター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。

-medbUser *MEDBUserID*

MEDBUserID は、メッセージング・エンジン・データベースにアクセスするためのユーザー ID です。このパラメーターのデフォルト値は dbUser パラメーターの値です。このパラメーターが必要となるのは、SCA がデータベースを使用しており、メッセージング・エンジンへのアクセスに Derby Embedded JDBC プロバイダーを使用しない場合に限られます。

-monitorGroups *groupList*

groupList は、ユーザー・レジストリーに登録されているグループの名前のリストです。Java EE ロール BPESystemMonitor と TaskSystemMonitor をそれらのグループにマップします。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。monitorUsers と monitorGroups のいずれかまたは両方を設定する必要があります。

-monitorUsers *userList*

userList は、ユーザー・レジストリーに登録されているユーザーの名前のリストです。Java EE ロール BPESystemMonitor と TaskSystemMonitor をそれらのグループにマップします。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。monitorUsers と monitorGroups のいずれかまたは両方を設定する必要があります。

-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。このパラメーターは、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。この値は使用すべきではありません。

-mqClusterName *mqClusterName*

mqClusterName は、キュー・マネージャーがメンバーになる WebSphere MQ クラスターの名前です。このパラメーターはオプションです。デフォルト値は

MQCluster 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が、この値は使用すべきではありません。

-mqCreateTables *{ true | false}*
このブール値パラメーターが有効となるのは、mqType オプションを WPM に設定し、Service Component Architecture (SCA) がメッセージ・ストアとして FILESTORE ではなくデータベースを使用している場合に限られます。このパラメーターは、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。 SCA 設定からデフォルト値が継承されますが、このパラメーターを使用すればそのデフォルト値をオーバーライドできます。

-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。このパラメーターは、リスナー・ポートとキュー接続ファクトリーの作成に使用されます。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が、この値は使用すべきではありません。

-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。このパラメーターが有効なのは、SCA がメッセージ・ストアとして FILESTORE ではなくデータベースを使用している場合に限られます。Business Process Choreographer は、SCA と同じデータベースを使用しますが、別のスキーマを使用することになります。このパラメーターを使用して、デフォルトのスキーマ名をオーバーライドできます。デフォルト値は、WPRBM00 などの生成された値です。

-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、Business Process Choreographer Service Integration (SI) バスに対して認証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。

-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。

-precompileJSPs { no | yes }
Java Server Pages (JSPs) をプリコンパイルするかどうかを識別します。デフォルトは no です。プリコンパイル済み JSP はデバッグできない点に注意してください。

-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が、この値は使用すべきではありません。

-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。

-remoteCluster *clusterName*
ローカル側の Business Process Choreographer 構成に接続せず、remoteNode および remoteServer を指定しない場合は、このパラメーターを使用します。このパラメーターが指定されていない場合、デフォルトで -cluster パラメーターの値が使用されます。

-remoteNode *nodeName*
ローカル側の Business Process Choreographer 構成に接続しない場合は、このパラメーターと remoteServer を使用します。このパラメーターが指定されていない場合、デフォルトで -node パラメーターの値が使用されます。

-remoteServer *serverName*
ローカル側の Business Process Choreographer 構成に接続しない場合は、このパラメーターと remoteNode を使用します。このパラメーターが指定されていない場合、デフォルトで -server パラメーターの値が使用されます。

-reportAtSnapshotRange *number*
スナップショット・レポートは、スナップショットの限定日時より古いすべてのイベントを評価することによって作成されます。このオプション・パラメーターは、スナップショット・レポートにイベントを組み込むことができる日数を定義します。この期間内に発行されたイベントのみがスナップショット・レポートで評価されます。デフォルトは 60 日です。このオプション・パラメーターが有効となるのは、`-reportFunction yes` オプションでレポート作成機能が有効になっている場合に限られます。

この値が高すぎると、大量のイベントを処理する必要が生じ、レポート生成に長い時間がかかる可能性があります。この値は、ビジネス環境におけるプロセス・インスタンスの最大期間に設定するようにしてください。

-reportCreateTables { true | false }
このオプション・パラメーターでは、Business Process Choreographer Explorer が初めてデータベースに接続する時点で、Business Process Choreographer Explorer レポート作成機能スキーマを作成するかどうかを指定します。デフォルトは true です。このオプション・パラメーターが有効となるのは、`-reportFunction yes` オプションでレポート作成機能が有効になっている場合に限られます。

-reportDataSource *jndiName*
jndiName は、データベースへの接続に使用されるデータ・ソース JNDI の

JNDI 名です。-reportFunction yes を指定した場合は、このパラメーターは必須です。データ・ソースは自動では作成されません。

-reportFunction { yes | no }
このオプション・パラメーターでは、Business Process Choreographer Explorer のレポート作成機能を有効にするかどうかを制御します。対話モードでのデフォルトは no です。バッチ・モード場合は、以前のバージョンとの互換性のためにデフォルトは yes になっています。

-reportSchemaName *schemaName*
このオプション・パラメーターでは、すべてのレポート・データベース・オブジェクトのプレフィックスとして使用するデータベース・スキーマを指定します。スキーマ名を指定しない場合は、固有のスキーマ名が生成されます。このオプション・パラメーターが有効となるのは、-reportFunction yes オプションでレポート作成機能が有効になっている場合に限られます。デフォルト値は WPRBC00 です。

-restAPIBFM *restAPIURL*
ここで、*restAPIURL* は Business Flow Manager REST API の URL で、Business Process Choreographer Explorer のグラフィカル・プロセス・ウィジェットをサポートするために必要です。スタンドアロン・サーバーでは、デフォルトが計算されます (http://*localhost*:9080/rest/bpm/bfm など)。 Network Deployment 環境では、デフォルト値はありません。

-restAPIHTM *restAPIURL*
ここで、*restAPIURL* は Human Task Manager REST API の URL で、Business Process Choreographer Explorer のグラフィカル・プロセス・ウィジェットをサポートするために必要です。スタンドアロン・サーバーでは、デフォルトが計算されます (http://*localhost*:9080/rest/bpm/htm など)。 Network Deployment 環境では、デフォルト値はありません。

-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。

例: 非対話モードでのスタンドアロン・サーバーの構成

Windows プラットフォームで DB2 データベースを使用してスタンドアロン・サーバー上の Business Process Choreographer を構成する場合のバッチ・モード・コマンドは、以下のようになります。

```
wsadmin -conntype none -f bpeconfig.jacl
        -adminGroups bpcadmins -monitorGroups bpcmonitors
        -createDB no
        -createEventCollector no
        -dbSchema WPRBE00 -dbUser db2user -dbPwd secret
        -dbServerName db2host.acme.com -dbJava d:¥¥programs¥¥IBM¥¥SQLLIB¥¥java
        -dbTablespaceDir d:¥¥DB2¥¥tablespacedir -mqType WPM
        -dbType DB2 -dbName BPEDB
        -driverType Universal
        -explorerHost http://wpshost.acme.com:80/bpc
        -jmsBFMRunAsUser jmsuser -jmsBFMRunAsPwd secret
        -jmsHTMRunAsUser escalationuser -jmsHTMRunAsPwd secret
        -mailServerName smtphost.acme.com -mailUser {}
        -mqCreateTables true
        -mqSchemaName WPRBM00
        -mqUser sibuser -mqPwd secret
```

```
        -reportFunction no
        -restAPIBFM http://wpshost.acme.com:80/rest/bpm/bfm
        -restAPIHTM http://wpshost.acme.com:80/rest/bpm/htm
```

その他のプラットフォームでは、ファイル・システム・パスのみが異なります。

例: 非対話モードでのクラスターの構成

DB2 データベースを使用して、Windows ノード「Node01」と UNIX ノード「Node02」の 2 つのノードからなるクラスター「cluster1」上で Business Process Choreographer を構成する場合のバッチ・モード・コマンドは、以下のようになります。

```
wsadmin -conntype none -profileName Dmgr01 -f bpeconfig.jacl
        -adminUsers bpcadmins
        -cluster cluster1
        -contextRootBFMREST /rest/bpm/bfm
        -contextRootBFMWS /BFMIF_cluster1
        -contextRootExplorer /bpc
        -contextRootHTMWS /HTMIF_cluster1
        -contextRootHTMREST /rest/bpm/htm
        -createEventCollector no
        -dbJava.acmeNode01 "c:¥¥Progam Files¥¥IBM¥¥SQLLIB¥¥java"
        -dbJava.acmeNode02 /home/db2inst1/sqllib
        -dbName BPEDB62
        -dbSchema WPRBE00
        -dbType DB2
        -dbUser db2user -dbPwd secret
        -createDB no
        -explorerHost http://wps.acme.com/bpc
        -jmsBFMRunAsUser jmsuser -jmsBFMRunAsPwd secret
        -jmsHTMRunAsUser escalationuser -jmsHTMRunAsPwd secret
        -mailServerName smtphost.acme.com -mailUser {}
        -maxListEntries 5000
        -medbUser db2user
        -monitorUsers bpcmonitors
        -mqCreateTables true
        -mqSchemaName WPRBM00
        -mqType WPM -hostName default_host
        -mqUser sibuser -mqPwd secret
        -reportFunction no
        -restAPIBFM http://wps.acme.com/rest/bpm/bfm
        -restAPIHTM http://wps.acme.com/rest/bpm/htm
```

その他のプラットフォームでは、ファイル・システム・パスのみが異なります。

構成スクリプトの対話式実行

bpeconfig.jacl スクリプトは対話式に実行できます。

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 Explorer レポート作成機能で必要な Event Collector アプリケーションは構成できません。 Event Collector は、このスクリプトをバッチ・モードで実行するか、 259 ページの『Business Process Choreographer Event Collector アプリケーションの構成』を実行することで構成できます。

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 (wsadmin `-tracefile` パラメーターを使用して別のファイル名を指定していない場合)

どちらのファイルも、ユーザーのプロファイルの logs ディレクトリー内にあります。ディレクトリー *profile_root*/logs 内スクリプトを接続モードで実行する場合、wsadmin スクリプト・クライアントが接続するアプリケーション・サーバーまたはデプロイメント・マネージャーに基づいて名前が付けられた `logs` ディレクトリー内のサブディレクトリーにある、ファイル `SystemOut.log` および `SystemErr.log` も検査してください。

関連タスク

195 ページの『bpeconfig.jacl スクリプトを使用した Business Process Choreographer の構成』

bpeconfig.jacl スクリプトを使用して、特定のサーバーまたはクラスターで、Business Process Choreographer と必要なすべてのリソースを構成する方法を説明します。

Business Process Choreographer 用のキュー・マネージャーとキューの作成

ここでは、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。

始める前に

WebSphere MQ が既にインストールされている必要があります。

注: WebSphere MQ のサポートは非推奨です。

このタスクについて

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。

手順

1. オプション: 実動システムを作成している場合、キュー・マネージャーがどのディスク・ドライブを使用するかについて計画します。 永続的なキュー・データおよび WebSphere MQ のログのデフォルト・ロケーションを使用すると、キュー・マネージャーのパフォーマンスに悪影響を及ぼすことがあります。
   WebSphere MQ の資料にある推奨事項に従って、これらのロケーションの変更を検討してください。
2. WebSphere MQ クラスター・セットアップを作成しなかった場合は、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. キュー・マネージャーとキューを作成します。 次のように入力します。

      ```
      cd install_root/ProcessChoreographer/config
      createQueues.sh queueManager
      ```

      ここで、

      queueManager
      既存のキュー・マネージャーの名前または新規キュー・マネージャー

に付ける名前です。指定されたキュー・マネージャーが既に存在する場合は、それを使用してキューが作成されます。キュー・マネージャーが存在しない場合は、デフォルトのキューが作成される前に作成され、開始されます。

3. WebSphere MQ クラスターを使用する WebSphere クラスター・セットアップを作成する場合は、クラスター化キュー・マネージャーとキューの作成のみ実行します。
4. 中央キュー・マネージャーを使用する WebSphere クラスター・セットアップを作成する場合は、次のアクションを実行します。
   a. WebSphere Process Server の ProcessChoreographer ディレクトリーの config サブディレクトリーから中央キュー・マネージャーをホストするサーバーへ、create queues スクリプト・ファイルをコピーします。 次のファイルをコピーします。

      install_root/ProcessChoreographer/config/createQueues.sh

   b. キュー・マネージャーをホストするサーバーで、WebSphere MQ がインストールされていることと、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。
   c. キュー・マネージャーとキューを作成します。 次のように入力します。

      ```
      cd install_root/ProcessChoreographer/config
      createQueues.sh queueManager
      ```

      ここで、*queueManager* は、新規キュー・マネージャーに付ける名前です。
   d. 新規キュー・マネージャーのリスナーを追加します:

      次のコマンドを入力します。

      ```
      runmqlsr -t tcp -p port -m queueManager &
      ```

      ここで、*port* は、リスナーが listen するポートです。
   e. ポートおよびキュー・マネージャー・サービスの定義を追加します。
      1) キュー・マネージャーのポートを /etc/services ファイルに追加します。

         ```
         <Service:Name> <port>/tcp
           <Service:Name>    キュー・マネージャー・サービスの名前
           <port>            キュー・マネージャーのポート
         ```

      2) /etc/services ファイルで指定されたサービスを /etc/inetd.conf ファイルに追加します。

         ```
         <Service:Name> stream tcp nowait mqm /usr/mqm/bin/amqcrsta amqrsta
                        -m QueueManager
           <Service:Name>    キュー・マネージャー・サービスの名前
           <Service:Name>    キュー・マネージャーの名前
         ```

タスクの結果

キュー・マネージャーとキューが作成されます。

Business Process Choreographer 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。

手順

1. 各ノードで、次のアクションを実行します。
   a. MQ クラスターを使用する場合は、以下の方法でそれをセットアップする計画を立てます。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。 bpeconfig.jacl を使用して、Business Process Choreographer をクラスター上に構成した場合は、その後でアプリケーション・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 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。

      注: WebSphere MQ を使用すべきではありません。
   b. 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。
   c. 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、次のコマンドを入力してキューを作成します。

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

      ここで、

      getQueueManager
      get キュー・マネージャーに付ける固有の名前。このキュー・マネージャーは、すべてのローカル・キューをホストします。

      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。

      putQueueManager
      put キュー・マネージャーの固有の名前。このキュー・マネージャーがホストとなるキューはありません。このことにより、メッセージは必ずすべての get キューに配布されます。

      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。

d. 次のコマンドを入力して、WebSphere MQ コマンド・プロセッサーを開始します。

```
runmqsc getQueueManager
```

e. 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)
```

f. 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(' ')
```

g. 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)
```

ここで、

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。

h. 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。

```
runmqlsr -t tcp -p port -m QueueManager
```

2. オプション: 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(*)
   ```

タスクの結果

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。

生成済みの SQL スクリプトを使用した Business Process Choreographer 用のデータベース・スキーマの作成

Business Process Choreographer を構成すると、Business Process Choreographer 用のデータベース・オブジェクトを作成する SQL スクリプトが生成されます。

始める前に

管理コンソールまたは bpeconfig.jacl スクリプトを使用して Business Process Choreographer を構成しました。bpeconfig.jacl スクリプトを使用して Business Process Choreographer を構成した場合は、ユーザーが `-createDB no` オプションを使用してデータベース・オブジェクトの作成を先延ばしにしました。

このタスクについて

Business Process Choreographer の構成時に提供した、関連するすべての構成パラメーターが、生成された SQL ファイル内で置き換えられています。ハイパフォーマンスの Business Process Choreographer 構成のためのデータベースを必要としているか、またはデータベース管理者がデータベースを作成する必要があります。あるいは、その両方が必要です。

手順

1. 生成された `createSchema.sql` SQL スクリプトを見つけます。
   - 管理コンソールを使用するか、接続モードで `bpeconfig.jacl` スクリプトを実行して、Network Deployment 環境に Business Process Choreographer を構成した場合、`createSchema.sql` スクリプト・ファイルはデプロイメント・マネージャーのノード上に生成されます。
   - 切断モードで `bpeconfig.jacl` スクリプトを実行して Business Process Choreographer を構成した場合、`createSchema.sql` スクリプト・ファイルはスタンドアロン・サーバーのノード上に生成されます。

   `createSchema.sql` という名前の ASCII SQL スクリプトと、`createSchema.ddl` という名前の、それと等価な EBCDIC DDL スクリプトがあります。
   - スキーマ修飾子を指定した場合は、どちらのファイルも *profile_root*/dbscripts/ProcessChoreographer/*database_type*/*database_name*/*database_schema* にあります。
   - スキーマ修飾子を指定**しなかった**場合は、どちらのファイルも *profile_root*/dbscripts/ProcessChoreographer/*database_type*/*database_name* にあります。

   各部の意味は、次のとおりです。

database_type
以下のストリングのいずれかで、生成されるスクリプトでサポートされるデータベース・システムが識別されます。
- DB2zOSV8
- DB2zOSV9

database_name
データベースの名前です。

database_schema
スキーマを使用している場合は、そのスキーマの名前です。

2. DB2 for z/OS データベースを手動で作成する必要があります。
3. データベースがリモートの場合は、生成されたスクリプトをデータベース・ホストにコピーします。これを実行する権限を付与されていない場合は、データベース管理者にこのスクリプトのコピーを渡し、管理者と要件について検討してください。
4. この SQL スクリプトは、以下のようにして、ユーザーまたはデータベース管理者がカスタマイズできます。
   a. 管理コンソールを使用して Business Process Choreographer を構成した場合は、次のプレースホルダーを実際の値に置き換えてください。
      - DB2 on z/OS の場合: `@STOGRP@` をストレージ・グループ名に置き換えます。デフォルト値は SYSDEFLT です。
   b. ハイパフォーマンス・システムが必要な場合には、 163 ページの『BPEDB データベースの計画』のステップ 6 (165 ページ) で計画したディスクおよびテーブル・スペースの割り振りを指定します。
5. 以下のいずれかのコマンドを使用して、データベース・ホストで SQL スクリプトを実行します。

オプション	説明

z/OS 上の DB2 の場合	ASCII バージョンの場合:
	`db2 -tf createSchema.sql`
	
	EBCDIC バージョンの場合:
	`db2 -tf createSchema.ddl`

6. 存在するすべての Business Process Choreographer 構成について、データベースにリモートでアクセスするよう Java Database Connectivity (JDBC) を構成します。 以下のいずれかで、次のステップを実行します。
   - Business Process Choreographer が構成されているクラスターのメンバーをホストする各ノード。
   - Business Process Choreographer を実行するサーバー。
   a. データベース・サーバーが Business Process Choreographer サーバーと異なる場合、アプリケーション・サーバーをホストするサーバーに、適切なタイプ 2 のデータベース・クライアントまたはタイプ 4 の JDBC ドライバーをインストールします。

b. タイプ 2 の JDBC ドライバーを使用している場合は、データベース・クライアントに新規データベースを認識させます。データベースは、カタログ化され、別名を使用してアクセス可能である必要があります。

c. 管理コンソールを使用して、データベースとの接続をテストします。

1) 「**リソース**」 → 「**JDBC**」 → 「**Business Integration Data Sources**」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。

注: クラスター化された Business Process Choreographer 構成の場合、データ・ソースはクラスター・レベルで定義されます。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) JNDI 名が jdbc/BPEDB のデータ・ソースを見つけて選択します。

4) 「**テスト接続**」をクリックします。

5) 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。

タスクの結果

Business Process Choreographer データベースが存在します。

SQL スクリプトを使用した Business Process Choreographer のためのデータベースの作成

Business Process Choreographer を構成する前に、あるいは製品をインストールする前でも、Business Process Choreographer のためのデータベースを手動で作成することができます。

始める前に

163 ページの『BPEDB データベースの計画』を実行している。

このタスクについて

組織で、データベースを別々のデータベース管理者によって作成することが必要な場合があります。管理コンソールまたは bpeconfig.jacl スクリプトを使用して Business Process Choreographer を構成する場合、カスタマイズ済みの SQL スクリプトが作成され、これを DBA に渡して BPEDB データベースを作成することができます。ただし、Business Process Choreographer を構成する前や、あるいは製品のインストール前にデータベースを作成したい場合、DBA はカスタマイズ済みでない SQL スクリプトを使用する必要があります。このトピックでは、カスタマイズ済みでない SQL スクリプト (製品メディアから入手可能) の使用方法について説明します。

手順

1. ローカルの DB2 for z/OS データベースを使用するには、そのデータベースを手動で作成する必要があります。
2. データベースをホストするサーバーで、以下のようにします。 221 ページの『Business Process Choreographer のための DB2 for z/OS データベースの作成』を参照してください。

タスクの結果

Business Process Choreographer データベースが存在します。

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

Business Process Choreographer を構成、または製品をインストールする前に、Business Process Choreographer 用に Derby データベースを作成する場合のみ、このタスクを使用します。

始める前に

163 ページの『BPEDB データベースの計画』は完了しています。Derby データベースは、WebSphere Process Server と共にインストールされています。ただし、製品をインストールする前にデータベースを作成するには、Derby のインストール済み環境がデータベース・サーバー上に既に存在する必要があります。

このタスクについて

BPEDB という名前の Derby 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。

手順

1. Derby Embedded データベースではなく、Derby Network Server データベースを作成する場合、Derby Network Server が稼働中で、Derby Network Server JDBC プロバイダーの使用を計画済みであることを確認してください。
2. 以下のいずれかを実行することで、データベースの親ディレクトリーを作成します。
   - デフォルトの場所にデータベースを作成する準備として、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。*profile_root*/databases を作成します。 新規ディレクトリーに移動します。
   - デフォルト以外の場所にデータベースを作成する準備として、新しいデータベースを作成するディレクトリーに移動します。
3. データベース作成スクリプト *media_root* または *extract_directory*¥dbscripts¥ProcessChoreographer/Derby/createDatabase.sql をコピーします。
4. ヘッダーの指示に従って、データベース作成スクリプト createDatabase.sql のコピーをカスタマイズします。データベースの名前を含める必要があります。サンプル・ファイルは ASCII フォーマットで提供されています。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 createDatabase.sql) を使用してファイルを ASCII フォーマットで直接編集し、次に iconv コマンドを使用して、このファイルを EBCDIC に変換すれば、vi を使用できます。

   ```
   iconv -t IBM-1047 -f ISO8859-1 createDatabase.sql >
   createDatabase_EBCDIC.sql
   ```
5. データベースを作成します。 次のように入力します。

```
java -Djava.ext.dirs=install_root/Derby/lib
     -Dij.protocol=jdbc:db2j: com.ibm.db2j.tools.ij
     install_root/dbscripts/ProcessChoreographer/Derby/createDatabase.sql
```

タスクの結果

Business Process Choreographer のためのデータベースが存在します。

Business Process Choreographer のための DB2 for z/OS 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。

始める前に

163 ページの『BPEDB データベースの計画』は完了しています。

WebSphere Process Server for z/OS での DB2 Universal JDBC Driver のサポートに関する情報については、『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、以下の更新が必要です。

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

このタスクについて

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と、オプションとして、このデータベースがアプリケーション・サーバーをホストするサーバーからアクセス可能なことを検査する方法について説明します。

手順

1. オプション: WebSphere Process Server が z/OS サーバーにインストール済みである必要があります。
2. Business Process Choreographer データベースのすべてのスクリプト・ファイルを、データベースをホストする z/OS サーバーにコピーします。 このスクリプトは以下のディレクトリーにあります。
   - 製品メディア上のロケーション: *media_root* または *extract_directory*/dbscripts/ProcessChoreographer/*database_type*/
   - インストール後のロケーション:*install_root*/dbscripts/ProcessChoreographer/*database_type*/

   ここで、*database_type* は以下のいずれかになります。
   - DB2zOSV8

- DB2zOSV9

3. 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. データベースおよびストレージ・グループを作成します。 以下のいずれかを実行します。
      - DB2 管理メニューを使用して新しいデータベースおよびストレージ・グループを作成します。
      - 163 ページの『BPEDB データベースの計画』で計画した値を使用し、ヘッダーの指示に従って createDatabase.sql スクリプト・ファイルのコピーを編集し、スクリプトを実行します。スクリプトを実行するには、以下のコマンドを入力します。

        ```
        db2 -tf createDatabase.sql
        ```

   e. WebSphere Process Server を実行しているリモート・サーバーから、どのユーザー ID を使用してデータベースに接続するかを決定します。通常ではセキュリティー上の理由から、このユーザー ID はデータベースの作成に使用したユーザー ID ではありません。
   f. 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ための権限をユーザー ID に付与します。このユーザー ID には、データベース用新規テーブルを作成する権限も必要です。
   g. 接続したユーザー ID のスキーマでテーブルおよびビューを作成するか、またはスキーマ修飾子をカスタマイズするかを決定します。単一のユーザー ID が、同じ名前を持つテーブルがある複数のデータベースにアクセスする場合、名前の競合を回避するために異なるスキーマ修飾子を使用する必要があります。
   h. 163 ページの『BPEDB データベースの計画』で計画した内容、およびヘッダーの指示に従って、createTablespace.sql テーブル・スペース作成スクリプトのコピーをカスタマイズします。 @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。
   i. テーブル・スペース作成スクリプトのカスタマイズ・バージョンを実行します。 例えばスクリプトを実行するには、以下のコマンドを入力します。

      ```
      db2 -tf createTablespace.sql
      ```

      テーブル・スペースを除去する場合は、dropTablespace.sql スクリプトをカスタマイズし実行します。
   j. 163 ページの『BPEDB データベースの計画』で計画した内容、およびヘッダーの指示に従って、createSchema.sql スキーマ作成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) @SCHEMA@ をスキーマ修飾子で置き換えるか、@SCHEMA@ (後続のドットも含む) をスクリプトから除去します。カスタムのスキーマ修飾子は DB2 Universal JDBC ドライバーと一緒にのみ使用できます。

k. スキーマ作成スクリプトのカスタマイズ・バージョンを実行します。 例えばスクリプトを実行するには、以下のコマンドを入力します。

```
db2 -tf createSchema.sql
```

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したい場合、dropSchema.sql スクリプトを使用してスキーマを除去しますが、スクリプトを実行する前に @SCHEMA@ を置換します。

タスクの結果

Business Process Choreographer のためのデータベースが存在します。

次のタスク

SQL 定義が提供されています。これを DB2 環境へ手動で追加する必要があります。

担当者ディレクトリー・プロバイダーの構成

Business Process Choreographer では、4 つの担当ディレクトリー・プロバイダーのいずれかを使用して、誰がプロセスを開始できるか、または誰がアクティビティーやタスクを要求できるかを決定できます。 2 つのプロバイダーは、デフォルトの構成 (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、WebSphere Application Server のユーザー・レジストリー) で使用できます。Virtual Member Manager 担当者ディレクトリー・プロバイダーおよび LDAP 担当者ディレクトリー・プロバイダーは、通常はデフォルトの構成で使用できますが、場合によっては構成作業が必要になることもあります。

このタスクについて

サポートされている担当者ディレクトリー・サービスの各タイプには、対応する担当者ディレクトリー・プロバイダー・プラグインが必要です。表 27 に、サポートされている担当者ディレクトリー・プロバイダーおよびそのプラグイン (デフォルトでインストールされる) を示します。

表 27. サポートされている担当者ディレクトリー・プロバイダーおよびそのプラグイン

担当者ディレクトリー・プロバイダー	担当者ディレクトリー・プロバイダー・プラグイン名
Virtual Member Manager (VMM)	VMM people directory provider
Lightweight Directory Access Protocol (LDAP) ディレクトリー	LDAP people directory provider
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	System people directory provider
WebSphere Application Server のユーザー・レジストリー	User Registry people directory provider

- VMM および LDAP を使用するには、多くの場合、使用前に構成をカスタマイズする必要が生じます。その方法については、後のトピックで説明します。
- ユーザー・レジストリーおよびシステム・プラグインはデフォルト構成で使用できます。これらの担当者ディレクトリー・プロバイダーはカスタマイズしないで使用できるため、多くの場合、開発環境およびテスト環境に適しています。

Virtual Member Manager 担当者ディレクトリー・プロバイダーの構成

Business Process Choreographer で担当者割り当て (誰がプロセスを開始できるか、あるいは誰がアクティビティーやタスクを要求できるかを決定する作業) を実行できるように、Virtual Member Manager (VMM) 担当者ディレクトリー・プロバイダーを構成します。VMM 担当者ディレクトリー・プロバイダーのデフォルトの構成は、そのまま使用できます。構成を行う必要があるのは、カスタムの担当者割り当て基準を導入する場合のみです。

始める前に

VMM 担当者ディレクトリー・プロバイダーを構成するには、フェデレーテッド・リポジトリーの構成を完了しておく必要があります。

手順

1. VMM 用の標準変換ファイルをコピーし、それに別の名前 (例えば `myVMMTransformation.xsl`) を付けます。 VMM 用の標準の XSL 変換は、*install_root*`/ProcessChoreographer/Staff/VMMTransformation.xsl` にあります。
2. 227 ページの『LDAP 変換ファイルの適合』の説明に従って、組織のスキーマに合うように変換ファイルのコピーを修正します。

   注意:
   変換ファイルのオリジナル版は変更しないでください。このファイルは、サービス・パックやフィックスパックを適用するときに、警告なしに上書きされる可能性があるからです。
3. Business Process Choreographer がクラスター上に構成されている場合は、変換ファイルのコピーを `ProcessChoreographer/Staff` ディレクトリーに格納して、そのクラスターのメンバーをホストしている各 WebSphere Process Server インストール済み環境で使用できるようにします。
4. 管理コンソールで、**「リソース」 → 「担当者ディレクトリー・プロバイダー」** をクリックします。
5. 以下のリストから該当するノードを選択します。

オプション	説明
スタンドアロン・プロファイルの場合	1 つのノードしか表示されません。
Business Process Choreographer が 1 台のサーバー上に構成されている Network Deployment 環境	このサーバーを含むノードを選択します。

オプション	説明
Business Process Choreographer がクラスター上に構成されている Network Deployment 環境	クラスターのメンバーをホストするすべてのノードで、担当者ディレクトリー・プロバイダーを構成する必要があります (ステップ 6 を実行します)。最初のノードを選択し、そのノードで担当者ディレクトリー・プロバイダーを構成して、次に、クラスター・メンバーをホストする他のすべてのノードについてこの構成 (ステップ 6) を繰り返します。

6. 新規 VMM 担当者ディレクトリー構成を作成するには、以下のようにします。
   a. 「**VMM 担当者ディレクトリー・プロバイダー (VMM People Directory Provider)**」をクリックします。
   b. 「**追加プロパティー**」で、「**担当者ディレクトリー構成**」を選択します。
   c. 「**新規**」 → 「**参照**」をクリックして、ステップ 2 (224 ページ)で修正した Extensible Stylesheet Language (XSL) 変換ファイルのコピーを選択します。ノード・エージェントが稼働している場合は、リモート・ノードのファイル・システムを参照してファイルを選択できます。
   d. 「**次へ**」をクリックして、選択したノード上の `ProcessChoreographer/Staff` ディレクトリーにファイルをコピーします。
   e. 新しい担当者ディレクトリー構成の管理名および (オプションで) 説明を入力します。
   f. この構成をシステムに識別させる固有の Java Naming and Directory Interface (JNDI) 名 (例えば `bpe/staff/myvmmconfiguration`) を入力します。

      注: 他に構成パラメーターはありません。
   g. 「**OK**」をクリックしてから「**保管**」をクリックします。
7. プロバイダー構成を活動化するには、プロバイダーを構成したサーバーを停止してから始動します。

タスクの結果

VMM 担当者ディレクトリー・プロバイダーが構成されました。この手順で問題が発生した場合は、「*Troubleshooting WebSphere Process Server*」(PDF) を参照してください。

LDAP 担当者ディレクトリー・プロバイダーの構成

Business Process Choreographer で担当者割り当て (誰がプロセスを開始できるか、あるいは誰がアクティビティーやタスクを要求できるかを決定する作業) を実行できるように、Lightweight Directory Access Protocol (LDAP) 担当者ディレクトリー・プロバイダーを構成します。

始める前に

LDAP を構成するには、 175 ページの『担当者ディレクトリー・プロバイダーの計画』の説明に従って、LDAP の計画を完了している必要があります。

このタスクについて

LDAP 担当者ディレクトリー・プロバイダー構成は、ローカルの LDAP サーバーを指す URL で初期設定されています。この URL は後で、実際の LDAP サーバー (通常、アプリケーション・サーバーに対してはリモートになっている) を指すよう変更する必要があります。LDAP 担当者ディレクトリー・プロバイダーは、匿名アクセスを許可している LDAP サーバーに対して構成されます。

手順

1. LDAP 用の標準変換ファイルをコピーし、それに別の名前 (例えば myLDAPTransformation.xsl) を付けます。 LDAP 用の標準の XSL 変換は、*install_root*/ProcessChoreographer/Staff/LDAPTransformation.xsl 内にあります。
2. 227 ページの『LDAP 変換ファイルの適合』の説明に従って、組織のスキーマに合うように変換ファイルのコピーを修正します。

   注意:
   変換ファイルのオリジナル版は変更しないでください。このファイルは、サービス・パックやフィックスパックを適用するときに、警告なしに上書きされる可能性があるからです。
3. Business Process Choreographer がクラスター上に構成されている場合は、変換ファイルのコピーを ProcessChoreographer/Staff ディレクトリーに格納して、そのクラスターのメンバーをホストしている各 WebSphere Process Server インストール済み環境で使用できるようにします。
4. 管理コンソールで、「**リソース**」 → 「**担当者ディレクトリー・プロバイダー**」をクリックします。
5. 以下のリストから該当するノードを選択します。

オプション	説明
スタンドアロン・プロファイルの場合	1 つのノードしか表示されません。
Business Process Choreographer が 1 台のサーバー上に構成されている Network Deployment 環境	このサーバーを含むノードを選択します。
Business Process Choreographer がクラスター上に構成されている Network Deployment 環境	クラスターのメンバーをホストするすべてのノードで、担当者ディレクトリー・プロバイダーを構成する必要があります (ステップ 6 を実行します)。最初のノードを選択し、そのノードで担当者ディレクトリー・プロバイダーを構成して、次に、クラスター・メンバーをホストする他のすべてのノードについてこの構成 (ステップ 6) を繰り返します。

6. 選択したノードに新規 LDAP 構成を作成するには、以下のようにします。
   a. 「**LDAP 担当者ディレクトリー・プロバイダー (LDAP People Directory Provider)**」をクリックします。
   b. 「**追加プロパティー**」で、「**担当者ディレクトリー構成**」をクリックします。

c. 「**新規**」 → 「**参照**」をクリックして、ステップ 2 (226 ページ)で修正した Extensible Stylesheet Language (XSL) 変換ファイルのコピーを選択します。ノード・エージェントが稼働している場合は、リモート・ノードのファイル・システムを参照してファイルを選択できます。

d. 「**次へ**」をクリックして、選択したノード上の `ProcessChoreographer¥Staff` ディレクトリーにファイルをコピーします。

e. 新しい担当者ディレクトリー構成の管理名および (オプションで) 説明を入力します。

f. このプロバイダーをヒューマン・タスクで参照するのに使用する固有の Java Naming and Directory Interface (JNDI) 名を入力します。例えば `bpe/staff/ldapserver1` と入力します。

g. 「**適用**」をクリックし、「**カスタム・プロパティー**」をクリックします。

h. ステップ 2 (176 ページ) で計画した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。 オプションの追加プロパティーについては、JNDI に定義されているプロパティーを設定できます。例えば、LDAP 参照を使用可能にするために、名前が `java.naming.referral` で値が `follow` の追加プロパティーを作成できます。

   providerURL に対しては、`ldap://` または `ldaps://` で始まる URL を指定できます。高可用性を実現するために、複数の LDAP サーバーにミラーリングされたデータが格納されている場合は、各 LDAP の URL をスペース文字で区切って入力します。

i. 変更を適用するには、「**保管**」をクリックします。

7. プロバイダー構成を活動化するには、プロバイダーを構成したサーバーを停止してから始動します。

8. これらのステップのいずれかで問題が発生する場合は、「*Troubleshooting WebSphere Process Server*」(PDF) を参照してください。

タスクの結果

ヒューマン・タスクおよびプロセスは、担当者割り当てサービスを使用して担当者割り当て照会を解決し、どの担当者がどのアクティビティーを実行できるのかを判別できるようになりました。

LDAP 変換ファイルの適合

LDAP 変換 XSL ファイルを組織の 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、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、事前定義の担当者割り当て基準をマップします。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の識別名 (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。

前述の名前と異なるオブジェクト・クラス名および属性名を LDAP スキーマで使用している場合は、以下の手順を実行してください。

1. LDAP 用の標準変換ファイルをコピーし、それに別の名前 (例えば myLDAPTransformation.xsl) を付けます。LDAP 用の標準の XSL 変換は、*install_root*/ProcessChoreographer/Staff/LDAPTransformation.xsl 内にあります。
2. コピーしたファイル内で、ご使用の LDAP スキーマで使用されている名前と一致するように、オブジェクト・クラスの名前および属性名を変更します。多くの場合、このファイルの変数宣言部分を編集することにより、すべての担当者割り当て基準の設定を変更できます。

```
<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>
```

注意:
標準変換ファイルのオリジナル版は変更しないでください。このファイルは、サービス・パックやフィックスパックを適用するときに、警告なしに上書きされる可能性があるからです。

以下の例に示すように、個々の担当者割り当て基準を変換する XSL テンプレート内で変更を適用できます。

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:resultObject>
<xsl:attribute name="objectclass">groupOfUniqueNames</xsl:attribute>
<xsl:attribute name="usage">recursive</xsl:attribute>
<sldap:resultAttribute>
<xsl:attribute name="name">uniqueMember</xsl:attribute>
<xsl:attribute name="destination">intermediate</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>

<sldap:resultObject>
<xsl:attribute name="objectclass"><xsl:value-of select="$DefaultPersonClass"/></xsl:attribute>
<xsl:attribute name="usage">simple</xsl:attribute>
```

```
<sldap:resultAttribute>
<xsl:attribute name="name">cn</xsl:attribute>
<xsl:attribute name="destination">userID</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultMailAttribute"/></xsl:attribute>
<xsl:attribute name="destination">eMailAddress</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultLocaleAttribute"/></xsl:attribute>
<xsl:attribute name="destination">preferredLocale</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>

</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">
(&
...
<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:resultObject>
<xsl:attribute name="objectclass">mypersonclass</xsl:attribute>
<xsl:attribute name="usage">simple</xsl:attribute>

<sldap:resultAttribute>
<xsl:attribute name="name">myuid</xsl:attribute>
<xsl:attribute name="destination">userID</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
```

```
<xsl:attribute name="name"><xsl:value-of select="$DefaultMailAttribute"/></xsl:attribute>
<xsl:attribute name="destination">eMailAddress</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultLocaleAttribute"/></xsl:attribute>
<xsl:attribute name="destination">preferredLocale</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>

<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:resultObject>
<xsl:attribute name="objectclass"><xsl:value-of select="$DefaultPersonClass"/></xsl:attribute>
<xsl:attribute name="usage">simple</xsl:attribute>

<sldap:resultAttribute>
<xsl:attribute name="name">managerentry</xsl:attribute>
<xsl:attribute name="destination">intermediate</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>
</sldap:user>
</sldap:intermediateResult>

<sldap:user>
...
<xsl:attribute name="name">name</xsl:attribute>
...
<sldap:resultObject>
<xsl:attribute name="objectclass"><xsl:value-of select="$DefaultPersonClass"/></xsl:attribute>
<xsl:attribute name="usage">simple</xsl:attribute>

<sldap:resultAttribute>
<xsl:attribute name="name">name</xsl:attribute>
<xsl:attribute name="destination">userID</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultMailAttribute"/></xsl:attribute>
<xsl:attribute name="destination">eMailAddress</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultLocaleAttribute"/></xsl:attribute>
<xsl:attribute name="destination">preferredLocale</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>

</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">
(&
...
<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:resultObject>
<xsl:attribute name="objectclass">mypersonclass</xsl:attribute>
<xsl:attribute name="usage">simple</xsl:attribute>

<sldap:resultAttribute>
<xsl:attribute name="name">myuid</xsl:attribute>
<xsl:attribute name="destination">userID</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultMailAttribute"/></xsl:attribute>
<xsl:attribute name="destination">eMailAddress</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultLocaleAttribute"/></xsl:attribute>
<xsl:attribute name="destination">preferredLocale</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>
</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:resultObject>
<xsl:attribute name="objectclass">mypersonclass</xsl:attribute>
<xsl:attribute name="usage">simple</xsl:attribute>

<sldap:resultAttribute>
<xsl:attribute name="name">myuid</xsl:attribute>
<xsl:attribute name="destination">userID</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultMailAttribute"/></xsl:attribute>
<xsl:attribute name="destination">eMailAddress</xsl:attribute>
</sldap:resultAttribute>
<sldap:resultAttribute>
<xsl:attribute name="name"><xsl:value-of select="$DefaultLocaleAttribute"/></xsl:attribute>
<xsl:attribute name="destination">preferredLocale</xsl:attribute>
</sldap:resultAttribute>
</sldap:resultObject>

</sldap:user>
```

担当者の代替の構成

このトピックでは、Business Process Choreographer の担当者の代替を構成する方法について説明します。

始める前に

フェデレーテッド・リポジトリーに WebSphere セキュリティーを構成しました。カスタムの担当者割り当て基準を導入する場合は、 224 ページの『Virtual Member Manager 担当者ディレクトリー・プロバイダーの構成』も実行しました。プロパテ

ィー拡張の保管にファイル・レジストリー、プロパティー拡張レジストリー、または既存の Lightweight Directory Access Protocol (LDAP) スキーマのどれを使用するのかが明確になっています。

このタスクについて

実稼働環境で担当者の代替を使用するには、このトピックでの説明に従って、Virtual Member Manager (VMM) プロパティー拡張リポジトリーを使用する必要があります。ただし、シングル・サーバーのテスト環境でのみ担当者の代替を使用する場合は、VMM を構成することなく、デフォルトでフェデレーテッド・リポジトリーに関連付けられているファイル・レジストリーを使用できます。

手順

1. 属性 「isAbsent」、「substitutes」、「substitutionStartDate」、および「substitutionEndDate」を PersonAccount の VMM 定義に追加します。
   a. wimxmlextension.xml ファイルを以下のようにして見つけます。*profile_root*/config/cells/*cell_name*/wim/model にあります。
   b. wimxmlextension.xml ファイルのバックアップ・コピーを作成します。
   c. wimxmlextension.xml ファイルのオリジナル・コピーを編集し、このファイルに以下の定義が含まれていることを確認します。これらの定義では、ユーザー代替に必要な 2 つの属性が、PersonAccount エンティティー・タイプに追加されます。

```
<wim:propertySchema nsURI="http://www.ibm.com/websphere/wim"
     dataType="STRING" multiValued="false" propertyName="isAbsent">
   <wim:applicableEntityTypeNames>PersonAccount
   </wim:applicableEntityTypeNames>
</wim:propertySchema>

<wim:propertySchema nsURI="http://www.ibm.com/websphere/wim"
     dataType="STRING" multiValued="true" propertyName="substitutes">
   <wim:applicableEntityTypeNames>PersonAccount
   </wim:applicableEntityTypeNames>
</wim:propertySchema>
<wim:propertySchema nsURI="http://www.ibm.com/websphere/wim"
     dataType="STRING" multiValued="false" propertyName="substitutionStartDate">
   <wim:applicableEntityTypeNames>PersonAccount
   </wim:applicableEntityTypeNames>
</wim:propertySchema>

<wim:propertySchema nsURI="http://www.ibm.com/websphere/wim"
     dataType="STRING" multiValued="false" propertyName="substitutionEndDate">
   <wim:applicableEntityTypeNames>PersonAccount
   </wim:applicableEntityTypeNames>
</wim:propertySchema>
```

   ファイル・レジストリー fileRegistry.xml を使用している場合は、ステップ 4 (234 ページ) までスキップしてください。
2. プロパティー拡張リポジトリーをセットアップします。 プロパティー拡張リポジトリーのセットアップについて詳しくは、『フェデレーテッド・リポジトリー構成におけるプロパティー拡張リポジトリーの構成』を参照してください。
   a. データベースがプロパティー拡張の保管に使用できることを確認します。
   b. JDBC ドライバー・クラスがサーバーのクラス・パスで使用可能になっていることを確認します。 「**環境**」 → 「**WebSphere 変数**」をクリックして確認します。必要に応じて、クラスパスに JDBC ドライバーを追加します。DB2 の場合は、db2jcc.jar,db2jcc_license_cu.jar および db2jcc_license_cisuz.jar をサーバーのクラス・パスに追加し、「**適用**」 → 「**保管**」をクリックします。

c. 管理コンソールを使用して、VMM 用に DB2 Universal JDBC ドライバー・プロバイダーおよびタイプ 4 のデータ・ソースを構成します。 データ・ソースの webSphereDefaultIsolationLevel カスタム・プロパティーの値を 2 に設定します。デフォルトの分離レベルの変更について詳しくは、『Changing the default isolation level for non-CMP applications and describing how to do so using a new custom property webSphereDefaultIsolationLevel』を参照してください。

d. サーバーを再始動します。

e. wimlaproperties.xml ファイルのバックアップ・コピーを作成します。 これは、*install_root*/etc/wim/setup にあります。

f. wimlaproperties.xml ファイルのオリジナル・コピーを編集して、以下の定義を追加します。

```
<wimprop:property wimPropertyName="isAbsent" dataType="String"
     valueLength="128" multiValued="false">
    <wimprop:applicableEntityName>
  <wimprop:entityName>PersonAccount</wimprop:entityName>
    </wimprop:applicableEntityName>
</wimprop:property>

<wimprop:property wimPropertyName="substitutes" dataType="String"
     valueLength="128" multiValued="true">
    <wimprop:applicableEntityName>
  <wimprop:entityName>PersonAccount</wimprop:entityName>
    </wimprop:applicableEntityName>
</wimprop:property>
<wimprop:property wimPropertyName="substitutionStartDate" dataType="String"
     valueLength="128" multiValued="false">
    <wimprop:applicableEntityName>
  <wimprop:entityName>PersonAccount</wimprop:entityName>
    </wimprop:applicableEntityName>
</wimprop:property>

<wimprop:property wimPropertyName="substitutionEndDate" dataType="String"
     valueLength="128" multiValued="false">
    <wimprop:applicableEntityName>
  <wimprop:entityName>PersonAccount</wimprop:entityName>
    </wimprop:applicableEntityName>
</wimprop:property>
```

g. アプリケーション・サーバー (または Network Deployment 環境では、デプロイメント・マネージャー) が稼働していることを確認します。wsadmin ユーティリティーには -conntype NONE オプションを使用しないよう注意してください。

h. VMM 管理タスク setupIdMgrPropertyExtensionRepositoryTables を使用して、プロパティー拡張リポジトリー・データベースに代替プロパティーを作成します。詳細については、『wsadmin コマンドを使用した、エントリー・マッピング・リポジトリー、プロパティー拡張機能リポジトリー、またはカスタム・レジストリー・データベース・リポジトリーの設定』を参照してください。

i. Lightweight Directory Access Protocol (LDAP) ユーザー・リポジトリーを使用する場合は、 wimconfig.xml ファイルを見つけます。 ファイルを編集し、以下のエントリーを追加して、LDAP リポジトリーから代替属性を除外します。

```
<config:repositories xsi:type="config:LdapRepositoryType"
      adapterClassName="com.ibm.ws.wim.adapter.ldap.LdapAdapter"
      id="ldaprepo1" ...>
      ...
      <config:attributeConfiguration>
         <config:propertiesNotSupported name="isAbsent"/>
         <config:propertiesNotSupported name="substitutes"/>
         <config:propertiesNotSupported name="substitutionEndDate"/>
         <config:propertiesNotSupported name="substitutionStartDate"/>
      </config:attributeConfiguration>
```

j. 拡張プロパティー・リポジトリーを活動化します。

1) setIdMgrPropertyExtensionRepository コマンドを使用します。詳しくは、wsadmin コマンドを使用した、エントリー・マッピング・リポジトリー、プロパティー拡張機能リポジトリー、またはカスタム・レジストリー・データベース・リポジトリーの設定を参照してください。例えば、VMMDB という名前の DB2 データベース、および VMMDS という名前のデータ・ソースを使用します。

```
$AdminTask setIdMgrPropertyExtensionRepository {
-dataSourceName jdbc/VMMDS
-databaseType db2
-dbURL jdbc:DB2://host:port/VMMDB
-dbAdminId userID
 -dbAdminPassword password
 -JDBCDriverClass com.ibm.db2.jcc.DB2Driver
-entityRetrievalLimit 10 }
```

2) wimconfig.xml ファイルに、以下のような項目が含まれていることを確認します。

```
<config:propertyExtensionRepository
   adapterClassName="com.ibm.ws.wim.lookaside.LookasideAdapter"
   id="LA"
   databaseType="db2"
   dataSourceName="jdbc/VMMDS"
   dbAdminId="userID"
   dbAdminPassword="{xor}PasswordXOR"
   dbURL="jdbc:DB2://host:port/VMMDB"
   entityRetrievalLimit="10"
   JDBCDriverClass="com.ibm.db2.jcc.DB2Driver"/>>
```

3. LDAP スキーマを使用して代替情報を保持している場合: LDAP スキーマには、既に (おそらくは異なる名前の) 「isAbsent」、 「substitutes」、および「substitutionStartDate」、および「substitutionEndDate」の定義が存在している可能性も、そうでない可能性もあります。既存の定義が存在しているか、新規の定義を作成するかには関係なく、以下を確認してください。

   a. LDAP ディレクトリーが書き込み操作を許可している必要があります。

   b. 不在情報の属性 (「isAbsent」) のタイプは、Boolean または String である必要があります。

   c. 代理人になることができるユーザーを定義している属性 (「substitutes」) は String タイプおよび多値で、最大許容文字数は 128 文字である必要があります。

   d. 既存の、または選択した属性名が「isAbsent」、 「substitutes」、および「substitutionStartDate」、「substitutionEndDate」でない場合は、管理コンソールで属性名を定義する必要があります。それには、**「サーバー」 → 「クラスター」 → 「WebSphere Application Server クラスター」 → 「クラスター名」**または**「サーバー」 → 「サーバー・タイプ」 → 「WebSphere Application Server」 → 「サーバー名」**のいずれかをクリック後、**「構成」**タブの**「ビジネス・インテグレーション」**セクションで、**「Business Process Choreographer」**を展開し、**「Human Task Manager」**をクリックし、**「構成」**タブで**「カスタム・プロパティー」**を選択して、カスタム・プロパティー Substitution.SubstitutesAttribute、Substitution.AbsenceAttribute、 Substitution.StartDateAttribute、および Substitution.EndDateAttribute に任意の名前を設定します。

4. サーバーを再始動します。

5. Human Task Manager で代替を使用可能にします。

a. 「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」を選択した後、「**Human Task Manager**」、続いて、「**ランタイム**」または「**構成**」をクリックします。

b. 代替を使用可能にするには、「**置換を使用可能にする**」を選択します。

c. 他のユーザーに対する代替の実行を管理者以外に許可する場合は、「**置換管理を管理者に制限**」のチェックを外します。

注: この設定は、ユーザーが自分自身に対する代替を変更する機能には影響しません。

d. 「**適用**」をクリックします。

e. ステップ 5a で「**構成**」を選択した場合は、サーバーを再始動して、代替の設定をアクティブにします。

6. これらのステップのいずれかで問題が発生する場合は、「*Troubleshooting WebSphere Process Server*」(PDF) を参照してください。

タスクの結果

不在ユーザーに対するユーザー代替をサポートするよう担当者割り当てサービスが構成されました。

Business Process Choreographer Explorer の構成

スクリプトを実行するか、または管理コンソールを使用して、Business Process Choreographer Explorer を構成できます。

始める前に

Business Process Choreographer が構成済みです。

このタスクについて

以下の 1 つ以上が該当します。

- 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 構成を管理しようとしています。
- 既に管理されている Business Process Choreographer 構成に 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。
- オプションの Business Process Choreographer Explorer レポート作成機能 (これは、以前のバージョンで Business Process Choreographer Observer という名称で提供されていました) を構成します。

Business Process Choreographer Explorer を構成するには、以下のいずれかを実行します。

手順

1. スクリプトを使用する場合は、 237 ページの『clientconfig.jacl スクリプト・ファイルを使用した Business Process Choreographer Explorer の構成』を実行します。
2. 管理コンソールを使用する場合は、『管理コンソールを使用した Business Process Choreographer Explorer の構成』を実行します。

タスクの結果

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

管理コンソールを使用した Business Process Choreographer Explorer の構成

管理コンソールを使用して、Business Process Choreographer Explorer およびオプションの Business Process Choreographer Explorer レポート作成機能を構成できます。

手順

1. 「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Business Process Choreographer Explorer**」をクリックします。
2. 新規のインスタンスを構成するには、「**追加**」をクリックします。
3. 以下のフィールドに値を入力します。
   - サーバーの始動時に新しいインスタンスを自動的に始動する場合は、「**自動開始を使用可能にする**」を選択します。
   - 「**コンテキスト・ルート**」はデプロイメント・ターゲット・サーバーまたはクラスター上で固有のものである必要があります。
   - **Explorer による検索結果の制限**。
   - **管理対象の Business Process Choreographer コンテナー**。
   - 「**Business Flow Manager REST API URL**」。スタンドアロン・サーバーの場合、サーバーの Web コンテナーを指すデフォルト値が用意されています。
   - 「**Human Task Manager REST API URL**」。スタンドアロン・サーバーの場合、サーバーの Web コンテナーを指すデフォルト値が用意されています。
4. オプション: Business Process Choreographer Explorer レポート作成機能を構成する場合は、以下を実行します。
   a. Business Process Choreographer イベント・コントローラーがインストールされ、構成されていることを確認します。
   b. 「**レポート作成機能の有効化**」を選択します。
   c. 視覚化する Business Process Choreographer Event Collector を選択します。リストが空の場合は、最初に Business Process Choreographer イベント・コントローラーをインストールして、構成する必要があります。詳しくは、 241 ページの

ページの『Business Process Choreographer Explorer レポート作成機能および Event Collector の構成』を参照してください。

d. 「**スナップショット範囲でレポートする**」で、何日分のデータを視覚化するのかを指定します。

5. 「**適用**」をクリックします。 進行状況を示すメッセージが表示されます。
6. オプション: 問題が報告されている場合は、SystemOut.log ファイルを確認します。
7. BPCExplorer_*scope* という名前のエンタープライズ・アプリケーションを開始します。 *scope* は、Business Process Choreographer Explorer を構成するサーバーまたはクラスターを示します。

タスクの結果

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

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

このスクリプト・ファイルは、Business Process Choreographer Explorer と、サーバーまたはクラスター上の必要なすべてのリソースを構成します。そのスクリプト・ファイルを使用して、既存のインスタンスの構成変更を変更することも可能です。例えば、maxListEntries を変更したり、Business Process Choreographer Explorer レポート作成機能を構成したりすることができます。

目的

このスクリプト・ファイルは、Business Process Choreographer Explorer を構成します。このスクリプト・ファイルは、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。

場所

clientconfig.jacl スクリプト・ファイルは、以下の Business Process Choreographer の config ディレクトリーにあります。

- *smpe_root*/ProcessChoreographer/config

Network Deployment 環境でのスクリプトの実行

構成スクリプトは、 wsadmin コマンドで実行されます。 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。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にコンフィギュレーター権限も管理者権限もない場合は、wsadmin -user および -password のオプションを付けて、コンフィギュレーターまたは管理者の権限を持つユーザー ID を指定します。

Business Process Choreographer Explorer の非対話式での構成

現行ディレクトリーを *install_root* に変更し、以下を実行します。

次のコマンドを入力します。

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

parameters は次のとおりです。

```
    ( [-node nodeName][-server serverName] )
      [-cluster clusterName]
      [-contextRootExplorer explorerContextRoot]
      [-explorerHost explorerURL]
      [-hostName explorerVirtualHostname]
      [-precompileJSPs { yes | no }]
  ( ( [-remoteNode nodeName][-remoteServer serverName] )
          | [-remoteCluster clusterName] )
      [-maxListEntries maximum]
       [-reportAtSnapshotRange number]
       [-reportCreateTables { true | false }]
       -reportDataSource jndiName
       [-reportFunction { yes | no }]
       [-reportSchemaName schemaName]       -restAPIBFM restAPIURL
      -restAPIHTM restAPIURL
```

注: このスクリプトをバッチ・モードで実行する場合は、すべての必須パラメーターが含まれていなければなりません。このスクリプトを対話式に実行する場合は、コマンド行で指定されていない必須パラメーターがあると、プロンプトが表示されます。

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。

-cluster *clusterName*

clusterName は、Business Process Choreographer Explorer を構成するクラスターの名前です。このパラメーターはオプションです。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。

-contextRootExplorer *contextRootExplorer*

ここで *contextRootExplorer* は、Business Process Choreographer Explorer のコンテキスト・ルートです。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。デフォルト値は `/bpc` です。

-explorerHost *explorerURL*

ここで *explorerURL* は、Business Process Choreographer Explorer の URL です。このパラメーターの値は、管理対象 Business Process Choreographer 構成の Human Task Manager を、この特定の Business Process Choreographer Explorer インスタンスにリンクするときに使用されます。バッチ・モードでは、このパラメーターのデフォルト値は空ストリングであり、したがってリンクは作成されません。このリンクは、管理コンソールを使用して後でいつでも作成または変更することができます。

-hostName *VirtualHostname*

VirtualHostname は、Business Process Choreographer Explorer が実行される仮想ホストです。デフォルト値は `default_host` です。

-maxListEntries *maximum*

maximum は、照会に対し Business Process Choreographer Explorer から戻される結果の最大数です。デフォルトは 10000 です。

-node *nodeName*

nodeName は、Business Process Choreographer Explorer を構成するノードの名前です。このパラメーターを指定しないと、デフォルトはローカル・ノードになります。

-precompileJSPs { no | yes }

Java Server Pages (JSPs) をプリコンパイルするかどうかを識別します。デフォルトは no です。プリコンパイル済み JSP はデバッグできない点に注意してください。

-remoteCluster *clusterName*

ローカル側の Business Process Choreographer 構成に接続せず、remoteNode および remoteServer を指定しない場合は、このパラメーターを使用します。このパラメーターが指定されていない場合、デフォルトで -cluster パラメーターの値が使用されます。

-remoteNode *nodeName*

ローカル側の Business Process Choreographer 構成に接続しない場合は、このパラメーターと remoteServer を使用します。このパラメーターが指定されていない場合、デフォルトで -node パラメーターの値が使用されます。

-remoteServer *serverName*

ローカル側の Business Process Choreographer 構成に接続しない場合は、このパラメーターと remoteNode を使用します。このパラメーターが指定されていない場合、デフォルトで -server パラメーターの値が使用されます。

-reportAtSnapshotRange *number*

スナップショット・レポートは、スナップショットの限定日時より古いすべてのイベントを評価することによって作成されます。このオプション・パラメーターは、スナップショット・レポートにイベントを組み込むことができる日数を定義します。この期間内に発行されたイベントのみがスナップショット・レポートで評価されます。デフォルトは 60 日です。このオプション・パラメーターが有効となるのは、`-reportFunction yes` オプションでレポート作成機能が有効になっている場合に限られます。

この値が高すぎると、大量のイベントを処理する必要が生じ、レポート生成に長い時間がかかる可能性があります。この値は、ビジネス環境におけるプロセス・インスタンスの最大期間に設定するようにしてください。

-reportCreateTables { true | false }

このオプション・パラメーターでは、Business Process Choreographer Explorer が初めてデータベースに接続する時点で、Business Process Choreographer Explorer レポート作成機能スキーマを作成するかどうかを指定します。デフォルトは true です。このオプション・パラメーターが有効となるのは、`-reportFunction yes` オプションでレポート作成機能が有効になっている場合に限られます。

-reportDataSource *jndiName*

jndiName は、データベースへの接続に使用されるデータ・ソース JNDI の JNDI 名です。`-reportFunction yes` を指定した場合は、このパラメーターは必須です。データ・ソースは自動では作成されません。

-reportFunction { yes | no }

このオプション・パラメーターでは、Business Process Choreographer Explorer のレポート作成機能を有効にするかどうかを制御します。対話モードでのデフォルトは no です。バッチ・モード場合は、以前のバージョンとの互換性のためにデフォルトは yes になっています。

-reportSchemaName *schemaName*

このオプション・パラメーターでは、すべてのレポート・データベース・オブジェクトのプレフィックスとして使用するデータベース・スキーマを指定します。スキーマ名を指定しない場合は、固有のスキーマ名が生成されます。このオプション・パラメーターが有効となるのは、`-reportFunction yes` オプションでレポート作成機能が有効になっている場合に限られます。デフォルト値は WPRBC00 です。

-restAPIBFM *restAPIURL*

ここで、*restAPIURL* は Business Flow Manager REST API の URL で、Business Process Choreographer Explorer のグラフィカル・プロセス・ウィジェットをサポートするために必要です。スタンドアロン・サーバーでは、デフォルトが計算されます (`http://`*`localhost`*`:9080/rest/bpm/bfm` など)。 Network Deployment 環境では、デフォルト値はありません。

-restAPIHTM *restAPIURL*

ここで、*restAPIURL* は Human Task Manager REST API の URL で、Business Process Choreographer Explorer のグラフィカル・プロセス・ウィジェットをサポートするために必要です。スタンドアロン・サーバーでは、デフォルトが計算されます (`http://`*`localhost`*`:9080/rest/bpm/htm` など)。 Network Deployment 環境では、デフォルト値はありません。

-server *serverName*

serverName は、Business Process Choreographer Explorer 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。

ログ・ファイル

clientconfig.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。

- clientconfig.log
- wsadmin.traceout

どちらのファイルも、ユーザーのプロファイルの logs ディレクトリー内にあります。ディレクトリー *profile_root*`/logs` 内スクリプトを接続モードで実行する場合、wsadmin スクリプト・クライアントが接続するアプリケーション・サーバーまたはデプロイメント・マネージャーに基づいて名前が付けられた `logs` ディレクトリー内のサブディレクトリーにある、ファイル `SystemOut.log` および `SystemErr.log` も検査してください。

Business Process Choreographer Explorer レポート作成機能および Event Collector の構成

Business Process Choreographer Explorer レポート作成機能の使用はオプションです。しかし、使用するには、事前にデータベースをセットアップし、アプリケーションをインストールしておく必要があります。

始める前に

179 ページの『Business Process Choreographer Explorer レポート作成機能の計画』および 235 ページの『Business Process Choreographer Explorer の構成』を実行しましたが、レポート作成機能を構成しませんでした。

このタスクについて

Business Process Choreographer Explorer レポート作成機能をその独自のデータベースと一緒に構成します。

手順

1. Business Process Choreographer 用のデータベースがまだ存在しない場合は、『レポート・データベースの準備』を実行します。
2. 259 ページの『Business Process Choreographer Event Collector アプリケーションの構成』を実行します。
3. Business Process Choreographer Explorer の構成時に、レポート作成機能を使用可能にしなかった場合は、 302 ページの『Business Process Choreographer Explorer レポート作成機能の使用可能化』を実行します。
4. 266 ページの『Business Process Choreographer Explorer レポート作成機能の構成パラメーターの変更』を実行します。
5. 301 ページの『Business Process Choreographer のロギング可能化』を実行します。
6. 274 ページの『Business Process Choreographer Explorer レポート作成機能の確認』を実行します。

タスクの結果

Business Process Choreographer Explorer レポート作成機能が構成され、稼働しています。

次のタスク

Business Process Choreographer Explorer レポート作成機能を使用して、 430 ページの『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』の説明のように、レポートを生成できます。

レポート・データベースの準備

データベースに対してアクションを実行します。

SQL スクリプトを使用したレポート・データベースの作成:

Business Process Choreographer を構成する前に、あるいは製品をインストールする前でも、Business Process Choreographer Explorer レポート作成機能のためのデータベースを手動で作成することができます。

始める前に

168 ページの『レポート・データベースの計画』を実行している。

このタスクについて

組織で、データベースを別々のデータベース管理者によって作成することが必要な場合があります。管理コンソールまたは bpeconfig.jacl スクリプトを使用して Business Process Choreographer を構成する場合、カスタマイズ済みの SQL スクリプトが作成され、これを DBA に渡して BPEDB データベースを作成することができます。ただし、Business Process Choreographer を構成する前や、あるいは製品のインストール前にデータベースを作成したい場合、DBA はカスタマイズ済みでない SQL スクリプトを使用する必要があります。このトピックでは、カスタマイズ済みでない SQL スクリプト (製品メディアから入手可能) の使用方法について説明します。

手順

1. ローカルの DB2 for z/OS データベースを使用するには、そのデータベースを手動で作成する必要があります。
2. データベースをホストするサーバーで、以下のようにします。 『USS における SQL スクリプトを使用した Business Process Choreographer Explorer レポート作成機能のための DB2 for z/OS データベースの準備』を参照してください。

タスクの結果

Business Process Choreographer Explorer レポート作成機能用のデータベースが作成されました。

Business Process Choreographer Explorer レポート作成機能のための DB2 for z/OS データベースの準備:

レポート・データベースをリモートで、または UNIX System Services 内で準備できます。

USS における SQL スクリプトを使用した Business Process Choreographer Explorer レポート作成機能のための DB2 for z/OS データベースの準備:

ここでは、スクリプトを使用してを UNIX System Services (USS) で使用して、DB2 for z/OS データベースを準備する方法について説明します。

手順

1. 以下のように DB2 環境を準備します。
   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. サブシステムのロケーション名を決定します。ロケーション名を見つけるには、DB2 システム・パネルで確認するか、またはサブシステムの DB2 管理メニュー・オプション「**SQL ステートメントの実行 (Execute SQL statements)**」を選択し、以下の SQL 照会を入力します。

```
select current server from sysibm.sysdummy1
```

e. ストレージ・グループを作成し、その名前 (例えば OBSVRSG) を書き留めます。

f. 新規データベースを使用する場合は、例えば OBSVRDB という名前の新規データベースを作成します。 必要であれば、既存のデータベースおよびストレージ・グループ (例えば、Business Process Choreographer データベースの BPEDB) を再使用することができます。

g. 使用するスキーマ修飾子を決定します。

h. データベースのセットアップに使用するユーザー ID *user_ID* を決定します。 これは、実行時にデータベースにアクセスするために使用されるユーザー ID ではありません。

i. 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ための以下の権限をユーザー ID が持っていることを確認します。
   - ストレージ・グループを使用する権限
   - データベース OBSVRDB を使用する権限
   - データベース OBSVRDB 内にテーブル・スペースを作成する権限
   - データベース OBSVRDB 内にテーブルを作成する権限

j. Java 実装の Business Process Choreographer ユーザー定義関数 (UDF) を使用する場合は、このユーザー ID が以下の権限を所有していることも確認します。
   - SYSIBM.SYSJAROBJECTS に対して選択を実行する権限
   - スキーマ SQLJ に対して以下のストアード・プロシージャーを実行する権限
     - INSTALL_JAR
     - REMOVE_JAR
     - REPLACE_JAR
     - DB2_INSTALL_JAR
     - DB2_REMOVE_JAR
     - DB2_REPLACE_JAR
   - コレクション DSNJAR に属するパッケージの実行権限

k. Java 実装の Business Process Choreographer ユーザー定義関数 (UDF) を使用する場合は、Java ユーザー定義関数および解釈済み Java ルーチンを実行するための DB2 環境を準備します。 以下を実行します。
   1) DB2 資料に説明されているように、DB2 付属のストアード・プロシージャーを使用可能にし、DB2Universal JDBC ドライバーによって使用されるテーブルを定義します。
   2) DB2 資料に説明されているように、解釈済み Java ルーチン向け環境をセットアップします。

3) このプロシージャー中に作成された WLM アプリケーション環境の名前を書き留めておいてください。

2. USS にログオンします。
3. 使用しているデータベース・システム用の Business Process Choreographer Event Collector のデータベース・スクリプトがあるディレクトリーに移動します。 使用している DB2 のバージョンに応じて、以下のコマンドのいずれかを入力します。

```
cd install_root/dbscripts/ProcessChoreographer/DB2zOSV8
cd install_root/dbscripts/ProcessChoreographer/DB2zOSV9
```

注: WebSphere Process Server をインストール済みでない場合、スクリプトは製品メディアの *media_root* または *extract_directory*/dbscripts/ProcessChoreographer/DB2zOSV*x*/ (ここで *x* はバージョン番号) でも入手できます。

4. テーブル・スペースを作成します。
   a. ASCII createTablespace_Observer.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。

      注: SQL 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、または実行に使用するツールによっては、このファイルを EBCDIC に変換する必要があります。このファイルを EBCDIC に変換するには、以下のコマンドを入力します。

      ```
      iconv -t IBM-1047 -f ISO8859-1 createTablespace_Observer.sql > createTablespace_Observer.sql_EBCDIC.sql
      ```

      このファイルを変換して ASCII に戻すには、以下のコマンドを入力します。

      ```
      iconv -t ISO8859-1 -f IBM-1047 createTablespace_Observer_EBCDIC.sql > createTablespace_Observer_ASCII.sql
      ```

   b. 使用しているデータベースに接続していることを確認し、createTablespace_Observer.sql スクリプトのカスタマイズ・バージョンを実行します。
5. スキーマを作成します。
   a. createSchema_Observer.sql スクリプトのヘッダーの説明に従って、この SQL ファイルを編集します。

      注: この SQL 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、または実行に使用するツールによっては、このファイルを EBCDIC に変換する必要があります。詳しくは、ステップ 4 の注を参照してください。

   b. 使用しているデータベースに接続していることを確認し、createSchema_Observer.sql スクリプトのカスタマイズ・バージョンを実行します。
6. Java 実装のユーザー定義関数 (UDF) を使用する場合は、255 ページの『Java ユーザー定義関数または SQL ユーザー定義関数の選択』を実行します。
7. 管理コンソールを使用して、このデータベースを指す XA データ・ソースを作成します。

タスクの結果

レポート・データベース用のデータベース・スキーマが作成されました。

関連概念

255 ページの『Business Process Choreographer Explorer レポート作成機能のユーザー定義関数』

Business Process Choreographer Explorer レポート作成機能では、タイム・スライスまたは時間間隔に基づいてレポートを実行できます。これにより、SQL 照会が生成されます。このようなレポートを実行するため、Business Process Choreographer Explorer レポート作成機能ではレポート・データベースに特定のユーザー定義関数 (UDF) がインストールされている必要があります。

関連タスク

255 ページの『Java ユーザー定義関数または SQL ユーザー定義関数の選択』

レポート・データベースで Java ベースのユーザー定義関数 (UDF) と SQL ベースのユーザー定義関数を切り替えるには、setupEventCollector ツールを使用します。

リモート・システムからの Business Process Choreographer Explorer レポート作成機能のための DB2 for z/OS データベースの作成:

ここでは、対話式メニュー方式のツール、および `createTablespace_Observer.sql` スクリプトを システムで使用して、レポート・データベース用のスキーマを準備する方法について説明します。

始める前に

WebSphere Process Server が サーバーにインストール済みである必要があります。

手順

1. 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. ストレージ・グループを作成し、その名前 (例えば OBSVRSG) を書き留めます。
   e. 新規データベースを使用する場合は、例えば OBSVRDB という名前の新規データベースを作成します。 必要であれば、既存のデータベースおよびストレージ・グループ (例えば、Business Process Choreographer データベースの BPEDB) を再使用することができます。
   f. 使用するスキーマ修飾子を決定します。
   g. データベースのセットアップに使用するユーザー ID *user_ID* を決定します。 これは、実行時にデータベースにアクセスするために使用されるユーザー ID ではありません。
   h. 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ための以下の権限をユーザー ID が持っていることを確認します。
      - ストレージ・グループを使用する権限
      - データベース OBSVRDB を使用する権限

- データベース OBSVRDB 内にテーブル・スペースを作成する権限
- データベース OBSVRDB 内にテーブルを作成する権限

i. Java 実装の Business Process Choreographer ユーザー定義関数 (UDF) を使用する場合は、このユーザー ID が以下の権限を所有していることも確認します。
   - SYSIBM.SYSJAROBJECTS に対して選択を実行する権限
   - スキーマ SQLJ に対して以下のストアード・プロシージャーを実行する権限
     - INSTALL_JAR
     - REMOVE_JAR
     - REPLACE_JAR
     - DB2_INSTALL_JAR
     - DB2_REMOVE_JAR
     - DB2_REPLACE_JAR
   - コレクション DSNJAR に属するパッケージの実行権限

j. Java 実装の Business Process Choreographer ユーザー定義関数 (UDF) を使用する場合は、Java ユーザー定義関数および解釈済み Java ルーチンを実行するための DB2 環境を準備します。 以下を実行します。
   1) DB2 資料に説明されているように、DB2 付属のストアード・プロシージャーを使用可能にし、DB2Universal JDBC ドライバーによって使用されるテーブルを定義します。
   2) DB2 資料に説明されているように、解釈済み Java ルーチン向け環境をセットアップします。
   3) このプロシージャー中に作成された WLM アプリケーション環境の名前を書き留めておいてください。

2. WebSphere Process Server をホストするサーバーで、以下のようにします。
   a. ネイティブの DB2 クライアントを使用する場合は、リモート・データベースをカタログし、そのデータベースとの接続を確立できることを確認します。DB2 コマンド行のウィンドウで、以下のコマンドを入力します。

```
catalog tcpip node zosnode remote host_name server IP_port ostype mvs
catalog database location as database_alias at node zosnode
        authentication dcs
catalog dcs database database_alias parms ',,INTERRUPT_ENABLED'
```

ここで、

zosnode
リモートの z/OS ノードのローカル別名です。

host_name
リモートの z/OS システムの TCP/IP アドレスか、または別名です。

IP_port
DB2 サブシステムが listen しているポート番号です。

database_alias
リモート・データベースにアクセスするためのローカルの別名です。

location
リモートの DB2 のロケーション名です。このロケーション名を見つけるには、TSO にログオンし、使用可能な照会ツールの 1 つを使用して、選択したサブシステムで以下の SQL 照会を入力します。

```
select current server from sysibm.sysdummy1
```

リモート・システムに接続できることを確認するには、以下を入力します。

```
db2 connect to database_alias user userid using password
```

b. 使用しているデータベース・システムに対応した Business Process Choreographer Explorer レポート作成機能のスクリプトがあるディレクトリーに移動します。 使用している DB2 のバージョンに応じて、以下のコマンドのいずれかを入力します。

```
cd install_root/dbscripts/ProcessChoreographer/DB2zOSV8
cd install_root/dbscripts/ProcessChoreographer/DB2zOSV9
```

c. ASCII createTablespace_Observer.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. スクリプトのカスタマイズ・バージョンを実行します。

```
db2 -tf createTablespace_Observer.sql
```

テーブル・スペースを除去する場合は、dropTablespace_Observer.sql スクリプトを使用します。

e. 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。

f. 272 ページの『setupEventCollector ツール』の説明に従って、このツールを開始して Event Collector をセットアップします。

g. オプション 1 を選択して、Event Collector アプリケーション用にデータベースを準備します。

h. 以下が表示された場合:

```
c) Derby
8) DB2 V8/V9 on z/OS
```

8 を入力して、DB2 on z/OS を選択します。

i. このツールを使用すると、SQL ファイルを作成し、そのファイルを自分の現在のユーザー ID で実行するのではなく、データベース管理者に渡して実行することができます。 以下が表示された場合:

```
Do you want to create an SQL file only (delay database preparation)?
  y) yes
  n) no
```

- SQL の実行を遅らせない場合は、n を入力します。
- SQL の実行を遅らせる場合は、y を入力します。以下のメッセージが表示されます。

```
Even if you want to delay the configuration,
your entered values can be checked within the database.
Do you want to perform these checks?
  y) yes
  n) no
```

– 入力した値がデータベース内でチェックされるようにするには、y を入力します。
– そうでない場合は、n を入力します。

入力した内容によっては、以下のプロンプトのうち、表示されないものがあります。表示されないステップはスキップしてください。

j. 以下が表示された場合:

```
Specify the JDBC driver type to be used:

  2) Connect using type 2 (using a native database client)
  4) Connect using type 4 (directly via JDBC)
```

以下のようにして JDBC ドライバーのタイプを指定します。

- ネイティブのデータベース・クライアントを使用している場合は、2 を入力します。
- それ以外は 4 を入力して、タイプ 4 の JDBC ドライバーを選択します。

k. 以下が表示された場合:

```
Specify the name of database in local catalog:  [BPEDB]
```

ローカルの DB2 クライアントにカタログされたときのデータベースの名前を入力します。この名前は、ステップ 2a (246 ページ) で *database_alias* に使用した値です。

l. 以下が表示された場合:

```
Specify the location name/connection target: []
```

接続先のサブシステムのロケーション名を入力します。

注: このロケーション名を判別するには、SQL プロセッサーを使用してログオンし、以下の SQL ステートメントを実行します。

```
SELECT CURRENT SERVER FROM SYSIBM.SYSDUMMY1
```

m. 以下が表示された場合:

```
Specify the name of the database as known by the subsystem: [OBSVRDB]
```

z/OS ホスト上のサブシステムでデータベースを識別するための名前を入力します。

n. 以下が表示された場合:

```
Specify the hostname of the database server:  [localhost]
Specify the port where the database server is listening:  [446]
```

z/OS データベース・サーバーで使用されているホスト名とポート番号を入力します。

o. 以下が表示された場合:

```
Specify the directory of your JDBC driver:  []
```

DB2 JDBC ドライバー用の db2jcc.jar JAR ファイルおよび db2jcc_license_cisuz.jar JAR ファイルがあるディレクトリーを入力します。

p. 以下が表示された場合:

```
Specify userid to connect to the database 'database_name' [db2admin] :
Specify the password for userid 'user_ID' :
```

データベースに接続するためのユーザー ID およびパスワードを入力します。これは、ステップ 1g (245 ページ) で説明されているユーザー ID *user_ID* です。

q. 以下が表示された場合:

```
Specify the database schema to be used. [user_ID] :
```

データベース・オブジェクトに使用するデータベース・スキーマの名前を入力します。

r. 以下が表示された場合:

```
Note: The Java UDFs are more precise, but they require a jar file
installed to the database.
Visit the reporting function documentation for details.

  1) Java
  2) SQL

  0) Exit Menu
```

- より厳密な Java ベースのユーザー定義関数 (UDF) を使用する場合は、1 を入力します。この場合は、JAR ファイルがデータベースにインストールされている必要があります。
- あまり厳密ではない SQL ベースの UDF を使用する場合は、2 を入力します。

s. 以下が表示された場合:

```
Specify the DB2 storage group name to be used. [OBSVRSG] :
```

ステップ 1d (245 ページ) のストレージ・グループ名を入力します。

t. 以下が表示された場合:

```
Specify the WLM environment name where the UDF should run. [] :
```

ステップ 1j (246 ページ) で書き留めておいた WLM 環境を入力します。 必要なテーブル・スペースの有無を確認し、JAR ファイルをデータベースにロードすると、成功したことが以下により示されます。

```
The setup of the database completed successfully.
```

3. 管理コンソールを使用して、このデータベースを指す XA データ・ソースを作成します。

タスクの結果

レポート・データベース用のデータベース・スキーマが作成されました。

関連概念

255 ページの『Business Process Choreographer Explorer レポート作成機能のユーザー定義関数』

Business Process Choreographer Explorer レポート作成機能では、タイム・スライスまたは時間間隔に基づいてレポートを実行できます。これにより、SQL 照会が生成されます。このようなレポートを実行するため、Business Process Choreographer Explorer レポート作成機能ではレポート・データベースに特定のユーザー定義関数 (UDF) がインストールされている必要があります。

関連タスク

255 ページの『Java ユーザー定義関数または SQL ユーザー定義関数の選択』

レポート・データベースで Java ベースのユーザー定義関数 (UDF) と SQL ベースのユーザー定義関数を切り替えるには、setupEventCollector ツールを使用します。

Business Process Choreographer Explorer レポート作成機能のための Derby データベースの準備:

スクリプトまたは対話式ツールのいずれかを使用して、レポート・データベースを準備できます。

SQL スクリプトを使用した Business Process Choreographer Explorer レポート作成機能のための Derby データベースの準備:

ここでは、createSchema_Observer.sql スクリプトを使用して、レポート・データベースのスキーマを準備する方法について説明します。

このタスクについて

レポート・データベースにはスキーマを作成する必要があります。既存のデータベースにスキーマを作成することも、あるいはスクリプト・ファイルで新規データベースを作成することもできます。

手順

1. バッチ・モードで bpeconfig.jacl スクリプトを使用するか、または管理コンソールを使用して Business Process Choreographer を構成した場合、生成された SQL スクリプトを使用して Business Process Choreographer Explorer レポート作成機能用のデータベースを作成します。
   a. 生成された SQL ファイルを見つけます。 createSchema_Observer.sql という名前の ASCII SQL スクリプトと、createSchema_Observer.ddl という名前の、それと等価な EBCDIC DDL スクリプトがあります。
      - スキーマ修飾子を指定した場合は、どちらのファイルも *profile_root*/dbscripts/ProcessChoreographer/Derby/*database_name*/*database_schema*にあります。
      - スキーマ修飾子を指定**しなかった**場合は、どちらのファイルも *profile_root*/dbscripts/ProcessChoreographer/Derby/*database_name* にあります。

      各部の意味は、次のとおりです。

      database_name
      データベースの名前です。

database_schema
スキーマを使用している場合は、そのスキーマの名前です。

collection_name
コレクションの名前です。

b. Derby ネットワーク・サーバー環境で、SQL スクリプトをネットワーク・サーバーにコピーします。

c. JAR ファイル `bpcodbutil.jar` を *install_root* ディレクトリーの `lib` サブディレクトリーから、データベース・サーバー上の同じディレクトリーにコピーします。

d. 組み込み Derby ドライバーを使用して既存のデータベースに接続する場合は、そのデータベースを使用しているサーバーおよびその他のすべてのアプリケーションを停止します。

e. スクリプトを実行してスキーマを作成します。 以下に例を挙げます。

- OBSVRDB という名前のデータベースを作成するには、データベースを作成するディレクトリーで次のコマンドを入力します。

```
java -Dij.protocol=jdbc:derby:
     -Dij.database=OBSVRDB;create=true
     org.apache.derby.tools.ij
     createSchema_Observer.sql
```

- OBSVRDB という名前のデータベースが既に存在する場合は、データベースが作成されたディレクトリーで次のコマンドを入力します。

```
java -Dij.protocol=jdbc:derby:
     -Dij.database=OBSVRDB
     org.apache.derby.tools.ij
     createSchema_Observer.sql
```

2. 対話モードで bpeconfig.jacl スクリプトを使用して Business Process Choreographer を構成した場合、または Business Process Choreographer をまだ構成していない場合、SQL スクリプトは生成されません。標準の SQL スクリプトのコピーをカスタマイズする必要があります。

a. データベースの標準の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。

b. Derby ネットワーク・サーバー環境で、`*Observer.sql` スクリプトをネットワーク・サーバーにコピーします。

c. JAR ファイル `bpcodbutil.jar` を *install_root* ディレクトリーの `lib` サブディレクトリーから、データベース・サーバー上の同じディレクトリーにコピーします。

d. 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。 新規データベースを作成する場合は、データベース名に `;create=true` を付加してください。例えば、データベース名が OBSVRDB の場合は、パラメーター `-Dij.database=OBSVRDB` を `-Dij.database=OBSVRDB;create=true` で置き換えます。

注: Windows プラットフォームの場合は、メモ帳エディターを使用しないでください。メモ帳エディター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。

e. 組み込み Derby ドライバーを使用して既存のデータベースに接続する場合は、そのデータベースを使用しているサーバーおよびその他のすべてのアプリケーションを停止します。

f. 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。 このスクリプト・ファイルは、ASCII フォーマットで提供されています。 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 コマンド (`viascii createSchema_Observer.sql`) を使用して直接 ASCII フォーマットで編集し、次に iconv コマンドを使用して、このファイルを EBCDIC に変換できます。

g. 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。

3. 管理コンソールを使用して、このデータベースを指す XA データ・ソースを作成し、接続をテストします。

タスクの結果

レポート・データベース用のデータベース・スキーマが作成されました。

setupEventCollector ツールを使用した Business Process Choreographer Explorer レポート作成機能のための Derby データベースの準備:

ここでは、対話式メニュー方式のツールの setupEventcollector を使用して、サポートされている任意のプラットフォームにレポート・データベース用の Derby データベースを準備する方法について説明します。

手順

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。
2. 組み込み Derby ドライバーを使用して既存のデータベースに接続する場合は、そのデータベースを使用しているサーバーおよびその他のすべてのアプリケーションを停止します。`-conntype none` をツールの開始時に使用する計画を立てます。
3. 272 ページの『setupEventCollector ツール』の説明に従って、このツールを開始して Event Collector をセットアップします。
4. 以下が表示された場合:

```
1) Prepare a database for the Event Collector and reporting function
2) Install the Event Collector application
3) Remove the Event Collector application and related objects
4) Change configuration settings of an installed Event Collector
5) Drop the database schema of the Event Collector and reporting function
6) Administer reporting function related user-defined functions

0) Exit Menu
```

オプション 1 を選択して、Event Collector アプリケーション用にデータベースを準備します。 以下のメニューが表示されます。

```
Prepare a database for the Event Collector and reporting function

Select the type of your database provider:

     c) Derby

     8) DB2 V8/V9 on z/OS

     0) Exit Menu
```

5. c を入力して Derby を選択します。
6. このツールを使用すると、SQL ファイルを作成し、そのファイルを自分の現在のユーザー ID で実行するのではなく、データベース管理者に渡して実行することができます。 以下が表示された場合:

```
Do you want to create an SQL file only (delay database preparation)?
  y) yes
  n) no
```

   - SQL の実行を遅らせない場合は、n を入力します。
   - SQL の実行を遅らせる場合は、y を入力します。以下が表示されます。

```
Even if you want to delay the configuration,
your entered values can be checked within the database.
Do you want to perform these checks?
  y) yes
  n) no
```

     – 入力した値がデータベース内でチェックされるようにするには、y を入力します。
     – そうでない場合は、n を入力します。

     入力した内容によっては、以下のプロンプトのうち、表示されないものがあります。表示されないステップはスキップしてください。

7. 以下が表示された場合:

```
Specify the JDBC driver type to be used:

  1) Connect using the embedded or embedded 40 JDBC driver
  2) Connect using the network or network 40 JDBC driver

Your selection: [1]
```

   - 組み込み JDBC ドライバーを使用して接続するには、1 を入力します。

     重要: このドライバーを使用してデータベースを構成している間は、他のアプリケーション (WebSphere Process Server など) がデータベースに接続されていないことを確認してください。
   - ネットワーク JDBC ドライバーを使用するには、2 を入力します。

8. 以下が表示された場合:

```
Specify the name of your database [database_name]
```

   このデータベースへの完全修飾パスを入力します。

   注: デフォルト値 **...¥BPEDB** は、Business Process Choreographer で使用されるのと同じデータベースです。パフォーマンスをさらに向上させるには、別のデータベースを使用してください。

9. 以下が表示された場合:

```
Specify the database schema to be used. [APP] :
```

データベース・オブジェクトに使用するデータベース・スキーマ名を入力します。スペース文字を入力するか、またはこのフィールドを空のままにしておくと、デフォルト・スキーマ APP が使用されます。

10. 以下が表示された場合:

    ```
    Specify the hostname of the database server:  [localhost]
    Specify the port where the database server is listening:  [1527]
    ```

    Derby ネットワーク・サーバーのホスト名とポート番号を入力します。

11. 以下が表示された場合:

    ```
    Specify the directory of your JDBC driver:  [B:\w\p\derby\lib]
    ```

    - 組み込み JDBC ドライバーの場合は、derby.jar ファイルがあるディレクトリーを入力します。
    - ネットワーク JDBC ドライバーの場合は、derbyclient.jar があるディレクトリーを入力します。

12. 以下が表示された場合:

    ```
    Specify userid to connect to the database database_name:  []
    ```

    - サーバーが認証を要求している場合は、使用している Derby ネットワーク・サーバーへの接続権限を付与されているユーザー ID を入力します。
    - 認証が要求されていない場合で、値を入力しなかった場合は、ユーザー ID dummy が使用されます。これは、Derby JDBC ドライバーは常にネットワーク・サーバーとの接続にユーザー ID を必要とするためです。

13. 以下が表示された場合:

    ```
    The application server must be stopped to update a Derby database.
    This must be done outside wsadmin using 'stopServer server_name'.
    After the server is stopped, come back to this prompt and enter
    'c' to continue.
    Please stop the server 'server_name' now.
    Press 'c' to continue, 'a' to abort:
    ```

    a. 以下のコマンドを使用して、wsadmin の外部でサーバーを停止します。

       ```
       stopServer server_name
       ```

    b. サーバーを停止した場合、c を押して続行します。続行しない場合は、a を押して、ステップ 4 (252 ページ) に示されているメインメニューに戻ります。

14. 以下が表示された場合:

    ```
    Specify the database schema to be used. [APP] :
    ```

    データベース・オブジェクトに使用されるスキーマの名前を入力するか、または Enter を押してデフォルトを使用します。

15. 以下のメッセージが表示されることを確認します。これにより、データベースが正常に準備されたことがわかります。

    ```
    The setup of the database completed successfully.
    ```

16. 以下が表示された場合:

    ```
    Restart the server now using 'startServer server_name'.
    After the server is up again, come back to this prompt and enter
    'c' to continue.
    Press 'c' to continue, 'a' to abort:
    ```

    a. 以下のコマンドを使用して、サーバーを始動します。

```
startServer server_name
```

b. サーバーの始動が完了するまで待機してからこのプロンプトに戻り、c を押して続行します。続行しない場合は、a を押して、ステップ 4 (252 ページ) に示されているメインメニューに戻ります。

成功したことは、次のメッセージによって示されます。

```
WASX7074I: ローカル・ホストをホストする SOAP コネクターの再接続が完了しました
(WASX7074I: Reconnect of SOAP  connector to host localhost completed.)
```

17. 管理コンソールを使用して、このデータベースを指す XA データ・ソースを作成し、接続をテストします。

タスクの結果

レポート・データベース用のデータベース・スキーマが作成されました。

Java ユーザー定義関数または SQL ユーザー定義関数の選択

レポート・データベースで Java ベースのユーザー定義関数 (UDF) と SQL ベースのユーザー定義関数を切り替えるには、setupEventCollector ツールを使用します。

関連概念

『Business Process Choreographer Explorer レポート作成機能のユーザー定義関数』
Business Process Choreographer Explorer レポート作成機能では、タイム・スライスまたは時間間隔に基づいてレポートを実行できます。これにより、SQL 照会が生成されます。このようなレポートを実行するため、Business Process Choreographer Explorer レポート作成機能ではレポート・データベースに特定のユーザー定義関数 (UDF) がインストールされている必要があります。

Business Process Choreographer Explorer レポート作成機能のユーザー定義関数:

Business Process Choreographer Explorer レポート作成機能では、タイム・スライスまたは時間間隔に基づいてレポートを実行できます。これにより、SQL 照会が生成されます。このようなレポートを実行するため、Business Process Choreographer Explorer レポート作成機能ではレポート・データベースに特定のユーザー定義関数 (UDF) がインストールされている必要があります。

次のいずれかの UDF 実装をインストールできます。

SQL 実装

データベース・システムの組み込み時間関数を使用してプレーン SQL で実装されている UDF の SQL 実装を使用します。

SQL 実装のインストールは、Java 実装のインストールよりも容易です。これは、SQL 実装の場合、必要な操作は、提供される SQL スクリプトの実行のみであるためです。これらのスクリプトをインストールするときに必要な管理権限は、Java 実装の場合よりも低いです。また、SQL 実装は Java 実装よりも高いパフォーマンスを備えています。しかし、組み込み時間関数の制約により、SQL 実装による UDF は、十分な正確性がなく、ニーズに対応できない場合があります。例えば DB2 では、組み込み時間関数は各月の長さが 30 日であると想定しますが、これが原因で誤った結果が生じる可能性があります。

Derby データベースでは SQL 実装は使用できません。

Java 実装

Java 言語で実装されている UDF の Java 実装を使用します。

Java 実装をインストールするには、データベース・システムのメカニズムを使用します。Java 実装による UDF では、正確なレポートが生成されます。ただし、Java 実装をインストールする手順は SQL 実装のインストール手順よりも多く、データベースでは SQL 実装よりも高い管理権限を必要とします。例えば DB2 z/OS データベースでは、UDF を実行するためにワークロード・マネージャー (WLM) 環境をセットアップする必要があります。

選択したデータベースのセットアップ方法に応じて、デフォルトの実装は異なります。

- SQL スクリプトを使用するか、または初回アクセス時にテーブルを作成する機能を使用するようにデータベースをセットアップした場合は、デフォルトで SQL 実装がインストールされます。
- setupEventCollector ツールを使用するか、またはプロファイル作成ウィザード (Derby データベースでのみ使用可能) で Business Process Choreographer サンプル構成を使用するようにデータベースをセットアップした場合は、デフォルトで Java 実装がインストールされます。

初期セットアップ後に、UDF の実装を変更できます。これについては、 255 ページの『Java ユーザー定義関数または SQL ユーザー定義関数の選択』で説明します。

関連タスク

255 ページの『Java ユーザー定義関数または SQL ユーザー定義関数の選択』
レポート・データベースで Java ベースのユーザー定義関数 (UDF) と SQL ベースのユーザー定義関数を切り替えるには、setupEventCollector ツールを使用します。

242 ページの『USS における SQL スクリプトを使用した Business Process Choreographer Explorer レポート作成機能のための DB2 for z/OS データベースの準備』
ここでは、スクリプトを使用してを UNIX System Services (USS) で使用して、DB2 for z/OS データベースを準備する方法について説明します。

245 ページの『リモート・システムからの Business Process Choreographer Explorer レポート作成機能のための DB2 for z/OS データベースの作成』
ここでは、対話式メニュー方式のツール、および `createTablespace_Observer.sql` スクリプトを システムで使用して、レポート・データベース用のスキーマを準備する方法について説明します。

関連資料

272 ページの『setupEventCollector ツール』
setupEventCollector は、Business Process Choreographer Event Collector アプリケーションの対話式での構成または削除、データベースのセットアップ、およびデータベースのユーザー定義関数の管理に使用されます。このツールでは wsadmin スクリプトが使用されます。Business Process Choreographer Explorer レポート作成機能を使用する場合は、Event Collector を構成する必要があります。

setupEventCollector ツールを使用した Java ユーザー定義関数と SQL ユーザー定義関数との間での選択:

ここでは、対話式メニュー方式のツールを使用して、Business Process Choreographer レポート・データベースで Java ベースのユーザー定義関数 (UDF) と SQL ベースのユーザー定義関数を切り替える方法について説明します。

このタスクについて

setupEventCollector はデフォルトでは Java ベースの UDF を使用しますが、このツールを使用すると SQL ベースの UDF に切り替えることができます。元に戻したい場合も、このツールを使用して Java ベースの UDF に戻すことができます。

手順

1. 272 ページの『setupEventCollector ツール』の説明に従って、このツールを開始して Event Collector をセットアップします。 以下のメニューが表示されます。

```
1) Prepare a database for the Event Collector and reporting function
2) Install the Event Collector application
3) Remove the Event Collector application and related objects
4) Change configuration settings of an installed Event Collector
5) Drop the database schema of the Event Collector and reporting function
6) Administer reporting function related user-defined functions

0) Exit Menu
```

2. オプション 6 を選択して、Business Process Choreographer Explorer レポート作成機能用のユーザー定義関数を管理します。 以下のメニューが表示されます。

```
c) Derby

8) DB2 V8/V9 on z/OS
```

3. データベース・バージョンのオプション 8 を選択します。
   a. 以下のメニューが表示された場合:

```
Specify which type should be used to connect to the Database:

  2) Connect using type 2 (using a native DB2 client)
  4) Connect using type 4 (directly via JDBC)
```

   以下のいずれかのオプションを選択します。

   2 ネイティブの DB2 クライアントが使用される、タイプ 2 の JDBC 接続の場合。この場合は、プロンプトが出て、以下を入力するよう要求されます。

   データベース名

   データベース・ユーザー ID

   パスワード

   データベース・ロケーション (サブシステム)

   4 直接接続する、タイプ 4 の JDBC ドライバーの場合。この場合は、プロンプトが出て、以下を入力するよう要求されます。

   データベース名

   データベース・サーバー・ホスト名

   データベース・サーバー・ポート番号

   データベース・ロケーション (サブシステム)

データベース・ユーザー ID

パスワード

4. データベースとの接続が確立すると、データベースの UDF を管理するためのメニューが表示されます。

```
6) Administer reporting function related user-defined functions

  1) Activate Java based user-defined functions
  2) Activate SQL based user-defined functions
  3) Determine current state
  4) List, install or remove the jar file containing the java based functions
```

   a. Java ベースの UDF をアクティブ化するには、オプション 1 を選択します。

      1) 以下が表示された場合:

```
Specify the database schema to be used:
```

         データベース・スキーマの名前を入力します。

      2) 以下が表示された場合:

```
WARNING: Switching the UDF implementation type may break any
running reporting function applications. Continue anyway?
y) yes
n) no
Your selection:
```

         重要なレポートを実行中の場合、n を入力して切り替えを続行しないか、または完了するまで待機します。続行する場合は、y を入力します。

      3) 続行する場合は、以下のように表示されます。

```
Removing the user-defined functions ...

The jar file with jar_id 'DB2INST1.BPCODBUTIL' is updated
with the current version.
Loading the jar file 'B:¥w¥p¥lib¥bpcodbutil.jar' into the database.
The jar file 'BPCODBUTIL' was successfully installed.

Creating the Java based user-defined functions ...
```

      4) 以下のメッセージによって、成功したことがわかります。

```
The setup of the database completed successfully.
```

   b. SQL ベースの UDF をアクティブ化するには、オプション 2 を選択します。

      1) 以下が表示された場合:

```
Specify the database schema to be used:
```

         データベース・スキーマの名前を入力します。

      2) 以下が表示された場合:

```
WARNING: Switching the UDF implementation type may break any
running reporting function applications. Continue anyway?
y) yes
n) no
Your selection:
```

         続行する場合は y を、続行しない場合は n を入力します。

      3) 以下が表示された場合:

```
Removing the user-defined functions ...

Creating the SQL based user-defined functions ...

Do you also want to remove the jar file from the database?
y) yes
n) no
Your selection:
```

データベースから JAR ファイルを除去する場合は y を、除去しない場合は n を入力します。

4) 以下のメッセージによって、成功したことがわかります。

```
The setup of the database completed successfully.
```

c. オプション: 選択した UDF 実装が Java であるのか、SQL であるのかを判別し、Java がアクティブである場合には、JAR ファイルがインストールされていることも確認するには、オプション 3 を選択します。 例えば、Java 実装がアクティブである場合は、以下のようなメッセージを受け取ります。

```
The active UDF implementation is Java.
Tested functionality of the UDF, is working
```

d. オプション: Java ベースの UDF に必要な JAR ファイルをインストールまたは除去するか、あるいはデータベースにインストールされているすべての JAR ファイルをリストするには、オプション 4 を選択します。以下のメニューが表示されます。

```
List, install or remove jar files containing the java based functions

  1) Install the jar file containing the reporting function functions
     into the database
  2) Remove the jar file containing the reporting function functions
     from the database
  3) List installed jar files

  0) Exit Menu
```

- オプション 1 を選択して、JAR ファイルをインストールします。
- オプション 2 を選択して、JAR ファイルを除去します。
- オプション 3 を選択して、データベースにインストールされている JAR ファイルをリストします。
- オプション 0 を選択して、メニューを終了します。

e. オプション 0 を繰り返し選択して、ステップ 1 (257 ページ) で表示されているメニューに戻ります。

タスクの結果

レポート・データベースでは、選択した UDF を使用します。

Business Process Choreographer Event Collector アプリケーションの構成

Business Process Choreographer Event Collector は、 Business Process Choreographer Explorer レポート作成機能を使用するための前提条件です。 Event Collector アプリケーションのインストールと構成は、対話式ツールまたは管理コンソールで行うことができます。

始める前に

Event Collector アプリケーションのインストール先となるデプロイメント・ターゲットに、Common Event Infrastructure (CEI) を構成する必要があります。

このタスクについて

Business Process Choreographer Event Collector を構成するには、以下のいずれかを実行します。

setupEventCollector ツールを使用した Business Process Choreographer Event Collector の構成:

ここでは、対話式メニュー方式のツールを使用して、Event Collector アプリケーションをサーバーまたはクラスターにインストールして構成する方法について説明します。

手順

1. 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。 次のコマンドを入力します。

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

2. 272 ページの『setupEventCollector ツール』の説明に従って、このツールを開始して Event Collector をセットアップします。 「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 and reporting function
     2) Install the Event Collector application
     3) Remove the Event Collector application and related objects
     4) Change configuration settings of an installed Event Collector
     5) Drop the database schema of the Event Collector and reporting function
     6) Administer reporting function related user-defined funtions

     0) Exit Menu
   ```

3. Business Process Choreographer Event Collector アプリケーションをインストールするには、以下のようにします。

   a. オプション 2 を選択します。 以下が表示されます。

      ```
      Create required objects and install the WebSphere Business Process
      Choreographer Event Collector application ...
      ```

   b. スタンドアロン・サーバーにインストールする場合は、以下が表示されます。

      ```
      Working on node 'your_node_name', server 'your_server_name'.
      ```

   c. アプリケーションをデプロイメント・マネージャーにインストールする場合は、使用可能なすべてのターゲットのリストからデプロイメント・ターゲットを選択する必要があります。 以下に例を挙げます。

      ```
      Select the deployment target to install to:

        1) Cluster 'cluster1'
        2) Node 'Node04', Server 'managed1'
        3) Node 'Node04', Server 'managed2'

        0) Exit Menu
      ```

d. ツールが Network Deployment 環境の既存の Event Collector インストールを検索している間、以下のように表示されます。

```
Searching for an already installed Event Collector on 'deployment_target'
```

e. 既に Event Collector アプリケーションのインスタンスがインストールされている場合は、以下が表示されます。

```
Do you want to overwrite the existing application?
  o) Overwrite
  a) Abort
```

- 既存の Event Collector アプリケーションを上書きする場合は、o を入力します。すべてのインストール値が再入力され、Event Collector アプリケーションが更新されます。
- Event Collector をインストールせずに終了する場合は、a を入力します。

4. 以下が表示された場合:

```
Specify the JNDI name of the database where the WebSphere Business Process
Choreographer Event Collector should store the collected events.
Enter '?' to get a list.
Your selection : [jdbc/BPEDB]
```

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。以下に例を挙げます。

```
jdbc/BPEDB
jdbc/DefaultEJBTimerDataSource
jdbc/mediation/messageLog
```

5. 以下が表示された場合:

```
Specify the database schema to be used.
Enter a space character or leave empty to use the default schema of the
datasource. [] :
```

Event Collector がイベントを保管しているデータベース表のスキーマの名前を入力します。データ・ソース定義の認証別名で指定されているユーザー ID をスキーマとして使用するには、スペース文字を入力するか、またはそのフィールドを空にしておきます。

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!
```

6. サーバーで CEI ロギングが有効に設定されていない場合は、以下のように表示されます。

```
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)
```

- スクリプトにより指定のサーバーで CEI ロギングを有効にする場合は、y と入力します。

- スクリプトにより指定のサーバーで CEI ロギングを有効にしない場合は、n と入力します。

注: Business Process Choreographer Explorer レポート作成機能 を使い始める時は、CEI ロギングが有効になっていることが重要です。

7. 以下のプロンプトが表示された場合:

   ```
   Do you want to save the changes? (y/n)
   ```

   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。setupEventCollector.log という名前のログ・ファイルを確認します (このファイルは、プロファイルの logs ディレクトリーにあります)。
8. 0 を入力して、メニューを終了します。
9. 変更を有効にします。
   - ツールの開始時に -conntype NONE オプションを指定した場合は、サーバーを再始動しなければ変更内容はアクティブになりません。
   - ツールの開始時に -conntype NONE オプションを指定しなかった場合で、Business Process Choreographer Event Collector のインストール中にサーバーで CEI ロギングを使用可能にした場合は、管理コンソールを使用して BPEContainer アプリケーションを停止してから再始動してください。

タスクの結果

Business Process Choreographer Event Collector アプリケーションがインストールおよび構成されています。

管理コンソールを使用した Business Process Choreographer Event Collector の構成:

ここでは、管理コンソールを使用して、特定のサーバーまたはクラスターに Business Process Choreographer Event Collector のインスタンスをインストールする方法について説明します。

始める前に

レポート・データベースの準備が完了しています。

手順

1. 管理コンソールで、次のように Business Process Choreographer Event Collector の構成ページに移動します。 「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「**クラスター名**」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「**サーバー名**」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」、「**Business Process Choreographer Event Collector**」をクリックします。
2. 新規構成を作成するには、以下のようにします。
   a. 以下のフィールドで値を入力するか、または選択します。
      - データベース名。

- スキーマ名。
- 初めてデータベースが使用されるときにデータベース表を作成するためのオプションを有効にするか、またはクリアします。
- データベースに接続するためのユーザー名およびパスワード。
- データベース・サーバーのホスト名または IP アドレス。
- データベース・サーバーのポート番号。
- JDBC プロバイダー。
- 監視ターゲット:
  - **管理対象の Business Process Choreographer コンテナー**
  - **既存のイベント・グループ名**
  - **イベント・グループ名**

b. 「**適用**」をクリックして、アプリケーションをデプロイします。

c. 問題がある場合は、SystemOut.log ファイルを確認します。 問題がない場合は、変更をマスター構成に保存します。

d. 「**アプリケーション**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere エンタープライズ・アプリケーション**」をクリックしてアプリケーションを開始し、アプリケーション BPCECollector_*scope* を選択します。ここで、*scope* はデプロイメント・ターゲットを示しています。次に「**開始**」をクリックします。

タスクの結果

Business Process Choreographer Event Collector が構成されています。

Business Process Choreographer Explorer レポート作成機能のマイグレーション後の使用可能化

既存の Business Process Choreographer Observer と Explorer の構成をマイグレーションして使用することもできますが、Business Process Choreographer Observer の代わりに使用できる新しい Business Process Choreographer Explorer のレポート作成機能は無効になります。

始める前に

旧リリースからのマイグレーションを行い、Business Process Choreographer Explorer の構成と既存の Business Process Choreographer Observer の構成をマイグレーションしておきます。

このタスクについて

旧リリースの Business Process Choreographer Observer は変更されず、マイグレーション元のコード・レベルのままです。Business Process Choreographer Observer アプリケーションを手動で削除し、新しい Business Process Choreographer Explorer レポート作成機能の使用に切り替えるまでは、旧リリースの URL (デフォルトは *host*:*port*/bpcobserver) も正常に動作します。その切り替えは、いつでも都合のよいときに実行できます。

手順

1. マイグレーションのソース・リリースで Business Process Choreographer Observer と Business Process Choreographer Explorer を構成していた場合は、マイグレーション時に JACL テンプレート・スクリプトが生成されます。Business Process Choreographer Explorer レポート作成機能を有効にするには、そのスクリプトを編集して実行する必要があります。
   a. スクリプト・ファイルを見つけます。
      - スタンドアロン・サーバーでは、*profile_root*/ProcessChoreographer/migrate_BPCObserver_*scope*.jaclに生成されます。*scope* の値は、*nodeName_serverName* または *clusterName* のいずれかになります。
      - Network Deployment 環境では、デプロイメント・マネージャーのプロファイルに生成されます。Business Process Choreographer Explorer インスタンスが実行されるすべてのプロファイルのマイグレーションが完了するまでは、スクリプトを実行しないでください。
   b. 生成されたスクリプト・ファイルに記述されている指示に従い、そのスクリプト・ファイルを編集します。
   c. 選択した Business Process Choreographer Explorer インスタンスでレポート作成機能を有効にするために、スクリプト・ファイルに記述されている指示に基づいてカスタマイズしたスクリプトを実行します。
   d. Network Deployment 環境の場合は、レポート・データベースを指すデータ・ソースが存在すること、そのデータ・ソースが Business Process Choreographer Explorer レポート作成機能のスコープ内で可視であることを確認してください。 以前の Business Process Choreographer Observer および Business Process Choreographer Explorer の構成が別のデプロイメント・ターゲットに配置されていた場合、または別々のクラスターにインストールされている Business Process Choreographer Explorer の複数のインスタンスでレポート作成機能を使用可能にする場合は、新しいデータ・ソースを作成する必要があります。
2. Business Process Choreographer Explorer のサーバーが実行中であること、および Business Process Choreographer Explorer アプリケーションが開始されていることを確認します。
3. 既存のクライアントを使用可能にします。 すべてのユーザーに、古い URL (デフォルトは *host*:*port*/bpcobserver) ではなく新しい URL (デフォルトは *host*:*port*/bpc) を使用するように通知するか、 Web サーバーで自動リダイレクトを構成します。
4. クライアントから Business Process Choreographer Explorer レポート作成機能にアクセスできるかどうか、およびその機能が正しく動作しているかどうかをテストします。
5. 旧リリースの Business Process Choreographer Observer 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、以下のようにします。
   a. 管理コンソールで、「**アプリケーション**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere 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. Business Process Choreographer Observer インスタンスを見つけます。名前の先頭は BPCObserver_*scope* になっています。

- 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。

注: コンテキスト・ルートがデフォルトの /bpcobserver でない場合は、アプリケーション名にはコンテキスト・ルート *_contextRoot* も付加されています。

c. Business Process Choreographer Observer アプリケーションをアンインストールするには、削除するアプリケーション・インスタンスを選択してから、「**アンインストール**」 → 「**OK**」 → 「**保管**」をクリックします。

タスクの結果

Business Process Choreographer Explorer レポート作成機能が有効になり、既存のユーザーはその機能にアクセスできるようになりました。旧リリースの Business Process Choreographer Observer アプリケーションは削除されました。

管理コンソールを使用した Business Process Choreographer Explorer レポート作成機能の構成

ここでは、管理コンソールを使用して Business Process Choreographer Explorer レポート作成機能のインスタンスを、特定の Event Collector 用のデータ・ソースに接続するように構成する方法について説明します。

始める前に

Business Process Choreographer Event Collector と Business Process Choreographer Explorer は構成しましたが、Business Process Choreographer Explorer レポート作成機能を構成するためのオプションは選択しませんでした。

手順

1. 管理コンソールで、次のようにして Business Process Choreographer Explorer の構成ページに移動します。 「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Business Process Choreographer Explorer**」をクリックします。
2. レポート作成機能を有効にする Business Process Choreographer Explorer インスタンスを選択します。
3. 「**レポート作成機能の有効化**」のオプションが無効になっている場合は、その機能が既に構成されています。
4. Business Process Choreographer Explorer レポート作成機能用のオプションが使用可能になっている場合は、以下を実行することにより、この機能を構成できます。
   a. Business Process Choreographer イベント・コントローラーがインストールされ、構成されていることを確認します。

b. 「**レポート作成機能の有効化**」を選択します。

c. 視覚化する Business Process Choreographer Event Collector を選択します。リストが空の場合は、最初に Business Process Choreographer イベント・コントローラーをインストールして、構成する必要があります。詳しくは、 241 ページの『Business Process Choreographer Explorer レポート作成機能および Event Collector の構成』を参照してください。

d. 「**スナップショット範囲でレポートする**」で、何日分のデータを視覚化するのかを指定します。

5. 「**適用**」をクリックします。 進行状況を示すメッセージが表示されます。

6. オプション: 問題が報告されている場合は、`SystemOut.log` ファイルを確認します。

タスクの結果

Business Process Choreographer Explorer レポート作成機能が構成され、いつでも使用できます。

次のタスク

同一または別のデプロイメント・ターゲットにさらに多くの Business Process Choreographer Explorer レポート作成機能のインスタンスを構成できますが、各インスタンスは異なる Event Collector データ・ソースに接続する必要があります。

Business Process Choreographer Explorer レポート作成機能の構成パラメーターの変更

Business Process Choreographer Explorer レポート作成機能アプリケーションと Event Collector アプリケーションの構成パラメーターを調整することは、検査を使用可能にし、パフォーマンスを向上させるために重要です。

デフォルト値の変更

デフォルト値は、テスト・システムよりも実動システムに適しています。開発またはテストを行うために Business Process Choreographer をセットアップする場合は、その構成が機能することを検証する前に以下の構成パラメーターを変更するのが適切です。

- BPCEventTransformerEventCount の値をゼロに変更する。
- BPCEventTransformerToleranceTime の値をゼロに変更する。

このように値を変更することにより、実動システムの場合よりも低い率でイベントが発行される場合でも、それらのイベントを 1 分以内に確実に使用可能にすることができます。

Event Collector の構成パラメーター

数値パラメーターを調整すると、イベント変換プログラムが起動される頻度、および Business Process Choreographer Explorer レポート作成機能でイベントが使用可能になるときの経過時間に影響します。

構成パラメーター	データ型/単位	デフォルト値	説明
ObserverSchemaName	ストリング	設定なし	これにより、すべてのデータベース・オブジェクトのプレフィックスとして使用されるデータベース・スキーマを識別します。空のままにすると、デフォルトにより、データベースへの接続で使用されるユーザー ID がプレフィックスとして使用されます。このユーザー ID は、データ・ソース定義の一部として管理コンソールで設定します。このパラメーターの値を指定する場合は、データ・ソースで指定されるユーザー ID に対して、このスキーマのデータベース・オブジェクトにアクセスするのに十分な権限を与える必要があります。セットアップの作成時にスキーマを指定しなかった場合、あるいはランタイム・ユーザー ID やデータベース・プロバイダーを変更する場合は、このパラメーターを変更する必要があります。

構成パラメーター	データ型/単位	デフォルト値	説明
BPCEventTransformer EventCount	整数/イベント数	500	イベントを Business Process Choreographer Explorer レポート作成機能に適した形式に変換するために、Event Collector が変換プログラムを起動するときまでに収集されるイベントの数。 開発、テスト、および実験を行う場合は、おそらくデフォルト値が高すぎるため、長時間に渡ってイベントを監視できない状態になります。短時間でイベントを監視できるようにするには、この値をゼロに設定してください。その後は、イベントが発生するたびに変換プログラムが起動され、Business Process Choreographer Explorer レポート作成機能で表示可能になります。値をゼロに変更する場合は、まだ変換されていない過去のイベントすべてが、新規イベントの生成と同時に変換されます。実動システムで値をゼロにすることは推奨されません。
BPCEventTransformer MaxWaitTime	整数/分	10	BPCEventTransformer EventCount に指定したイベント数に到達していない場合でも、変換プログラムを起動させるまでに経過させることが可能な最大時間。

構成パラメーター	データ型/単位	デフォルト値	説明
BPCEventTransformer ToleranceTime	整数/分	10	イベントが Business Process Choreographer Explorer レポート作成機能で表示可能になるまでの分単位の最小経過期間。これにより、関連イベントを信頼できる方法で相関させることができます。値をゼロにしないでください。値をゼロにすると、前のイベントが到着する前にイベントが処理される可能性があります。 開発、テスト、および実験的使用の場合は、おそらくデフォルト値が高すぎるため、10 分間新規イベントを監視できなくなります。この値を 1 に設定すると、1 分より長く存続している変換済みイベントすべてが Business Process Choreographer Explorer レポート作成機能で表示可能になります。
ObserverCreateTables	ブール		このパラメーターは、EJB が初めてデータベースに接続する時点で、Business Process Choreographer Explorer レポート作成機能スキーマを作成するかどうかを示します。有効値は「true」および「false」です。例えば、既存のセットアップを再使用するが、データ・ソースは新しいものを使用したい場合は、このパラメーターを有効にすることも無効にすることもできます。

Event Collector が Common Event Infrastructure (CEI) からビジネス関連イベントを受信すると、そのイベントはデータベースに保存されます。しばらく時間が経過し、さらにイベントを受信すると、変換プログラムが開始されます。変換プログラムは、保管されたイベントのバッチ変換を実行し、レポートの生成で使用可能な形式でそれらのイベントを元のデータベースに書き込みます。Business Process Choreographer Explorer レポート作成機能では、変換プログラムによって処理されたイベントのみが使用可能になります。

Event Collector が新規イベントを受信するたびに、次の条件の一方または両方に当てはまる場合、変換プログラムのプロセスが開始されます。

- 変換プログラムが最後に開始されて以後に受信したイベントの数が BPCEventTransformerEventCount の値を超える。
- 変換プログラムが最後に開始されて以後の経過時間が BPCEventTransformerMaxWaitTime の値 (分) を超える。

これらの値を小さくすると、イベントはレポート生成のためにより早く使用可能になりますが、少数のイベントを変換するために余分のコストが生じます。このため、大量のイベントを処理して変換のスループットを改善させることと、レポート・データベースでイベントをできるだけ速く使用可能にする必要性との間で、平衡をとる必要があります。

変換プログラムは、開始されるたびに、BPCEventTransformerToleranceTime の値 (分) を超えたすべてのイベントを処理します。その値以内の直近イベントは処理されません。これは、イベントが発生順に公開されるとは限らないからです。BPCEventTransformerToleranceTime のデフォルト設定では、イベントを受け取って Event Collector テーブルに書き込むまでに 10 分を超えないことを前提としています。

Event Collector の構成パラメーターの変更

Event Collector パラメーターを変更するには、以下を実行します。

1. 272 ページの『setupEventCollector ツール』の説明に従って、このツールを開始して Event Collector をセットアップします。以下のメニューが表示されます。

```
1) Prepare a database for the Event Collector and reporting function
2) Install the Event Collector application
3) Remove the Event Collector application and related objects
4) Change configuration settings of an installed Event Collector
5) Drop the database schema of the Event Collector and reporting function
6) Administer reporting function related user-defined functions

0) Exit Menu
```

2. オプション 4 を選択して、変更可能なパラメーターのリストを表示します。

```
1) BPCEventTransformerEventCount
2) BPCEventTransformerMaxWaitTime
3) BPCEventTransformerToleranceTime
4) ObserverCreateTables
5) ObserverSchemaName

0) Exit Menu
```

3. 変更するパラメーターの番号を選択します。パラメーターの名前、説明、型、単位、および現行値が表示されます。
4. 指定された値を変更するには、新規の値を入力して Enter を押します。新規の値を入力せずに Enter を押すと、パラメーター・リストに戻ります。
5. 別のパラメーターの値を変更する場合は、ステップ 3 から操作を繰り返します。
6. 0 を入力して、リストを終了します。変更を保存するかどうかを尋ねられます。
7. すべての変更を保管するには y と入力し、そうでない場合は n を入力してすべての変更を廃棄します。
8. 変更を有効にするには、BPCECollector アプリケーションを再始動します。

Business Process Choreographer Explorer レポート作成機能の構成パラメーター

ReportAtSnapshotRange パラメーターの値は、スナップショット・レポートの効率に大きな影響を与えます。

構成パラメーターおよび clientconfig.jacl パラメーター	データ型/単位	デフォルト値	説明
ObserverSchemaName -reportSchemaName *schemaName*	ストリング	設定なし	これにより、すべてのデータベース・オブジェクトのプレフィックスとして使用されるデータベース・スキーマを識別します。空のままにすると、デフォルトにより、データベースへの接続で使用されるユーザー ID がプレフィックスとして使用されます。このユーザー ID は、データ・ソース定義の一部として管理コンソールで設定します。このパラメーターの値を指定する場合は、データ・ソースで指定されるユーザー ID に対して、このスキーマのデータベース・オブジェクトにアクセスするのに十分な権限を与える必要があります。セットアップの作成時にスキーマを指定しなかった場合、あるいはランタイム・ユーザー ID やデータベース・プロバイダーを変更する場合は、このパラメーターを変更する必要があります。 これは、Event Collector の値と一致している必要があります。

構成パラメーターおよび clientconfig.jacl パラメーター	データ型/単位	デフォルト値	説明
ReportAtSnapshotRange `-reportAtSnapshotRange` *`number`*	整数/日	`60`	スナップショット・レポートは、スナップショットの限定日時より古いすべてのイベントを評価することによって作成されます。これは、スナップショット・レポートにイベントを組み込むことができる期間を定義します。この期間内に発行されたイベントのみがスナップショット・レポートで評価されます。 この値が高すぎると、大量のイベントを処理する必要が生じ、レポート生成に長い時間がかかる可能性があります。この値は、ビジネス環境におけるプロセス・インスタンスの最大期間に設定するようにしてください。
ObserverCreateTables `-reportCreateTables { true | false }`	ブール	z/OS の DB2 の場合は、`false`。 その他のすべてのデータベース・タイプの場合は、`true`。	このパラメーターは、EJB が初めてデータベースに接続する際に、 Business Process Choreographer Explorer レポート作成機能スキーマを作成するかどうかを示します。有効値は「true」および「false」です。

Business Process Choreographer Explorer レポート作成機能の構成パラメーターの変更

Business Process Choreographer Explorer レポート作成機能 パラメーターを変更するために、`-reportSchemaName` *`schemaName`*、`-reportAtSnapshotRange` *`number`*、および `-reportCreateTables { true | false }` のいずれか 1 つのパラメーターを使用して、clientconfig.jacl スクリプトを実行することができます。これらのパラメーターについて詳しくは、237 ページの『clientconfig.jacl スクリプト・ファイルを使用した Business Process Choreographer Explorer の構成』を参照してください。

setupEventCollector ツール:

setupEventCollector は、Business Process Choreographer Event Collector アプリケーションの対話式での構成または削除、データベースのセットアップ、およびデータベースのユーザー定義関数の管理に使用されます。このツールでは wsadmin スクリプトが使用されます。Business Process Choreographer Explorer レポート作成機能を使用する場合は、Event Collector を構成する必要があります。

場所

このツールは、 Business Process Choreographer の構成スクリプト用サブディレクトリーにあります。 *install_root*/ProcessChoreographer/config

制約事項

- Network Deployment 環境では、デプロイメント管理ノードで、-profileName オプションを使用してデプロイメント・マネージャー・プロファイルを指定してツールを開始する必要があります。
- このツールは英語でのみ提供されています。
- このツールは UNIX System Services で実行する必要があります。

パラメーター

```
[-conntype SOAP | RMI | JMS | NONE]
[-user userID -password password]
[-profileName profileName]
( [-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。スタンドアロン・サーバー環境では、アプリケーション・サーバーが稼働していない場合に -conntype NONE オプションのみを指定します。Network Deployment 環境では、デプロイメント・マネージャーが稼働していない場合に -conntype NONE オプションのみを指定します。

-user *userID* **-password** *password*
管理セキュリティーが使用可能な場合には、ツールを使用するために有効なユーザー ID およびパスワードも指定します。

-profileName *profileName*
デフォルト・プロファイルを構成していない場合は、構成するプロファイルの名前を指定します。

-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。このパラメーターはオプションです。

-cluster *clusterName*
クラスター名 *clusterName*。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 アプリケーションを除去します。このオプションを指定しない場合、デフォルトでアプリケーションが構成されます。

-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、ツール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: -node、-server、および -cluster パラメーターを指定しないと、構成中にデプロイメント・ターゲットの入力を求めるプロンプトが出されます。

例: ツールの開始

このツールを開始して server1 というサーバーを操作するには、以下を入力します。

```
setupEventCollector.sh -server server1
```

「コマンド (Commands)」メニューが表示されます。

```
1) Prepare a database for the Event Collector and reporting function
2) Install the Event Collector application
3) Remove the Event Collector application and related objects
4) Change configuration settings of an installed Event Collector
5) Drop the database schema of the Event Collector and reporting function
6) Administer reporting function related user-defined functions

0) Exit Menu
```

ツールの使用

このツールで特定の作業を行う方法については、以下のトピックで説明します。

関連概念

255 ページの『Business Process Choreographer Explorer レポート作成機能のユーザー定義関数』

Business Process Choreographer Explorer レポート作成機能では、タイム・スライスまたは時間間隔に基づいてレポートを実行できます。これにより、SQL 照会が生成されます。このようなレポートを実行するため、Business Process Choreographer Explorer レポート作成機能ではレポート・データベースに特定のユーザー定義関数 (UDF) がインストールされている必要があります。

Business Process Choreographer Explorer レポート作成機能の確認

Business Process Choreographer Explorer レポート作成機能の構成後に、この機能が正しく動作していることを確認します。

始める前に

Business Process Choreographer Explorer レポート作成機能のデータベースは最初は空です。

手順

1. ビジネス・イベントをいくつか生成します。
   a. ブラウザーで URL `http://host:port/context_root` を開いて、Business Process Choreographer Explorer を開始します。ここで、*host* はアプリケーション・サーバーが実行されているホストの名前です。*port* は、アプリケーション・サーバーのポート番号です (デフォルトは 9080)。*context_root* は通常は `bpc` です。
   b. ビジネス・イベントを生成するアクションをいくつか実行します。例えば、1 つのプロセス・インスタンスを開始します。

2. 「**レポート**」をクリックします。 イベントが表示されない場合は、数分待ってから Event Collector アプリケーションを再始動し、ブラウザー・ビューを最新表示します。

   注: BPCEventTransfomerMaxWaitTime および BPCEventTransfomerToleranceTime のデフォルト値を使用すると、変換プログラムがトリガーされ、Event Collector テーブル内のイベントが処理されて使用可能になるのに十分な時間が経過するまでに、最大で 20 分かかることがあります。パラメーターの変更方法やテスト目的の推奨値など、これらのパラメーターについては、 266 ページの『Business Process Choreographer Explorer レポート作成機能の構成パラメーターの変更』を参照してください。
3. 使用可能であると想定しているイベントが表示されることを確認します。
4. 問題がある場合は、 803 ページの『Business Process Choreographer Explorer レポートのトラブルシューティング』を参照してください。

タスクの結果

Business Process Choreographer Explorer レポート作成機能が稼働しています。

リモート・クライアント・アプリケーションの構成

WebSphere Process Server クライアント・インストール上で稼働するリモート Business Process Choreographer クライアント・アプリケーションを構成します。

始める前に

182 ページの『リモート・クライアント・アプリケーションの計画』を実行済みであり、「単一セル」シナリオと「クロス・セル」シナリオのどちらを作成するかは決まっています。

手順

1. 「単一セル」シナリオの場合、つまり WebSphere Process Server クライアント・インストール済み環境が、そのクライアントの接続先の Business Process Choreographer サーバーまたはクラスターと同じセル内にある場合は、以下の手順を実行します。
   a. WebSphere Process Server クライアントを以下のようにインストールして構成します。
      1) WebSphere Process Server クライアントを、サーバーをインストールせずにインストールします。「**WebSphere Process Server - Client**」オプションを選択し、「**WebSphere Process Server**」オプションは選択しないようにします。

         注: クラスターで WebSphere Process Server クライアントを使用したい場合は、クラスター・メンバーをホストするすべての WebSphere Application Server インストール上に WebSphere Process Server クライアントをインストールする必要があります。
      2) プロファイルが既に存在する場合は、そのプロファイルが SDO 2.1.1 付きの Feature Pack for SCA バージョン 1.0 で拡張されていることを確認します。

3) プロファイルがまだ存在しない場合は、以下の手順を実行します。

a) プロファイル管理ツールを開始して「**SDO 2.1.1 付きの Feature Pack for SCA バージョン 1.0 を適用したカスタム・プロファイル (Custom profile with Feature Pack for SCA Version 1.0 with SDO 2.1.1)**」を選択します。または、manageprofiles コマンド行ツールを使用する場合は、以下のプロファイル・テンプレートを使用します。
install_root/profileTemplates/SCA/managed.sdo

b) プロファイルを WebSphere Process Server セルに統合します。このアクションは、後から addNode コマンドを使用して実行することもできます。

c) 管理コンソールで、WebSphere の「デフォルト」サーバー・テンプレートを使用して、WebSphere Process Server クライアント・ノード上にアプリケーション・サーバーを作成します。

b. オプション: 管理コンソールまたは clientconfig.jacl スクリプトを使用して、WebSphere Process Server クライアント内のアプリケーション・サーバー上に Business Process Choreographer Explorer を構成します。Business Process Choreographer コンテナーのターゲットでは、Business Flow Manager および Human Task Manager をホストする WebSphere Process Server サーバーまたはクラスターを選択するようにしてください。

c. オプション: カスタム・クライアント・アプリケーションをインストールして構成します。

1) WebSphere Process Server クライアント・インストール内のアプリケーション・サーバー上にカスタム・クライアント・アプリケーションをインストールします。

2) カスタム・クライアント・アプリケーションの EJB バインディングを以下のように編集します。

a) 管理コンソールを使用して、「**アプリケーション**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere 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) カスタム・クライアント・アプリケーションをクリックします。

c) 「**参照**」の下の「**EJB 参照**」を選択します。クライアント・アプリケーションによって指定されたリソース参照が表示されます。

d) Business Process Choreographer API EJB の参照を見つけます。ターゲット・リソースのデフォルトのリソース参照名および以下の JNDI 名が表示されます。

```
ejb/BusinessFlowManagerHome   com/ibm/bpe/api/BusinessFlowManagerHome
ejb/HumanTaskManagerHome      com/ibm/task/api/HumanTaskManagerHome
```

e) ターゲット・リソース JNDI 名を、セル内で Business Flow Manager および Human Task Manager API が存在する場所の値に変更します。

- Business Process Choreographer が同じセル内の別のサーバー上に構成されている場合、設定値は次のような構造になります。

```
cell/nodes/nodename/servers/servername/com/ibm/bpe/api/BusinessFlowManagerHome
cell/nodes/nodename/servers/servername/com/ibm/task/api/HumanTaskManagerHome
```

- Business Process Choreographer が同じセル内のクラスター上に構成されている場合、設定値は次のようになります。

```
cell/clusters/clustername/com/ibm/bpe/api/BusinessFlowManagerHome
cell/clusters/clustername/com/ibm/task/api/HumanTaskManagerHome
```

3) 変更を保存して、同期化します。

4) クライアント・アプリケーションを再始動します。

d. リモート成果物ローダー (RAL) のデフォルト構成を使用すると、クライアントとサーバーの間で成果物の非セキュア伝送を実行できます。 この接続のセキュリティーを使用可能にする方法については、『リモート成果物ローダー』を参照してください。

2. Business Process Choreographer が構成されている管理対象サーバーまたはクラスターが存在するセルに WebSphere Process Server クライアントが配置されない「クロスセル」シナリオを使用する場合は、別の Network Deployment セル用のスタンドアロン・プロファイルまたは管理対象プロファイルをホストする任意の WebSphere Application Server インストール済み環境に WebSphere Process Server クライアントをインストールできます。この Network Deployment セルで最低限必要となるのは、WebSphere Application Server Deployment Manager のみです。このような環境に WebSphere Process Server クライアントのインストール済み環境をセットアップして構成し、Business Process Choreographer 構成が存在するセルにアクセスするには、以下の手順を実行します。

a. WebSphere Process Server クライアントを以下のようにインストールして構成します。

1) WebSphere Process Server クライアントを、サーバーをインストールせずにインストールします。「**WebSphere Process Server - Client**」オプションを選択し、「**WebSphere Process Server**」オプションは選択しないようにします。

注: クラスターで WebSphere Process Server クライアントを使用したい場合は、クラスター・メンバーをホストするすべての WebSphere Application Server インストール上に WebSphere Process Server クライアントをインストールする必要があります。

2) プロファイルが既に存在する場合は、そのプロファイルが SDO 2.1.1 付きの Feature Pack for SCA バージョン 1.0 で拡張されていることを確認します。

3) プロファイルがまだ存在せず、**スタンドアロン**・プロファイルを使用する場合は、プロファイル管理ツールを開始して「**SDO 2.1.1 付きの Feature Pack for SCA バージョン 1.0 を適用したアプリケーション・サーバー (Application server with Feature Pack for SCA Version 1.0 with SDO 2.1.1)**」を選択します。または、manageprofiles コマンド行ツールを使用する場合は、以下のプロファイル・テンプレートを使用します。
install_root/profileTemplates/SCA/default.sdo例えば、「wp_profile」という名前のスタンドアロン・プロファイルを拡張するには、manageprofiles ツールを使用して、以下のコマンドを入力します。

```
install_root/bin/manageprofiles.sh -augment -templatePath
install_root/profileTemplates/SCA/default.sdo -profileName wp_profile
```

数分後、拡張が正常に実行されると、以下のメッセージが表示されます。

```
INSTCONFSUCCESS: プロファイルの拡張は正常に終了しました。
(INSTCONFSUCCESS: Profile augmentation succeeded.)
```

4) プロファイルがまだ存在せず、**管理対象**プロファイルを使用する場合は、以下の手順を実行します。

a) プロファイル管理ツールを開始して「**SDO 2.1.1 付きの Feature Pack for SCA バージョン 1.0 を適用したカスタム・プロファイル (Custom profile with Feature Pack for SCA Version 1.0 with SDO 2.1.1)**」を選択します。または、manageprofiles コマンド行ツールを使用する場合は、以下のプロファイル・テンプレートを使用します。
install_root/profileTemplates/SCA/managed.sdo

b) プロファイルを WebSphere Process Server セルに統合します。このアクションは、後から addNode コマンドを使用して実行することもできます。

c) 管理コンソールで、WebSphere の「デフォルト」サーバー・テンプレートを使用して、WebSphere Process Server クライアント・ノード上にアプリケーション・サーバーを作成します。

b. オプション: カスタム・クライアント・アプリケーションをインストールして構成します。

1) カスタム・クライアント・アプリケーションが Business Process Choreographer EJB API を使用することを確認します。

2) WebSphere Process Server クライアント・インストール内のアプリケーション・サーバーまたはクラスター上にカスタム・クライアント・アプリケーションをインストールします。

c. Business Process Choreographer が構成されているクラスターまたはサーバーに接続するための間接名前空間バインディング (複数可) を新規に定義します。

1) クライアント・セル上の管理コンソールを使用して、「**環境**」 → 「**命名**」 → 「**名前空間バインディング**」をクリックします。

2) 「**有効範囲**」で、セルを選択します。

3) クライアント・アプリケーションが Business Flow Manager EJB API と Human Task Manager EJB API のどちらか一方を使用するか、両方を使用するかに応じて、以下のステップを 1 回または 2 回実行して、一方または両方の EJB API の新規バインディングを作成します。

a) 「**新規**」をクリックします。

b) 「**バインディング・タイプ**」で、「**間接**」を選択します。次の画面では、以下のプロパティーを指定します。

i. 固有のバインディング ID 名。Service Component Architecture (SCA) に整合する固有名を自由に選択してかまいませんが、名前空間内のスラッシュを下線文字に置き換えることで、名前空間から有効な名前を得ることができます。例えば、

bpc/*remoteCellName_remoteNode_remoteServer*/com/ibm/bpe/api/BusinessFlowManagerHome

という名前空間は、以下のバインディング ID 名になります。

bpc_*remoteCellName_remoteNode_remoteServer*_com_ibm_bpe_api_BusinessFlowManagerHome

ii. バインディングに使用されるクライアントの名前空間。整合性を保つため、以下の規則に従ってください。

- リモート Business Process Choreographer 構成がサーバー上にある場合: bpc/*remoteCellName_remoteNode_remoteServer*/com/ibm/bpe/api/BusinessFlowManagerHome または bpc/*remoteCellName_remoteNode_remoteServer*/com/ibm/task/api/HumanTaskManagerHome
- リモート Business Process Choreographer 構成がクラスター上にある場合: bpc/*remoteCellName_remoteCluster*/com/ibm/bpe/api/BusinessFlowManagerHome または bpc/*remoteCellName_remoteCluster*/com/ibm/task/api/HumanTaskManagerHome

iii. クライアントの接続先となる Business Process Choreographer の構成が存在する、サーバーまたはクラスターによって使用されるネーミング・サーバーのためのプロバイダー URL プロパティー。例えば、corbaloc:iiop:myremotehostname:2809 などです。ブートストラップ・ポートが、Business Process Choreographer をホストするサーバー (またはクラスター・メンバーの 1 つ) の BOOTSTRAP_ADDRESS と一致するようにしてください。

c) Business Flow Manager API または Human Task Manager API が存在する場所のターゲット・リソース JNDI 名を指定します。

- Business Process Choreographer がスタンドアロン・サーバー上に構成されている場合、設定値は次のような構造になります。

```
com/ibm/bpe/api/BusinessFlowManagerHome
com/ibm/task/api/HumanTaskManagerHome
```

- Business Process Choreographer が Network Deployment 環境で構成されている場合、設定値は次のような構造になります。

```
cell/nodes/nodename/servers/servername/com/ibm/bpe/api/BusinessFlowManagerHome
cell/nodes/nodename/servers/servername/com/ibm/task/api/HumanTaskManagerHome
```

- Business Process Choreographer がクラスター上に構成されている場合、設定値は次のようになります。

```
cell/clusters/clustername/com/ibm/bpe/api/BusinessFlowFlowManagerHome
cell/clusters/clustername/com/ibm/task/api/HumanTaskManagerHome
```

4) 管理コンソールを使用して、クライアント・システムで以下の手順を実行します。

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**」 → ***client_application_name***の次にクライアント・アプリケーションの名前の上で、その次に「**参照**」セクションで、「**EJB 参照**」を選択します。をクリックします。

b) 定義した各名前空間に「**ターゲット・リソース JNDI 名**」フィールドが 1 つずつあります。ステップ 2c3bii (278 ページ) で指定した、Business Flow Manager と Human Task Manager のいずれかまたはその両方の JNDI 名を入力します。

c) 変更を保存して、同期化します。

d) クライアント・アプリケーションを再始動します。

タスクの結果

WebSphere Process Server クライアント・インストールを使用するリモート Business Process Choreographer クライアント・アプリケーションが構成されました。

関連概念

465 ページの『ビジネス・プロセスおよびヒューマン・タスクと対話するためのプログラミング・インターフェースの比較』
ビジネス・プロセスおよびヒューマン・タスクと対話する クライアント・アプリケーションの作成には、Enterprise JavaBeans (EJB)、Web サービス、 Java Message Service (JMS) および Representational State Transfer (REST) サービスなどの 汎用プログラミング・インターフェースを使用できます。これらのインターフェースは、それぞれ特性が異なります。

関連タスク

235 ページの『Business Process Choreographer Explorer の構成』
スクリプトを実行するか、または管理コンソールを使用して、Business Process Choreographer Explorer を構成できます。

465 ページの『第 9 章 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。

548 ページの『セッション Bean のリモート・インターフェースにアクセスする』
ビジネス・プロセスまたはヒューマン・タスク用の EJB クライアント・アプリケーションでは、Bean のリモート・ホーム・インターフェースを介して、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。

Business Process Choreographer の活動化

Business Process Choreographer の構成が完了したら、構成が反映されるサーバーとクラスターを再始動する必要があります。

このタスクについて

Business Process Choreographer を活動状態にするには、以下の手順を実行します。

手順

1. Business Process Choreographer をサーバーで構成した場合は、サーバーを再始動します。
2. Business Process Choreographer をクラスターで構成した場合は、クラスターを再始動します。
3. 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。

4. Business Flow Manager アプリケーションと Human Task Manager アプリケーションが正常に開始していることを確認します。 管理コンソールで「**アプリケーション**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere エンタープライズ・アプリケーション**」をクリックし、名前が BPEContainer_*scope* および TaskContainer_*scope* で始まるアプリケーションが起動したことを確認します。

   アプリケーション・サーバーで Business Process Choreographer を構成した場合は、*scope* の値は *nodeName_serverName*です。またクラスターで Business Process Choreographer を構成した場合は *clusterName* です。

タスクの結果

Business Process Choreographer が稼働しています。

次のタスク

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。

手順

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。

   制約事項: Network Deployment 環境では、インストールできる Business Process Choreographer インストール検査アプリケーションのインスタンスは 1 つだけです。例えば、同じ Network Deployment セル内に 2 つの Business Process Choreographer クラスターが存在する場合、どちらか一方のクラスターにのみ bpcivt.ear アプリケーションをインストールできます。後で別のクラスターにインストールする必要が生じた場合は、最初にインストールしたクラスターからあらかじめアンインストールしておく必要があります。
   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. Business Process Choreographer の構成場所に応じて、以下のいずれかを確認します。
   - アプリケーション・サーバーが稼働している。
   - クラスターの、少なくとも 1 つのメンバーが稼働している。
3. 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ことを確認します。
4. 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。
5. 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。

6. オプション: `bpcivt` アプリケーションを停止して除去します。
7. エラーが発生した場合は、以下のいずれかがその原因である可能性があります。
   - Business Process Choreographer がデータベースにアクセスできない場合は、そのデータベース・システムが実行していること、すべてのデータベース・クライアントが正しく構成されていること、およびデータ・ソースが正しく定義されていることを確認します。データ・ソースのユーザー ID およびパスワードが有効であることを確認します。
   - Business Process Choreographer が入力キューを読み取れない場合は、メッセージング・サービスが実行していることを確認し、JMS プロバイダーおよび JMS リソースが正しく定義されていることを確認します。

タスクの結果

Business Process Choreographer 構成の基本的な機能が作動します。

次のタスク

Business Process Choreographer Explorer、Business Process Choreographer Explorer レポート作成機能、または担当者ディレクトリー・プロバイダーなど、他のオプション・パーツを構成していた場合は、それらを個別にテストする必要があります。

Business Process Choreographer の開始動作に関する説明

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、Business Process Choreographer が使用不可となる理由について説明します。

Business Process Choreographer 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 Business Flow Manager および Human Task Manager が使用不可となる時間は、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。実行中でない関連するエンタープライズ・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。

Business Process Choreographer が構成されているスタンドアロン・ノードの統合

サーバーが開発モードで稼働していない場合は、スタンドアロン・プロファイル内にあるサーバーを、新規デプロイメント・マネージャー・セルに統合できます。

始める前に

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 Derby データベース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。また、メッセージング・エンジン・データベース用のデータベースには、リモートでアクセスできる必要があります。すなわち、このデータベースは Derby Embedded にすることも、FILESTORE にすることもできません。

このタスクについて

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 Network Deployment 環境に統合します。

手順

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 インフォメーション・センター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. 変更を有効にするために、サーバーを始動します。
4. 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。

タスクの結果

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。

次のタスク

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。

第 5 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。

手順

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、 288 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。

タスクの結果

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

Business Flow Manager、Human Task Manager、Business Process Choreographer Explorer、およびそれらに関連するリソースをサーバーまたはクラスターから削除します。

始める前に

Business Process Choreographer 構成を削除する前に、次の条件が満たされている必要があります。

- すべてのビジネス・プロセス・インスタンスおよびヒューマン・タスク・インスタンスを削除し、ビジネス・プロセスまたはヒューマン・タスク含むすべてのエンタープライズ・アプリケーションをアンインストールする必要があります。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にオペレーター権限も管理者権限もない場合は、wsadmin `-user` および `-password` のオプションを付けて、オペレーターまたは管理者の権限を持つユーザー ID を指定します。

- 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 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ローカルの Derby データベースは自動的に削除できます。

手順

1. Business Process Choreographer config 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。 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。

   必要な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。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。 このリストを使用すると、手動で削除するデータベースを識別できます。
- 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。
- Business Process Choreographer メッセージング・エンジン・メッセージ・ストアに FILESTORE が使用されている場合は、`bpeunconfig.jacl` スクリプトの `-deleteDB yes` オプションを使用することによっても、関連するディレクトリーが除去されます。
- レポート・データベースを除去するには、 272 ページの『setupEventCollector ツール』の説明に従ってこのツールを開始して Event Collector をセットアップし、オプション「**Drop the database schema of the Event Collector and reporting function**」を選択します。

4. オプション: `bpeunconfig.log` ログ・ファイルを確認します。このファイルは *profile_root* ディレクトリーの `logs` サブディレクトリーにあります。
5. オプション: WebSphere MQ を使用していた場合は、Business Process Choreographer が使用しているキュー・マネージャーを削除します。
6. 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、まだ `bpeunconfig.jacl` スクリプトによって実行されていません。それは、他のコンポーネントがまだそれらの設定を必要としているかどうかを判別できないからです。
   - BusinessCalendar システム・アプリケーションのインストール
   - 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 変数の設定

タスクの結果

Business Process Choreographer アプリケーションおよび関連付けられたリソース (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、アクティベーション・スペック、作業域区画、メール・セッション、および認証別名など) が除去されました。

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、Business Process Choreographer Explorer、および関連するリソースなど、Business Process Choreographer 構成の一部またはすべてを除去します。

始める前に

Business Process Choreographer 構成を削除する前に、ビジネス・プロセスまたはヒューマン・タスクが含まれたすべてのエンタープライズ・アプリケーションをアンインストールする必要があります。

手順

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**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere 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。

      以下で始まるアプリケーション名を探します。

      - `BPEContainer_`*`scope`* は Business Flow Manager アプリケーションです。
      - `TaskContainer_`*`scope`* は Human Task Manager 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。
      - `HTM_PredefinedTasks_V`*`nnn_scope`* および `HTM_PredefinedTaskMsg_V`*`nnn_scope`* は、Business Process Choreographer Business Space 用です。

      nnn はバージョン番号であり、*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・サーバー上に構成されていた場合は、そのサーバーが後でクラスターにプロモートされた場合でも、*scope* の値は *nodeName_serverName* になります。
      - 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: Business Process Choreographer を構成していた場合は、事前定義されたタスク、Business Flow Manager アプリケーション、および Human Task Manager アプリケーションをアンインストールします。

      1) 「`HTM_PredefinedTasks_V`*`nnn_scope`*」および「`HTM_PredefinedTaskMsg_V`*`nnn_scope`*」を選択し、「**アンインストール**」 → 「**OK**」 → 「**保管**」をクリックします。

      2) 「`BPEContainer_`*`scope`*」および「`TaskContainer_`*`scope`*」を選択し、「**アンインストール**」 → 「**OK**」 → 「**保管**」をクリックします。

d. オプション: Business Process Choreographer Explorer を構成していた場合は、構成したすべてのインスタンスをアンインストールします。
   - デフォルトのコンテキスト・ルート /bpc を使用していた場合は、「BPCExplorer_*scope*」を選択し、「**アンインストール**」 → 「**OK**」 → 「**保管**」をクリックします。
   - それ以外の場合は、「BPCExplorer_*scope_context_root*」を選択し、「**アンインストール**」 → 「**OK**」 → 「**保管**」をクリックします。

e. Business Process Choreographer Event Collector を構成していた場合は、Event Collector アプリケーション・インスタンスごとに、294 ページの『管理コンソールを使用した Business Process Choreographer Event Collector の除去』を実行します。

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*) を見つけ、その名前およびそれに関連付けられている認証データ別名 (存在する場合)、および Java Naming and Directory Interface (JNDI) 名 (デフォルト名は jdbc/BPEDB) を書き留めてから、そのデータ・ソースを除去します。

      データ・ソースを検索するには:

      1) 「**リソース**」 → 「**JDBC**」 → 「**データ・ソース**」をクリックします。
      2) 「**有効範囲**」に対して、Business Process Choreographer が構成されたサーバーまたはクラスターを選択します。

   b. オプション: Derby Embedded データベース以外のデータベースの場合は、まだ必要としているデータ・ソースがそれ以上含まれていない限り、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。「**リソース**」 → 「**JDBC**」 → 「**JDBC プロバイダー**」をクリックし、使用しているデータベースに対応する JDBC ドライバーを選択して、「**削除**」をクリックします。

      注: Business Process Choreographer の構成で、Derby Embedded データベース用の組み込みのデフォルト JDBC プロバイダーが使用されている場合は、この JDBC プロバイダーを削除することはできません。

   c. オプション: Business Flow Manager および Human Task Manager の接続ファクトリーおよびキューを除去します。 標準の JNDI 名は次のとおりです。

      Business Flow Manager の接続ファクトリー:

      ```
      jms/BPECF
      jms/BPECFC
      jms/BFMJMSReplyCF
      ```

      Business Flow Manager のキュー:

      ```
      jms/BPEIntQueue
      jms/BPERetQueue
      jms/BPEHldQueue
      jms/BFMJMSAPIQueue
      jms/BFMJMSCallbackQueue
      jms/BFMJMSReplyQueue
      ```

Human Task Manager の接続ファクトリー:

```
jms/HTMCF
```

Human Task Manager のキュー:

```
jms/HTMIntQueue
jms/HTMHldQueue
```

接続ファクトリーおよびキューを削除する方法は、使用している JMS メッセージング・プロバイダーの種類によって異なります。

- 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**」 → 「**接続ファクトリー**」をクリックします。「**有効範囲**」に対して、Business Process Choreographer が構成されたサーバーまたはクラスターを選択します。次に、その接続ファクトリーを選択し、「**削除**」をクリックします。
  2) 「**リソース**」 → 「**JMS**」 → 「**キュー**」をクリックします。「**有効範囲**」に対して、Business Process Choreographer が構成されたサーバーまたはクラスターを選択します。次に、キューを選択し、「**削除**」をクリックします。
- WebSphere MQ の場合は、JMS キュー接続ファクトリーおよび JMS キューを除去します。
  1) 「**リソース**」 → 「**JMS**」 → 「**キュー接続ファクトリー**」をクリックします。「**有効範囲**」に対して、Business Process Choreographer が構成されたサーバーまたはクラスターを選択します。次に、その接続ファクトリーを選択し、「**削除**」をクリックします。
  2) 「**リソース**」 → 「**JMS**」 → 「**キュー**」をクリックします。「**有効範囲**」に対して、Business Process Choreographer が構成されたサーバーまたはクラスターを選択します。次に、キューを選択し、「**削除**」をクリックします。

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、アクティベーション・スペックを削除します。
   1) 「**リソース**」 → 「**JMS**」 → 「**アクティベーション・スペック**」をクリックします。「**有効範囲**」に対して、Business Process Choreographer が構成されたサーバーまたはクラスターを選択します。
   2) 以下のアクティベーション・スペックを削除します。

```
BPEInternalActivationSpec
BFMJMSAS
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、そのサーバーのリスナー・ポートを除去します。
   1) 「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「**サーバー名**」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
   BPEHoldListenerPort
   HTMInternalListenerPort

Business Process Choreographer をクラスターに構成していた場合は、そのクラスターのメンバーごとにこのステップを繰り返します。

f. オプション: 認証データ別名を削除します。

1) **「セキュリティー」 → 「グローバル・セキュリティー」**をクリックし、次に**「認証」**セクションで**「Java 認証・承認サービス」**を展開し、**「JCA 認証データ (JCA authentication data)」**をクリックします。

2) ステップ 2 (289 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。Business Process Choreographer 構成をバージョン 6.0.2.x からマイグレーションしていなかった場合は、その名前は以下のようにデプロイメント・ターゲットによって異なります。

- Business Process Choreographer が、*nodeName* という名前のノードの *serverName* という名前のサーバーに構成されている場合、その名前は通常 BPCDB_*nodeName*.*serverName*_Auth_Alias になります。
- Business Process Choreographer が *clusterName* という名前のクラスターに構成されている場合は、その名前は通常 BPCDB_*clusterName*_Auth_Alias になります。

3) ステップ 2c (289 ページ) で識別された接続ファクトリーのいずれかが認証データ別名を持っている場合は、この別名の除去は以下のようにして慎重に行う必要があります。

- Business Process Choreographer 構成をバージョン 6.0.x で作成**しなかった**場合、その名前は BPC_Auth_Alias で、Network Deployment 環境内のすべての Business Process Choreographer 構成でその名前が共用されます。

  重要: 最後の Business Process Choreographer 構成を除去している場合は、この認証別名のみを除去してください。そうしないと、残りの Business Process Choreographer 構成が作動を停止します。

- Business Process Choreographer 構成をバージョン 6.0.x で作成していた場合は、その名前は通常 *cellName*/BPEAuthDataAliasJMS_*scope* になります。ここで、*cellName* はセルの名前で、*scope* はデプロイメント・ターゲットを識別します。この認証別名を除去しても、他の Business Process Choreographer 構成には影響しません。

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) **「有効範囲」**に対して、Business Process Choreographer が構成されたサーバーまたはクラスターを選択します。

3) 「スケジューラー」ペインで、作業マネージャーの JNDI 名を書き留めておき、BPEScheduler という名前のスケジューラー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**」をクリックします。
   2) 「**有効範囲**」に対して、Business Process Choreographer が構成されたサーバーまたはクラスターを選択します。
   3) 「作業マネージャー」ペインで、ステップ 2g (291 ページ) で書き留めておいた JNDI 名の作業マネージャーを選択して削除します。
   4) さらに、JNDI 名が `wm/BPENavigationWorkManager` の作業マネージャーも削除します。

i. オプション: 作業域区画を除去します。
   1) 「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「**サーバー名**」をクリックし、次に「**コンテナー設定**」セクションで「**ビジネス・プロセス・サービス**」を展開し、「**作業域区画サービス**」をクリックします。
   2) 「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。

   Business Process Choreographer をクラスターに構成していた場合は、そのクラスターのメンバーごとにこのステップを繰り返します。

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**」 → 「**メール・プロバイダー**」をクリックします。
   2) 「**有効範囲**」に対して「**Cell=*cellName***」を選択します。ここで、*cellName* はセルの名前です。
   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**」セクションで「**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。ここで、*scope* は、ステップ 1b (288 ページ) で識別された有効範囲です。

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**」をクリックし、次に「**トポロジー**」セクションで「**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 Business Process Choreographer を構成していた場合は、*nodeName.serverName* または *clusterName* は、Business Process Choreographer が構成されたデプロイメント・ターゲットの名前に一致しません。

c. 「**追加プロパティー**」の下の「**メッセージ・ストア**」を選択します。
   - メッセージ・ストア・タイプが DATASTORE の場合は、データ・ソースの JNDI 名を書き留めておきます。サーバーでは、データ・ソースの JNDI 名は通常 jdbc/com.ibm.ws.sib/*nodeName*.*serverName*-BPC.*cellName*.Bus です。クラスター上では、データ・ソースの JNDI 名は通常 jdbc/com.ibm.ws.sib/*clusterName*-BPC.*cellName*.Bus です。
   - メッセージ・ストア・タイプが FILESTORE の場合は、Log、Permanent store、および Temporary store のパスを書き留めておきます。

d. 「**サービス統合**」 → 「**バス**」 → 「**BPC.*cellName*.Bus**」をクリックし、次に「**トポロジー**」セクションで「**バス・メンバー**」をクリックし、以下の名前のいずれかで識別されるバス・メンバーを除去します。
   - Business Process Choreographer をサーバー上に構成した場合は、*nodeName*:*serverName*。
   - Business Process Choreographer をクラスター上で構成した場合は、*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. ステップ 3c で書き留めたメッセージ・ストア・タイプが DATASTORE の場合は、「**リソース**」 → 「**JDBC**」 → 「**データ・ソース**」をクリックします。メッセージング・エンジンの有効範囲は、Business Process Choreographer が構成されたデプロイメント・ターゲットと同じではない場合があります。必要に応じて、別の有効範囲を試し、ステップ 3c で書き留めた JNDI 名を探してください。 データ・ソースが Derby データベースに対応している場合は、そのデータベースのファイル・システム・パスを書き留めておいてください。Business Process Choreographer をクラスターに構成していた場合は、そのクラスターのメンバーごとにこのステップを繰り返します。

4. BPC_REMOTE_DESTINATION_LOCATION 変数を削除します。 「**環境**」 → 「**WebSphere 変数**」をクリックし、「**有効範囲**」に対して、Business Process Choreographer が構成されたデプロイメント・ターゲットを選択し、BPC_REMOTE_DESTINATION_LOCATION 変数を選択して削除します。
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. アプリケーション・サーバーまたはクラスターを再始動します。
7. オプション: Business Process Choreographer データベースを削除します。
8. オプション: Business Process Choreographer Explorer レポート作成機能を専用のレポート・データベースとともに使用していた場合は、このデータベースを削除します。
9. 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。
10. Business Process Choreographer に WebSphere のデフォルトのメッセージングを使用している場合は、メッセージング・エンジン用のメッセージ・ストアを削除します。これは、そのメッセージ・ストアが再利用できないからです。
    a. ステップ 3c で書き留めたメッセージ・ストア・タイプが FILESTORE の場合は、Log、Permanent store、および Temporary store について書き留めたディレクトリーを除去します。

b. ステップ 3c (293 ページ) で書き留めたメッセージ・ストア・タイプが DATASTORE の場合は、データ・ソースが指しているデータベースを除去します。これが Derby データ・ソースの場合は、ステップ 3f (293 ページ) で書き留めたファイル・システム・パスを削除します。

タスクの結果

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 Explorer レポート作成機能によって必要とされる関連するリソースを除去します。

手順

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**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere 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_*scope* のチェック・ボックスを選択し、「**アンインストール**」 → 「**OK**」をクリックします。ここで、*scope* は、Event Collector が構成されたサーバーまたはクラスターを識別します。
3. 宛先キューを削除します。
   a. 「**サービス統合**」 → 「**バス**」 → 「**CEI.*cellName*.Bus**」をクリックします。
   b. 「**宛先リソース**」の下の「**宛先**」をクリックします。
   c. 以下の宛先キューを選択します。
      - BPCCEIConsumerQueueDestination_*scope*
      - BPCTransformerQueueDestination_*scope*

      ここで、*scope* は、Event Collector が構成されたサーバーまたはクラスターを識別します。
   d. 「**削除**」をクリックします。
4. BFMEvents のサーバー有効範囲を持つイベント・プロファイル・グループを削除します。
   a. 「**サービス統合**」 → 「**Common Event Infrastructure**」 → 「**イベント・サービス**」をクリックします。
   b. 「**追加プロパティー**」の下の「**イベント・サービス**」をクリックします。
   c. 「**デフォルトの Common Event Infrastructure イベント・サーバー (Default Common Event Infrastructure event server)**」をクリックします。
   d. 「**追加プロパティー**」の下の「**イベント・グループ**」をクリックします。
   e. BFMEvents のチェック・ボックスを選択します。
   f. 「**削除**」をクリックします。
5. JMS キュー接続ファクトリーを削除します。

a. 「**リソース**」 → 「**JMS**」 → 「**キュー接続ファクトリー**」をクリックします。

b. 「**有効範囲**」には、Event Collector が構成されたサーバーまたはクラスターを選択します。

c. BPCCEIConsumerQueueConnectionFactory 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。

6. JMS キューを削除します。

a. 「**リソース**」 → 「**JMS**」 → 「**キュー**」をクリックします。

b. 以下のキューのチェック・ボックスを選択します。

- BPCCEIConsumerQueue_*scope*
- BPCTransformerQueue_*scope*

c. 「**削除**」をクリックします。

7. JMS アクティベーション・スペックを削除します。

a. 「**リソース**」 → 「**JMS**」 → 「**アクティベーション・スペック**」をクリックします。

b. 以下のアクティベーション・スペックに対応するチェック・ボックスを選択します。

- BPCCEIConsumerActivationSpec
- BPCTransformerActivationSpec

c. 「**削除**」をクリックします。

8. 構成をバージョン 6.0.2 で作成した場合は、認証データ別名を削除します。

a. 「**セキュリティー**」 → 「**グローバル・セキュリティー**」をクリックし、次に「**認証**」セクションで「**Java 認証・承認サービス**」を展開し、「**JCA 認証データ (JCA authentication data)**」をクリックします。

b. BPCEventCollectorJMSAuthenticationAlias_*scope* を選択します。

c. 「**削除**」をクリックします。

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。

タスクの結果

Business Process Choreographer Event Collector 構成が除去されました。

第 3 部 管理

第 6 章 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。

Business Process Choreographer のクリーンアップ手順

不要になった後、データベースから削除できるランタイム・オブジェクトの概要と、使用可能なツール。

オブジェクトの削除に使用可能な各種のツール

削除対象のオブジェクトの種類に応じて、以下の 1 つ以上のツールを使用できます。

- クリーンアップ・サービス。
- 管理コンソール。
- 管理スクリプト。
- モデル化ツール。
- Failed Event Manager
- Business Process Choreographer Explorer
- Business Process Choreographer API

削除可能なオブジェクトおよび使用するツール

以下の Business Process Choreographer データベース・オブジェクトが不要になった場合は、削除できます。

API でアクセス可能なオブジェクト

プロセス・インスタンス、タスク・インスタンス、および API を使用して実行時に作成されたタスク・テンプレートを、Business Process Choreographer API を使用して削除する、ユーザー独自のクリーンアップ・ツールを作成します。エンタープライズ・アプリケーションの一部になっているテンプレートを API で削除することはできません。API の使用に関する一般情報については、465 ページの『第 9 章 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』を参照してください。

プロセスおよびタスク・テンプレート

テンプレートは以下の方法で削除できます。

- アプリケーションをアンインストールする。
  - 681 ページの『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』
  - 682 ページの『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』
- スクリプトを実行してテンプレートを削除する。
  - 336 ページの『無効になったプロセス・テンプレートの削除』

– 338 ページの『無効になったヒューマン・タスク・テンプレートの削除』

プロセスおよびタスク・インスタンス

インスタンスは以下の方法で削除できます。

- 管理コンソールを使用してクリーンアップ・サービスを構成し、適格なインスタンスを定期的に削除するジョブをスケジュールに入れます。これについては、 311 ページの『クリーンアップ・サービスとクリーンアップ・ジョブの構成』で説明します。
- 完了したインスタンスを削除するスクリプトを実行します。
  – 327 ページの『完了したプロセス・インスタンスの削除』
  – 329 ページの『完了したタスク・インスタンスの削除』
- WebSphere Integration Developer を使用して、以下のように、ビジネス・モデルの適切なプロパティーを設定します。

  ビジネス・プロセスの場合:

  プロパティー Automatically delete the process after completion には `Yes`、`No`、または `On successful completion` の値を設定できます。このプロパティーの値が `No` または `On successful completion` である場合は、プロセス・インスタンスを削除するようにクリーンアップ・ジョブを構成する意味があります。

  ヒューマン・タスクの場合:

  プロパティー Auto deletion mode の値は、`On completion` または `On successful completion` (デフォルト) を取ることができます。削除が実行されるのは、プロパティー Duration until task is deleted の値が Immediate であるか間隔が定義されている場合のみです (この場合に限って Auto deletion mode の値を変更できます)。プロパティー Duration until task is deleted の値が Never である場合は、自動削除が無効になり、Auto deletion mode プロパティーを変更できないため、ヒューマン・タスクを削除するようにクリーンアップ・ジョブを構成する意味があります。それ以外の場合で、Duration until task is deleted の値が Never ではなく、かつ Auto deletion mode の値が `On successful completion` である場合は、正常に完了しなかったヒューマン・タスクを削除するようにクリーンアップ・ジョブを定義する意味があります。
- テンプレートをアンインストールするほか、**-force** オプションを使用してすべてのインスタンスを削除します。このオプションについては、 682 ページの『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』を参照してください。
- 削除するインスタンスが少ない場合は、Business Process Choreographer Explorer を使用すると、インスタンスに関する詳細を確認してから削除できるので便利です。

注: 上記の方法を組み合わせてインスタンスを削除することができます。その場合、インスタンスは最初の削除試行で削除されます。

監査ログ項目

監査ログ項目は、deleteAuditLog.py スクリプトを実行することで削除できます。詳しくは、 325 ページの『管理スクリプトを使用した、監査ログ・エントリーの削除』を参照してください。

レポート作成イベント

レポート作成イベントは、observerDeleteProcessInstanceData.py スクリプトを実行することで削除できます。詳しくは、 333 ページの『レポート・データベースからのデータの削除』を参照してください。

担当者照会

使用していない担当者照会は、cleanupUnusedStaffQueryInstances.py スクリプトを実行することで削除できます。詳しくは、 339 ページの『管理スクリプトを使用した、未使用の担当者照会結果の除去』を参照してください。

保留キュー

処理不能なメッセージは保留キューに格納されます。保留キューには、削除されたインスタンスのメッセージが入っています。保留キュー内のメッセージを再生することにより、削除されたインスタンスのメッセージがすべて破棄され、保留キューを空にすることができます。

- Business Process Choreographer のページを使用したメッセージの再生方法および Failed Event Manager のページの使用については、 307 ページの『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』を参照してください。
- 321 ページの『管理スクリプトを使用した、失敗したメッセージの照会と再生』

関連タスク

311 ページの『クリーンアップ・サービスとクリーンアップ・ジョブの構成』
管理コンソールを使用して、特定の状態にあるビジネス・プロセスとヒューマン・タスクのインスタンスを定期的に削除するクリーンアップ・ジョブを構成し、スケジュールを設定します。

324 ページの『Business Process Choreographer オブジェクトの削除』
各種データベース・オブジェクトは、稼働中のシステムに蓄積されます。これには例えば、監査ログ・エントリー、タスクおよびプロセスのインスタンス、タスクおよびプロセスのテンプレート、レポート作成用データ、担当者照会などがあります。管理スクリプトを定期的に実行して、不要になったオブジェクトを Business Process Choreographer データベースから削除すると、ストレージ・スペースの浪費を防ぐのに役立ちます。

307 ページの『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。

Business Process Choreographer のロギング可能化

ここでは、Business Process Choreographer で Common Event Infrastructure (CEI) イベントを使用可能にする方法について説明します。

始める前に

Business Process Choreographer Explorer レポート作成機能でビジネス・プロセス・イベントをモニターするには、ビジネス・プロセスが Common Event Infrastructure (CEI) イベントを発行できるようにする必要があります。これは、ビジネス・プロセスをモデル化するときに指定します。ビジネス・プロセスを適切にモニターするためには、少なくとも『Process Started』イベントを発行する必要があります。
Business Process Choreographer Explorer レポート作成機能を使用してモニターできる CEI イベントのリストについては、『ビジネス・プロセス・イベント』を参照してください。ビジネス・プロセスが CEI イベントを発行できるようにする方法について詳しくは、WebSphere Integration Developer のインフォメーション・センターを参照してください。

このタスクについて

Business Process Choreographer Event Collector を、Business Process Choreographer が構成されているのと同じデプロイメント・ターゲットにインストールした場合は、setupEventCollector ツールを使用して、アプリケーションのインストール時に CEI ロギングを使用可能にできます。 管理コンソールを使用して Business Process Choreographer Event Collector をインストールした場合は、スクリプトまたは管理コンソールのいずれかを使用して、CEI ロギングを使用可能にする必要があります。

Jython スクリプトを使用して Business Process Choreographer に対する CEI ロギングを使用可能にするには、 316 ページの『スクリプトを使用した Business Process Choreographer のロギング可能化』を実行します。

管理コンソールを使用して Business Process Choreographer に対する CEI ロギングを使用可能にするには、 304 ページの『管理コンソールを使用した Common Base Event、監査証跡、およびタスク履歴の使用可能化』を実行します。

タスクの結果

ビジネス・プロセスおよびアクティビティーに対して Common Event Infrastructure イベントが発行されますが、このイベントは、Business Process Choreographer Event Collector によって受信できます。

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。

関連タスク

316 ページの『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。

Business Process Choreographer Explorer レポート作成機能の使用可能化

ここでは、管理コンソールを使用して Business Process Choreographer Explorer レポート作成機能を使用可能にし、特定の Event Collector 用のデータ・ソースに接続する方法について説明します。

始める前に

Business Process Choreographer Event Collector と Business Process Choreographer Explorer は構成しましたが、Business Process Choreographer Explorer レポート作成機能を構成するためのオプションは選択しませんでした。

このタスクについて

このタスクでは管理コンソールを使用しますが、clientconfig.jacl スクリプト・ファイルを使用して、レポート機能を使用可能にすることもできます。

手順

1. 管理コンソールで、次のようにして Business Process Choreographer Explorer の構成ページに移動します。 「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「**クラスター名**」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「**サーバー名**」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Business Process Choreographer Explorer**」をクリックします。
2. 構成された Business Process Choreographer Explorer インスタンスのリストが表示されます。 レポート機能を使用可能にするインスタンスを選択してください。 Business Process Choreographer Explorer レポート作成機能のオプションが選択済みの場合は、インスタンスが既に構成されています。
3. Business Process Choreographer Explorer レポート作成機能のオプションが選択済みでない場合は、以下を実行してインスタンスを構成します。
   a. Business Process Choreographer イベント・コントローラーがインストールされ、構成されていることを確認します。
   b. 「**レポート作成機能の有効化**」を選択します。
   c. 視覚化する Business Process Choreographer Event Collector を選択します。 リストが空の場合は、最初に Business Process Choreographer イベント・コントローラーをインストールして、構成する必要があります。詳しくは、 241 ページの『Business Process Choreographer Explorer レポート作成機能および Event Collector の構成』を参照してください。
   d. 「**スナップショット範囲でレポートする**」で、何日分のデータを視覚化するのかを指定します。
4. 「**適用**」をクリックします。 進行状況を示すメッセージが表示されます。
5. オプション: 問題が報告されている場合は、`SystemOut.log` ファイルを確認します。

タスクの結果

Business Process Choreographer Explorer レポート作成機能が構成され、いつでも使用できます。

管理コンソールを使用した Common Base Event、監査証跡、およびタスク履歴の使用可能化

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。このタスクを使用すると、Business Space または Task Instance History Representational State Transfer (REST) インターフェースによってタスク履歴データを活用することもできます。

このタスクについて

Business Flow Manager または Human Task Manager 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。変更内容が Business Flow Manager と Human Task Manager の両方に作用するようにするには、両方の設定を別々に変更する必要があります。

管理コンソールを使用した構成済みロギング・インフラストラクチャーの変更

このタスクを使用して、タスク履歴の状態監視者ロギング、監査ログ、または構成の Common Event Infrastructure ロギングを変更します。

このタスクについて

「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。

手順

1. 「Business Flow Manager」または「Human Task Manager」ペインを表示します。
   a. **「サーバー」** → **「クラスター」** → **「WebSphere Application Server クラスター」** → ***「クラスター名」***または**「サーバー」** → **「サーバー・タイプ」** → **「WebSphere Application Server」** → ***「サーバー名」***のいずれかをクリック後、**「構成」**タブの**「ビジネス・インテグレーション」**セクションで、**「Business Process Choreographer」**をクリックします。
   b. 以下のいずれかのオプションを選択します。
      - ビジネス・プロセスの場合は、**「Business Flow Manager」**をクリックします。
      - ヒューマン・タスクの場合は、**「Human Task Manager」**をクリックします。
2. **「構成」**タブの「一般プロパティー」セクションで、使用可能に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。

   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、Business Process Choreographer 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。

タスク履歴を使用可能にする

このオプションは、Human Task Manager のみで使用できます。Business Space でタスク履歴データを表示したり、Task Instance History Representational State Transfer (REST) インターフェースを使用してタスク履歴データを取得したりする場合は、このチェック・ボックスを選択します。

3. 変更を受け入れます。
   a. 「**OK**」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。
4. WebSphere Business Monitor が Service Component Architecture (SCA) イベントをモニターできるようにするには、カスタム・プロパティーを設定する必要があります。
   a. 管理コンソールで、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***cluster_name***」、または 「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***server_name***」 をクリックし、「**ビジネス・インテグレーション**」の下の「**Business Process Choreographer**」を展開して、「**Business Flow Manager**」 → 「**カスタム・プロパティー**」をクリックします。
   b. 「**新規**」をクリックして、新規カスタム・プロパティーを追加します。
   c. 名前 `Compat.SCAMonitoringForBFMAPI` と値 `true` を入力します。
   d. 変更を保存します。 設定は、次にサーバーを再始動するときにアクティブ化されます。

タスクの結果

状態監視者は、必要に応じて設定します。

次のタスク

サーバーを再始動して、変更を有効にします。Business Process Choreographer がクラスター上に構成されている場合は、そのクラスターを再始動します。

管理コンソールを使用した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、タスク履歴の状態監視者ロギング、監査ログ、またはセッションの Common Event Infrastructure ロギングを変更します。

このタスクについて

「**ランタイム**」タブで行った選択は、即時に有効になります。

手順

1. 「Business Flow Manager」または「Human Task Manager」ペインを表示します。

a. 「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」をクリックします。

b. 以下のいずれかのオプションを選択します。

- ビジネス・プロセスの場合は、「**Business Flow Manager**」をクリックします。
- ヒューマン・タスクの場合は、「**Human Task Manager**」をクリックします。

2. 「**ランタイム**」タブの「**一般プロパティー**」セクションで、使用可能に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。

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、Business Process Choreographer 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。

タスク履歴を使用可能にする
このオプションは、Human Task Manager のみで使用できます。Business Space でタスク履歴データを表示したり、Task Instance History Representational State Transfer (REST) インターフェースを使用してタスク履歴データを取得したりする場合は、このチェック・ボックスを選択します。

3. WebSphere Business Monitor が Service Component Architecture (SCA) イベントをモニターできるようにするには、カスタム・プロパティーを設定する必要があります。

a. 管理コンソールで、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***cluster_name***」、または 「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***server_name***」 をクリックし、「**ビジネス・インテグレーション**」の下の「**Business Process Choreographer**」を展開して、「**Business Flow Manager**」 → 「**カスタム・プロパティー**」をクリックします。

b. 「**新規**」をクリックして、新規カスタム・プロパティーを追加します。

c. 名前 `Compat.SCAMonitoringForBFMAPI` と値 `true` を入力します。

4. 「**ランタイム**」タブで加えた変更が、次にサーバーを再始動した後にも有効のままとなるようにしたい場合は、「**構成へのランタイム変更を保管します**」を選択します。

5. 「**OK**」をクリックして変更を受け入れます。

タスクの結果

状態監視者は、必要に応じて設定します。

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。

このタスクについて

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。

手順

1. Business Flow Manager の場合、Failed Event Manager 用の管理コンソール・ページを使用すると、保留キューのメッセージの確認および応答を最も柔軟に実行できます。
   a. 「**統合アプリケーション**」 → 「**Failed Event Manager**」 → 「**失敗したイベントの検索**」をクリックし、「**イベント・タイプ**」で「**BFM 保留**」を選択し、「**OK**」をクリックします。
   b. 検索結果にメッセージが含まれている場合は、いずれかのメッセージを選択してから、「**再サブミット**」をクリックしてメッセージを再生するか、「**削除**」をクリックして再生せずに保留キューからメッセージを削除することができます。
2. Business Process Choreographer の管理コンソール・ページを使用して、保留キューと保存キューにあるメッセージの数を確認し、それらのメッセージを再生するには、以下のようにします。
   a. 「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「**クラスター名**」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「**サーバー名**」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」をクリックします。
   b. 以下のいずれかのオプションを選択します。
      - ビジネス・プロセスの場合は、「**Business Flow Manager**」をクリックします。
      - ヒューマン・タスクの場合は、「**Human Task Manager**」をクリックします。

      保留キューおよび保存キュー内のメッセージの数が、「**ランタイム**」タブの「**一般プロパティー**」に表示されます。
   c. 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: WebSphere 管理セキュリティーが有効になっている場合は、管理者またはオペレーターの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。

タスクの結果

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。

関連概念

299 ページの『Business Process Choreographer のクリーンアップ手順』
不要になった後、データベースから削除できるランタイム・オブジェクトの概要と、使用可能なツール。

関連タスク

307 ページの『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。

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。

このタスクについて

保留キューおよび保存キュー内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。

手順

1. アプリケーション・サーバーまたはクラスターの Business Process Choreographer 管理ページを選択します。

   「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」をクリックします。
2. メッセージ数を最新表示します。
   a. 以下のいずれかのオプションを選択します。
      - ビジネス・プロセスの場合は、「**Business Flow Manager**」をクリックします。
      - ヒューマン・タスクの場合は、「**Human Task Manager**」をクリックします。

   b. 「**ランタイム**」タブで「**メッセージ・カウントの更新**」をクリックします。

タスクの結果

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内のメッセージの数
- ヒューマン・タスクの場合: 保留キュー内のメッセージの数
- 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。

次のタスク

このページで、これらのキュー内のメッセージを再生することもできます。

管理コンソールを使用した担当者照会結果の最新表示

担当者照会の結果は静的です。管理コンソールを使用して、担当者照会を最新表示します。

このタスクについて

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、担当者ディレクトリーに対して評価された担当者照会の結果をキャッシュに入れます。担当者ディレクトリーを変更する場合は、担当者割り当てを再評価するよう強制できます。

手順

担当者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**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」をクリックします。
2. 「**ランタイム**」タブで「**担当者照会の更新**」をクリックします。 すべての担当者照会は、更新されます。

注: WebSphere 管理セキュリティーが有効になっている場合は、管理者またはオペレーターの権限を持つユーザーに対してのみ最新表示ボタンが表示されます。
担当者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。代わりに管理スクリプトの使用を検討してください。

タスクの結果

関連タスク

323 ページの『管理スクリプトを使用した、担当者照会結果の最新表示』
担当者照会の結果は静的です。管理スクリプトを使用して、担当者照会を最新表示します。

更新デーモンを使用した担当者照会結果の最新表示

期限切れのすべての担当者照会結果を定期的に自動で最新表示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。

このタスクについて

担当者照会は、指定された担当者ディレクトリー・プロバイダ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、担当者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。

担当者照会結果を最新に保つために、期限切れの担当者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、キャッシュした担当者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。

手順

1. 次のようにして、Human Task Manager のカスタム・プロパティー・ページを開きます。
   a. 「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」をクリックします。
   b. 以下のいずれかのオプションを選択します。
      - サーバーを再始動することなく設定を変更したい場合は、「**ランタイム**」タブを選択します。
      - 変更を加えて、サーバーを再始動した後にその変更が有効になるようにしたい場合は、「**構成**」タブを選択します。
2. 「**担当者照会リフレッシュ・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担当者照会結果を最新表示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 時に最新表示が実行されます。
3. 「**担当者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、担当者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担当者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、この担当者照会は更新されます。デフォルトは 1 時間です。

4. 「**ランタイム**」タブで加えた変更が、次にサーバーを再始動した後にも有効のままとなるようにしたい場合は、「**構成へのランタイム変更を保管します**」を選択します。
5. 「**OK**」をクリックします。
6. 変更を保存します。「**構成**」タブで加えた変更を有効にするには、アプリケーション・サーバーを再始動します。

新規の有効期限の値は、新規の担当者照会にのみ適用され、既存の担当者照会には適用されません。

関連タスク

323 ページの『管理スクリプトを使用した、担当者照会結果の最新表示』
担当者照会の結果は静的です。管理スクリプトを使用して、担当者照会を最新表示します。

クリーンアップ・サービスとクリーンアップ・ジョブの構成

管理コンソールを使用して、特定の状態にあるビジネス・プロセスとヒューマン・タスクのインスタンスを定期的に削除するクリーンアップ・ジョブを構成し、スケジュールを設定します。

始める前に

クリーンアップ・サービスのスケジュールとして最適な時刻と曜日を特定します。例えば、データベースの負荷が最小になっている日時などが考えられます。さらに、クリーンアップ・サービスで削除するそれぞれのビジネス・プロセスとヒューマン・タスクについて、どの状態のインスタンスを削除候補にするのか、インスタンスがその状態になってからどれだけ時間が経過したら、次に予定されているクリーンアップでそのインスタンスを削除するのかを決定します。

このタスクについて

完了したインスタンスをしばらく保存した後に自動的に削除するように設定します。Business Flow Manager と Human Task Manager について、それぞれ別々のクリーンアップ・サービスを構成します。それぞれについて、まずサービスを有効にし、サービスのパラメーター (スケジュール、クリーンアップの最大期間、データベース・トランザクションのサイズなど) を定義する必要があります。その後、テンプレート・セットのクリーンアップ・ジョブを定義し、どの最終状態で、どれほどの時間が経過したら、インスタンスを削除候補にするのかを定義できます。
Human Task Manager のクリーンアップ・サービスでは、スタンドアロンのヒューマン・タスクのみが削除されますが、Business Flow Manager のクリーンアップ・サービスでは、ビジネス・プロセスが削除されると、すべての子プロセスとそのプロセスに含まれているインライン・ヒューマン・タスクも削除されます。セキュリティーが有効になっている場合は、Business Process Choreographer の構成で指定されているクリーンアップ・ユーザー ID をビジネス管理者ロールに組み込んでおく必要があります。

手順

1. **Business Flow Manager** のクリーンアップ・サービスを構成します。

a. クラスターでクリーンアップ・サービスを構成する場合は、管理コンソールで、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「**クラスター名**」をクリックし、次に「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、**Business Process Choreographer**」をクリックし、「**Business Flow Manager**」をクリックします。

b. スタンドアロン・サーバーでクリーンアップ・サービスを構成する場合は、管理コンソールで「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Servers**」 → 「**サーバー名**」をクリックし、次に「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」をクリックし、「**Business Flow Manager**」をクリックします。

c. 以下のいずれかのオプションを選択します。

- サーバーを再始動することなく設定を変更したい場合は、「**ランタイム**」タブを選択します。
- 変更を加えて、サーバーを再始動した後にその変更が有効になるようにしたい場合は、「**構成**」タブを選択します。

d. 「**追加プロパティー**」セクションで、「**クリーンアップ・サービス設定**」をクリックします。

e. クリーンアップ・サービスが使用可能になっていない場合は、「**クリーンアップ・サービスを使用可能にする**」を選択します。 クラスター構成の場合は、対象のクラスターのいずれか 1 つのクラスター・メンバーで実行されるように、クリーンアップ・サービスのスケジュールが設定されます。

f. 「**頻度**」で、Business Flow Manager のクリーンアップ・サービスを実行する時刻と頻度を指定します。負荷の小さいタイム・スロットの開始時刻を定義する WebSphere の crontab 形式のストリングを入力してください。 例えば、クリーンアップ・サービスを毎夜 11 時に実行する場合は、デフォルト値の `0 0 23 * * ?` を使用します。

g. 「**最大期間**」で、クリーンアップを実行する最大時間を入力します。 デフォルトは 120 分です。最大期間の値は、頻度として指定する時間間隔よりも短くする必要があります。

h. 「**トランザクション・スライス**」で、それぞれのデータベース・トランザクションごとに削除するビジネス・プロセス・インスタンスの数を入力します。 デフォルト値は 10 です。この値はクリーンアップ・サービスのパフォーマンスに影響するため、さまざまな値を試してみることをお勧めします。削除するヒューマン・タスクのサイズによっては、スライスのサイズを大きくすると、パフォーマンスが向上する可能性があります。ただし、トランザクションがタイムアウトになる場合は、値を小さくしてください。

i. 変更を保管します。

2. **Business Flow Manager** の新しいクリーンアップ・ジョブを追加します。

a. 管理コンソールの「**Business Flow Manager**」ページで、「**クリーンアップ・サービス・ジョブ**」をクリックします。

b. 新しいクリーンアップ・ジョブを作成するために、「**追加**」をクリックします。

c. これが唯一のクリーンアップ・ジョブではない場合は、「**順序番号**」で、ジョブの実行順序を示す順序番号を選択できます (番号はゼロから始まります)。

d. 「**クリーンアップ・ジョブ**」で、ジョブの名前を入力します。

e. 「**テンプレート**」で、どのビジネス・プロセス・テンプレートのインスタンス (インライン・ヒューマン・タスクも含む) を削除するかを指定するために 1 つ以上のビジネス・プロセス・テンプレート名を (1 行に 1 つずつ) 入力するか、アスタリスク (*) を入力してすべてのビジネス・プロセス・テンプレートを指定します。

f. 「**次の状態のインスタンスのみをクリーンアップ**」で、以下の状態のうちの 1 つ以上を選択します。

- **FINISHED**
- **TERMINATED**
- **FAILED**

g. 「**削除までの期間**」で、インスタンスが指定の状態のいずれかになってからクリーンアップ・ジョブによる削除候補になるまでの期間を指定します。「**分**」、「**時**」、「**日**」、「**月**」、「**年**」の各フィールドに整数を入力します。デフォルトは 2 時間です。

h. 「**適用**」または「**OK**」をクリックします。

i. 変更を保管します。

j. 必要に応じてこの手順を繰り返し、ビジネス・プロセス・インスタンスのクリーンアップ・ジョブをさらに定義します。

3. **Human Task Manager** のクリーンアップ・サービスを構成します。

a. クラスターでクリーンアップ・サービスを構成する場合は、管理コンソールで、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「**クラスター名**」をクリックし、次に「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、**Business Process Choreographer**」をクリックし、「**Human Task Manager**」をクリックします。

b. スタンドアロン・サーバーでクリーンアップ・サービスを構成する場合は、管理コンソールで「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Servers**」 → 「**サーバー名**」をクリックし、次に「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」をクリックし、「**Human Task Manager**」をクリックします。

c. クリーンアップ・サービスが使用可能になっていない場合は、「**クリーンアップ・サービスを使用可能にする**」を選択します。 クラスター構成の場合は、対象のクラスターのいずれか 1 つのクラスター・メンバーで実行されるように、クリーンアップ・サービスのスケジュールが設定されます。

d. 「**頻度**」で、Human Task Manager のクリーンアップ・サービスを実行する時刻と頻度を指定します。負荷の小さいタイム・スロットを定義した WebSphere の crontab 形式のストリングを入力してください。

ヒント: Business Flow Manager のクリーンアップ・サービスも使用可能になっている場合は、ステップ 1f (312 ページ) および 1g (312 ページ) で指定した値によって定義される時間枠とオーバーラップしないスケジュールを指定

します。例えば、Business Flow Manager のクリーンアップ・サービスが毎夜 1 時に始まり、最大実行時間が 2 時間になっている場合は、`0 0 3 * * ?` という値を入力して、Human Task Manager のクリーンアップ・サービスを毎夜 3 時に実行するように指定できます。

e. 「**最大期間**」で、クリーンアップを実行する最大時間を入力します。 デフォルトは 120 分です。最大期間の値は、頻度として指定する時間間隔よりも短くする必要があります。

f. 「**トランザクション・スライス**」で、それぞれのデータベース・トランザクションごとに削除するヒューマン・タスク・インスタンスの数を入力します。 デフォルト値は 10 です。この値はクリーンアップ・サービスのパフォーマンスに影響するため、さまざまな値を試してみることをお勧めします。削除するヒューマン・タスクのサイズによっては、スライスのサイズを大きくすると、パフォーマンスが向上する可能性があります。ただし、トランザクションがタイムアウトになる場合は、値を小さくしてください。

g. 変更を保管します。

4. **Human Task Manager** の新しいクリーンアップ・ジョブを追加します。

a. 管理コンソールの「**Human Task Manager**」ページで、「**クリーンアップ・ジョブ**」をクリックします。

b. 新しいクリーンアップ・ジョブを作成するために、「**追加**」をクリックします。

c. これが唯一のクリーンアップ・ジョブではない場合は、「**順序番号**」で、ジョブの実行順序を示す順序番号を選択できます (番号はゼロから始まります)。

d. 「**クリーンアップ・ジョブ**」で、ジョブの名前を入力します。

e. 「**テンプレート**」で、どのスタンドアロン・ヒューマン・タスク・テンプレートのインスタンスを削除するかを指定するために 1 つ以上のスタンドアロン・ヒューマン・タスク・テンプレート名を (1 行に 1 つずつ) 入力するか、アスタリスク (*) を入力してすべてのスタンドアロン・ヒューマン・タスク・テンプレートを指定します。 タスク・テンプレートの名前空間を指定する場合は、`myTaskTemplate (http://bpc/samples/task/)` のように、名前空間を括弧に入れて追加します。

注: Human Task Manager のクリーンアップ・サービスでは、Human Task Manager API を使用して開始されたインライン呼び出しタスクも削除できます。

f. 「**次の状態のインスタンスのみをクリーンアップ**」で、以下の状態のうちの 1 つ以上を選択します。

- **FINISHED**
- **TERMINATED**
- **FAILED**
- **INACTIVE**
- **EXPIRED**

g. 「**削除までの期間**」で、インスタンスが指定の状態のいずれかになってからクリーンアップ・ジョブによる削除候補になるまでの期間を指定します。「**分**」、「**時**」、「**日**」、「**月**」、「**年**」の各フィールドに整数を入力します。デフォルトは 2 時間です。

h. 「**適用**」または「**OK**」をクリックします。

i. 変更を保管します。

j. 必要に応じてこの手順を繰り返し、スタンドアロン・ヒューマン・タスク・インスタンスのクリーンアップ・ジョブをさらに定義します。

5. 「**構成**」タブで変更を加えた場合は、サーバーを再始動して、変更をアクティブにします。

タスクの結果

完了したインスタンスを削除するためのクリーンアップ・サービスをアクティブ化して、クリーンアップ・ジョブを定義しました。クリーンアップ・サービスの開始時と終了時には、メッセージ CWWBF0118I と CWWBF0119I が SystemOut.log ファイルに書き込まれます。1 つのクリーンアップ・ジョブの開始時と終了時には、メッセージ CWWBF0116I と CWWBF0117I が SystemOut.log ファイルに書き込まれます。クリーンアップ処理の進行状況の更新は、メッセージ CWWBF0120I と共に SystemOut.log に書き込まれます。

関連概念

299 ページの『Business Process Choreographer のクリーンアップ手順』
不要になった後、データベースから削除できるランタイム・オブジェクトの概要と、使用可能なツール。

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・サーバーの始動時に、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。

このタスクについて

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。クラスターでは、各クラスター・メンバーごとに、常にこのサーバー・レベルのセットアップを実行しなければなりません。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。新規のアプリケーション・サーバーをセットアップする場合、デフォルトで補正サービスが使用可能になっています。

注: 高可用性 (HA) 環境では、複数のサーバーが同じログ・ファイルへのアクセスを試みないように、クラスター内の各サーバーに、固有の補正ログおよびトランザクション・ログ・ディレクトリーが必要です。また、クラスター内の各サーバーは、クラスター内の他のサーバーのトランザクションおよび補正ログ・ディレクトリーにアクセスできなければなりません。補正ログが書き込まれるディレクトリーを変更するには、「**リカバリー・ログ・ディレクトリー**」フィールドにディレクトリーの絶対パス名を入力します。

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。

手順

1. 管理コンソールで、「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「**サーバー名**」をクリックします。
2. 「**構成**」タブの「**コンテナー設定**」の下で、「**コンテナー・サービス**」 → 「**補正サービス**」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。
3. 「**サーバー始動時にサービスを使用可能に設定**」チェック・ボックスが選択されていることを確認します。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。

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。

このタスクについて

管理スクリプトによってサーバーで長期実行処理が発生する場合は、接続タイムアウトの長さが十分でなければ、アクションが完了せずにスクリプトが失敗する可能性があります。接続タイムアウトが原因で wsadmin スクリプト・クライアントが終了する場合は、サーバーの `SystemOut.log` ファイルを調べて、スクリプトを再始動する必要があるかどうかを確認してください。タイムアウトが頻繁に発生する場合は、`soap.client.props` ファイルの com.ibm.SOAP.requestTimeout プロパティーの値を大きくすることを検討してください。一部のスクリプトには、実行する処理の量を調整するために指定できるパラメーターがあります。

Business Process Choreographer の管理スクリプトでは、クロス・セルの処理はサポートされていません。つまり、スクリプト・クライアントから接続できるのは、スクリプトを実行するプロファイルのノードが属しているセルのサーバーまたはデプロイメント・マネージャーに限られます。

関連タスク

302 ページの『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。

スクリプトを使用した Business Process Choreographer のロギング可能化

ここでは、`setStateObserver.py` スクリプトを使用して、 Common Event Infrastructure (CEI)、Business Process Choreographer の監査イベント、あるいは Human Task Manager のタスク・ヒストリー・ロギングを、使用可能または使用不可にする方法を説明します。

場所

setStateObserver.py スクリプトは、Business Process Choreographer の config ディレクトリーにあります。

- *smpe_root*/ProcessChoreographer/config

スクリプトの実行

setStateObserver.py スクリプトを実行するには、次のように入力します。

```
install_root/bin/wsadmin.sh
       -f install_root/ProcessChoreographer/config/setStateObserver.py
```

パラメーター

このスクリプト・ファイルは、以下のパラメーターを取ります。

-bfm
オプションで、使用可能または使用不可の設定が Business Process Choreographer の Business Flow Manager に適用されることを指定します。Business Flow Manager はビジネス・プロセスを実行します。

-cluster *clusterName*
ここで、*clusterName* は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。

-conntype *NONE*
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・マネージャーが稼働していない場合に限定してください。

-enable { CEI | AuditLog | TaskHistory}
オプションで、CEI ロギング、監査ロギング、または Human Task Manager タスク・ヒストリーを使用可能にするかどうかを指定します。複数の値を指定するには、セミコロンを分離文字に使用します。例えば、CEI と監査ロギングを使用可能にするには、 -enable CEI;AuditLog と指定します。値 TaskHistory は、-bfm が指定されている場合は無効です。

-disable { CEI | AuditLog | TaskHistory}
オプションで、CEI ロギング、監査ロギング、または Human Task Manager タスク・ヒストリーを使用不可にするかどうかを指定します。複数の値を指定するには、セミコロンを分離文字に使用します。例えば、CEI と監査ロギングを使用可能にするには、 -enable CEI;AuditLog と指定します。値 TaskHistory は、-bfm が指定されている場合は無効です。

-htm
オプションで、使用可能または使用不可の設定が Business Process Choreographer の Human Task Manager に適用されることを指定します。Human Task Manager はヒューマン・タスクを実行します。

-node *nodeName*
ここで、*nodeName* はノードの名前です。クラスターを指定する場合は、このオプションを指定しないでください。

-profileName *profileName*
この値は、z/OS では常に `default` です。

-server *serverName*
ここで、*serverName* はサーバーの名前です。クラスターを指定する場合は、このオプションを指定しないでください。

例

`server1` のビジネス・プロセス・イベントの CEI ロギングを使用可能にする場合:

```
wsadmin.sh -f setStateObserver.py -server server1 -enable CEI -bfm
```

Business Process Choreographer での時間帯の扱い方

時刻が表示されるか、パラメーターとして渡されるとき、使用される時間帯は、使用されているクライアント、インターフェース、またはパラメーター名によって異なります。

使用しているクライアントに応じて、時刻はクライアントまたはサーバーの地方時でブラウザーに表示されます。

管理スクリプトでは、時刻パラメーターの末尾は接尾部 `Local` または `UTC` です。これは、時刻がスクリプト・クライアントの地方時であると解釈されるか、協定世界時 (UTC) であると解釈されるかを示します。`Local` バージョンの時刻パラメーターを使用すると、時間帯の調整および夏時間調整のために計算を実行する必要がなくなります。

表 28. Business Process Choreographer インターフェースでの時間帯の使用

クライアントまたはインターフェース	使用または表示される時間帯
管理コンソール	サーバーのローカル時間帯
Business Process Choreographer Explorer	クライアントのローカル時間帯
Business Space	クライアントのローカル時間帯
管理スクリプト	UTC またはスクリプト・クライアントの地方時
API	UTC

例えば、deleteCompletedProcessInstances スクリプトには、-validFromUTC、-completedAfterLocal、-completedAfterUTC、-completedBeforeLocal、および -completedBeforeUTC パラメーターのタイム・スタンプ値を指定することができます。パラメーター名の接尾部は、時刻を UTC で指定する必要があるか、スクリプト・クライアントの地方時で指定する必要があるかを示します。

夏時間に従う時間帯では、表示される日時が夏時間調整を適用する期間に含まれる場合、表示される地方時は夏時間に合わせて調整されます。

管理スクリプト・パラメーター -validFromUTC は、異なるテンプレート・バージョン間で区別するために使用され、常に秒単位まで正確に指定する必要があります。時刻を使用するその他のスクリプト・パラメーター (-completedAfterLocal、-

completedAfterUTC、-completedBeforeLocal、および -completedBeforeUTC など) では、時刻を指定せずに日付を指定した場合、時刻はデフォルトの 00:00:00 になります。

スクリプトの実行によりプロセス・インスタンスを新規プロセス・テンプレート・バージョンにマイグレーションする

プロセス・テンプレートの新規バージョンをデプロイした後、新しいプロセス・インスタンスは新規バージョンに基づきますが、古いテンプレート・バージョンに基づく既存のインスタンスは、終了状態になるまで稼働し続けます。migrateProcessInstances.py スクリプトを使用して、実行中のインスタンスをマイグレーションできます。

始める前に

以下の条件を満たしている必要があります。

- テンプレートがデプロイされるアプリケーション・サーバーは稼働していなければなりません。
- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID に管理者権限がない場合は、wsadmin -user および -password のオプションを付けて、管理者権限を持つユーザー ID を指定します。

このタスクについて

migrateProcessInstances.py スクリプトを使用して、特定のプロセス・テンプレート・バージョンのインスタンスを最新バージョンまたは特定のバージョンにマイグレーションします。終了状態 (終了、強制終了、または失敗) にあるインスタンスはマイグレーションされません。指定されたテンプレートのインスタンスのうち、指定された「有効開始日」の値と同じバージョンのもののみがマイグレーションされます。インスタンスをマイグレーションするスクリプトを作成する場合は、MBean インターフェースを使用できます。

手順

1. 管理スクリプトがある 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. 無効になったプロセス・テンプレートのインスタンスをマイグレーションします。 次のコマンドを入力します。

   ```
   install_root/bin/wsadmin.sh -f migrateProcessInstances.py
          [([-node node_name] -server server_name ) | ( -cluster cluster_name )]
          ( -templateName template_name)
          (-sourceValidFromUTC timestamp )
          [(-targetValidFromUTC timestamp )]
          [(-slice slice_size
   ```

   各部の意味は、次のとおりです。

-node *node_name*
サーバー名を指定する場合はオプション。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 *server_name*
Business Process Choreographer が構成され、テンプレートがデプロイされているサーバーの名前。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 *cluster_name*
プロセス・テンプレートがデプロイされているクラスターの名前。このパラメーターは、Business Process Choreographer が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。

-templateName *template_name*
マイグレーションするプロセス・テンプレートの名前。

-sourceValidFromUTC *timestamp*
timestamp は、指定されたテンプレートのうち、どのバージョンのインスタンスをマイグレーションするかを指定します。

timestamp ストリングは、テンプレートが有効になった日付を協定世界時 (UTC) で指定します。その形式は「yyyy-mm-ddThh:mm:ss」(年、月、日、T、時、分、秒) でなければなりません。例えば、`2009-01-31T13:40:50` などです。管理コンソールでは、この日付はサーバーの地方時で表示されるため、必ずサーバーの時間帯を考慮に入れてください。

-targetValidFromUTC *timestamp*
(オプション) 指定されたプロセス・テンプレートのどのバージョンにインスタンスをマイグレーションするかを指定します。このパラメーターが指定されていない場合は、使用可能な最新バージョンのテンプレートが使用されます。*timestamp* ストリングの形式は、sourceValidFromUTC パラメーターと同じです。

-slice *slice_size*
このパラメーターはオプションです。値 *slice_size* は、1 回のトランザクションでマイグレーションされるプロセス・インスタンスの数を指定します。デフォルト値は 10 です。

3. スクリプトを実行すると、マイグレーションが実行されているノードおよびサーバーの名前が出力されます。 サーバーの `SystemOut.log` ファイルを調べて、進行状況情報と、インスタンスのマイグレーションで例外が発生したかどうかを確認してください。例えば、インスタンスが適切な状態でないこと、またはマイグレーション中に問題が発生したことが原因となります。

タスクの結果

インスタンスは新規テンプレート・バージョンにマイグレーションされました。

関連概念

318 ページの『Business Process Choreographer での時間帯の扱い方』
時刻が表示されるか、パラメーターとして渡されるとき、使用される時間帯は、使用されているクライアント、インターフェース、またはパラメーター名によって異なります。

管理スクリプトを使用した、失敗したメッセージの照会と再生

管理スクリプトを使用して、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。

始める前に

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。wsadmin `-conntype none` オプションは使用しないでください。
- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にオペレーター権限がない場合は、wsadmin `-user` および `-password` のオプションを付けて、オペレーター権限を持つユーザー ID を指定します。

このタスクについて

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。

手順

1. 管理スクリプトがある 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。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin.sh -f queryNumberOfFailedMessages.py
             [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
             [ -bfm | -htm ]
   ```

   各部の意味は、次のとおりです。

   -node *nodeName*

   Business Process Choreographer を構成するノードの名前。このパラメーターは、サーバー名を指定する場合のオプションです。デフォルトはローカル・ノードです。

   -server*serverName*

   Business Process Choreographer を構成するサーバーの名前。このパラメーターは、クラスター名を指定しない場合は必須です。

-cluster*clusterName*

Business Process Choreographer を構成するクラスターの名前。このパラメーターは、Business Process Choreographer がクラスターで構成されている場合は必須です。

-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すべてが表示されます。

Business Flow Manager 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。Human Task Manager 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。

ローカル・ノードのサーバーを確認する場合は、次のように入力します。

```
wsadmin -f queryNumberOfFailedMessages.py -server serverName
```

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。

以下のコマンドを入力します。

```
install_root/bin/wsadmin.sh -f replayFailedMessages.py
      (([-node nodeName] -server server_name) | (-cluster cluster_name))
    [ -bfm | -htm ]
      -queue replayQueue
```

各部の意味は、次のとおりです。

-queue *replayQueue*

オプションとして、再生するキューを指定します。*replayQueue* は、以下のいずれか 1 つの値を持つことができます。

`holdQueue` (デフォルト値)

`retentionQueue` (-bfm オプションが指定されている場合にのみ有効)

`both` (-htm オプションが指定されている場合は無効)

-node *nodeName*

Business Process Choreographer を構成するノードの名前。このパラメーターは、サーバー名を指定する場合のオプションです。デフォルトはローカル・ノードです。

-server*serverName*

Business Process Choreographer を構成するサーバーの名前。このパラメーターは、クラスター名を指定しない場合は必須です。

-cluster*clusterName*

Business Process Choreographer を構成するクラスターの名前。このパラメーターは、Business Process Choreographer がクラスターで構成されている場合は必須です。

-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 オプションを指定します。

管理スクリプトを使用した、担当者照会結果の最新表示

担当者照会の結果は静的です。管理スクリプトを使用して、担当者照会を最新表示します。

始める前に

以下の条件を満たしている必要があります。

- サーバー接続が必要になるため、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。
- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にオペレーター権限がない場合は、wsadmin `-user` および `-password` のオプションを付けて、オペレーター権限を持つユーザー ID を指定します。

このタスクについて

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、担当者ディレクトリーに対して評価された担当者照会の結果をキャッシュに入れます。担当者ディレクトリーを変更する場合は、担当者割り当てを再評価するよう強制できます。

手順

1. 管理スクリプトがある 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. 担当者割り当てを再評価するよう強制します。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin.sh -f refreshStaffQuery.py
          [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
          [-processTemplate templateName |
          (-taskTemplate templateName [-nameSpace nameSpace]) |
            -userlist username{,username}...]
   ```

   各部の意味は、次のとおりです。

   -node *nodeName*
   Business Process Choreographer を構成するノードの名前。このパラメーターは、サーバー名を指定する場合のオプションです。デフォルトはローカル・ノードです。

   -server*serverName*
   Business Process Choreographer を構成するサーバーの名前。このパラメーターは、クラスター名を指定しない場合は必須です。

-cluster*clusterName*

Business Process Choreographer を構成するクラスターの名前。このパラメーターは、Business Process Choreographer がクラスターで構成されている場合は必須です。

-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担当者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担当者割り当てが更新されます。更新は、デフォルト・ユーザーに対しては実行されず、タスク・ロールをモデル化するスタッフ照会に対して実行されます。更新が失敗すると、例えばプロセス管理者などのフォールバック・ユーザーの照会は更新されません。

-nameSpace *nameSpace*

タスク・テンプレートの名前空間。

-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担当者割り当てが更新されます。ユーザー・リストは引用符で囲むことができます。引用符を省略する場合は、ユーザー・リストではユーザー名の間にブランクを入れてはなりません。

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担当者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。

3. オプション: スクリプトによってサーバーで長期実行処理が発生する場合は、接続タイムアウトの長さが十分でなければ、アクションが完了せずにスクリプトが失敗する可能性があります。サーバーの `SystemOut.log` ファイルを調べて、スクリプトを再始動する必要があるかどうかを確認してください。タイムアウトが頻繁に発生する場合は、`soap.client.props` ファイルの com.ibm.SOAP.requestTimeout プロパティーの値を大きくするか、スクリプトのパラメーターを調整してサーバーで実行する処理の量を減らすことを検討してください。

関連タスク

309 ページの『管理コンソールを使用した担当者照会結果の最新表示』
担当者照会の結果は静的です。管理コンソールを使用して、担当者照会を最新表示します。

310 ページの『更新デーモンを使用した担当者照会結果の最新表示』
期限切れのすべての担当者照会結果を定期的に自動で最新表示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。

Business Process Choreographer オブジェクトの削除

各種データベース・オブジェクトは、稼働中のシステムに蓄積されます。これには例えば、監査ログ・エントリー、タスクおよびプロセスのインスタンス、タスクおよびプロセスのテンプレート、レポート作成用データ、担当者照会などがありま

す。管理スクリプトを定期的に実行して、不要になったオブジェクトを Business Process Choreographer データベースから削除すると、ストレージ・スペースの浪費を防ぐのに役立ちます。

関連概念

299 ページの『Business Process Choreographer のクリーンアップ手順』
不要になった後、データベースから削除できるランタイム・オブジェクトの概要と、使用可能なツール。

管理スクリプトを使用した、監査ログ・エントリーの削除

管理スクリプトを使用して、Business Flow Manager の監査ログ・エントリーの一部またはすべてを削除します。

始める前に

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。

- 削除する監査ログ・エントリーがあ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。
- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にオペレーター権限がない場合は、wsadmin `-user` および `-password` のオプションを付けて、オペレーター権限を持つユーザー ID を指定します。

このタスクについて

`deleteAuditLog.py` スクリプトを使用すると、データベースから Business Flow Manager の監査ログ・エントリーを削除できます。

手順

1. 管理スクリプトがある 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。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin.sh -f deleteAuditLog.py
              (([-node nodeName] -server server_name) | (-cluster cluster_name))
         ( -all | -timeUTC timestamp | -timeLocal timestamp
                | -processtimeUTC timestamp | -processtimeLocal timestamp )
         [-slice size]
   ```

   各部の意味は、次のとおりです。

   -cluster *cluster_name*
   Business Process Choreographer を構成するクラスターの名前。Business Flow Manager が WebSphere クラスター用に構成されている場合は必須です。

   -node *node_name*
   サーバー名を指定する場合はオプション。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 *server_name*
Business Process Choreographer を構成するサーバーの名前。クラスター名が指定されていない場合は必須です。

-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。

-timeLocal *timestamp*
このオプションを使用して、削除のカットオフ日時 (サーバーの地方時) を指定します。*timestamp* で指定した時刻より古い監査ログ・エントリーのみが削除されます。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` (サーバーの地方時) に設定されます。

-timeUTC *timestamp*
このオプションを使用して、削除のカットオフ日時 (協定世界時 (UTC)) を指定します。*timestamp* で指定した時刻より古い監査ログ・エントリーのみが削除されます。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` (UTC) に設定されます。

-processTimeLocal *timestamp*
このオプションを使用して、削除のカットオフ日時 (サーバーの地方時) を指定します。*timestamp* で指定した時刻より前に完了したプロセスに属する監査ログ・エントリーのみが削除されます。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` (サーバーの地方時) に設定されます。

-processTimeUTC *timestamp*
このオプションを使用して、削除のカットオフ日時 (UTC) を指定します。*timestamp* で指定した時刻より前に完了したプロセスに属する監査ログ・エントリーのみが削除されます。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` (UTC) に設定されます。

-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。

-timeLocal、-timeUTC、-processTimeLocal、および -processTimeUTC の各オプションは相互に排他的です。

3. オプション: スクリプトによってサーバーで長期実行処理が発生する場合は、接続タイムアウトの長さが十分でなければ、アクションが完了せずにスクリプトが失敗する可能性があります。サーバーの `SystemOut.log` ファイルを調べて、スクリプトを再始動する必要があるかどうかを確認してください。タイムアウトが頻繁に発生する場合は、`soap.client.props` ファイルの

com.ibm.SOAP.requestTimeout プロパティーの値を大きくするか、スクリプトのパラメーターを調整してサーバーで実行する処理の量を減らすことを検討してください。

関連概念

318 ページの『Business Process Choreographer での時間帯の扱い方』
時刻が表示されるか、パラメーターとして渡されるとき、使用される時間帯は、使用されているクライアント、インターフェース、またはパラメーター名によって異なります。

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、管理スクリプトを使用します。

始める前に

以下の条件を満たしている必要があります。

- 削除するインスタンスがあ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。サーバー接続が必要になるため、wsadmin `-conntype none` オプションを使用しないでください。
- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にオペレーター権限がない場合は、wsadmin `-user` および `-password` のオプションを付けて、オペレーター権限を持つユーザー ID を指定します。

このタスクについて

最上位のプロセス・インスタンスは、終了状態である finished、terminated、または failed のいずれかになっていれば、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。

手順

1. 管理スクリプトがある 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.sh -f deleteCompletedProcessInstances.py
       (([-node nodeName] -server server_name) | (-cluster cluster_name))
       (-all | [-finished] [-terminated] [-failed] )
       [-templateName templateName
       [-validFromUTC timestamp]]
       [-startedBy userID]
       [(-completedAfterLocal timestamp)|(-completedAfterUTC timestamp)]
       [(-completedBeforeLocal timestamp)|(-completedBeforeUTC timestamp)]
     ```

   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 *server_name*
Business Process Choreographer を構成するサーバーの名前。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 *cluster_name*
Business Process Choreographer を構成するクラスターの名前。Business Process Choreographer がクラスターで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。 `-all` オプションは、すべての終了状態 (完了、強制終了、および失敗) を意味します。`-all` を指定しない場合は、終了状態を 1 つ以上指定する必要があります。

-templateName *templateName*
(オプション) インスタンスが削除されるプロセス・テンプレートの名前を指定します。名前が同じで、`validFrom` 日付が異なるプロセス・テンプレートが複数存在する場合は、validFrom パラメーターを使用して特定のテンプレートを指定しない限り、その名前を持つすべてのプロセス・テンプレートのインスタンスが削除されます。

-validFromUTC *timestamp*
テンプレートが有効になった日付 (協定世界時 (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、`2009-11-20T12:00:00` などです。

-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。

-completedAfterLocal *timestamp*
(オプション) 指定された地方時より後に完了したインスタンスのみを削除することを指定します。*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、`2009-11-20T12:00:00` などです。日付のみを指定すると、時刻はデフォルトで `00:00:00` (サーバーの地方時) になります。

-completedAfterUTC *timestamp*
(オプション) 協定世界時で指定された時刻より後に完了したインスタンスのみを削除することを指定します。*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、`2009-11-20T12:00:00` などです。日付のみを指定すると、時刻はデフォルトで `00:00:00` (UTC) になります。

-completedBeforeLocal *timestamp*
(オプション) 指定された地方時より前に完了したインスタンスのみを削除す

ることを指定します。*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、2009-11-20T12:00:00 などです。日付のみを指定すると、時刻はデフォルトで 00:00:00 (サーバーの地方時) になります。

-completedBeforeUTC *timestamp*
(オプション) 協定世界時で指定された時刻より前に完了したインスタンスのみを削除することを指定します。*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、2009-11-20T12:00:00 などです。日付のみを指定すると、時刻はデフォルトで 00:00:00 (UTC) になります。

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.sh -f deleteCompletedProcessInstances.py
        -node myNode -server myServer
        -finished
        -startedBy Antje
```

3. オプション: スクリプトによってサーバーで長期実行処理が発生する場合は、接続タイムアウトの長さが十分でなければ、アクションが完了せずにスクリプトが失敗する可能性があります。サーバーの SystemOut.log ファイルを調べて、スクリプトを再始動する必要があるかどうかを確認してください。タイムアウトが頻繁に発生する場合は、soap.client.props ファイルの com.ibm.SOAP.requestTimeout プロパティーの値を大きくするか、スクリプトのパラメーターを調整してサーバーで実行する処理の量を減らすことを検討してください。

タスクの結果

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。

関連概念

318 ページの『Business Process Choreographer での時間帯の扱い方』
時刻が表示されるか、パラメーターとして渡されるとき、使用される時間帯は、使用されているクライアント、インターフェース、またはパラメーター名によって異なります。

完了したタスク・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、期限切れ、または失敗) になった最上位のタスク・インスタンスを選択して削除するには、管理スクリプトを使用します。

始める前に

以下の条件を満たしている必要があります。

- 削除するインスタンスがあ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。サーバー接続が必要になるため、wsadmin -conntype none オプションを使用しないでください。
- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。

- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にオペレーター権限がない場合は、wsadmin -user および -password のオプションを付けて、オペレーター権限を持つユーザー ID を指定します。

このタスクについて

最上位のタスク・インスタンスは、終了状態である finished、terminated、expired、または failed のいずれかになっていると、完了したと見なされます。最上位のタスク・インスタンスと、そのすべての関連データ (インスタンス・カスタム・プロパティー、エスカレーション・インスタンス、サブタスク・インスタンス、および後続タスク・インスタンスなど) をデータベースから選択して削除するための基準を指定します。最上位のタスク・インスタンスが forwarded 状態の場合は、その後続タスクが上記終了状態のいずれかになっていると、最上位のタスク・インスタンスは完了したと見なされます。

後続のタスク・インスタンスが後続タスクのチェーンを形成していて、その中の最後のタスク・インスタンスを除くすべてのタスク・インスタンスが forwarded 状態であり、チェーンの最後のタスク・インスタンスがその他の状態である場合があります。この場合、チェーンの最後のタスク・インスタンスが終了状態 finished、terminated、expired、または failed のいずれかになっていると、forwarded 状態にある最上位のタスク・インスタンスは完了したと見なされます。

通常、インライン・タスク・インスタンスは最上位のタスク・インスタンスとは見なされず、deleteCompletedTaskInstances.py スクリプトを使用して削除することはできません。これは、インライン・タスク・インスタンスがビジネス・プロセスに属しているためであり、インライン・タスクが属する完了済みプロセス・インスタンスを削除するには deleteCompletedProcessInstances.py を使用する必要があります。ただし、Human Task Manager API または Service Component Architecture (SCA) API を使用して作成されたインライン呼び出しタスク・インスタンスは、最上位のタスク・インスタンスと見なされ、deleteCompletedTaskInstances.py スクリプトを使用して削除することができます。

手順

1. 管理スクリプトがある 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. データベースからタスク・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.sh -f deleteCompletedTaskInstances.py
       (([-node nodeName] -server serverName) | (-cluster clusterName))
       (-all | [-finished] [-terminated] [-failed] [-expired] )
       [-templateName templateName -nameSpace nameSpace
       [-validFromUTC timestamp]]
       [-createdBy userID ]
       [(-completedAfterLocal timeStamp)|(-completedAfterUTC timeStamp)]
       [(-completedBeforeLocal timeStamp)|(-completedBeforeUTC timeStamp)]
     ```

     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はオプション。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のいずれかを指定できます。

-server *serverName*

Business Process Choreographer を構成するサーバーの名前。クラスター名が指定されていない場合は必須です。

-cluster *clusterName*

Business Process Choreographer を構成するクラスターの名前。Business Process Choreographer がクラスターで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のいずれかを指定できます。

-all | [-finished] [-terminated] [-failed] [-expired]

状態によってどのタスク・インスタンスを削除するかを指定します。`-all` オプションは、すべての終了状態 (`finished`、`terminated`、`failed`、および `expired`) を意味します。`-all` を指定しない場合は、終了状態を 1 つ以上指定する必要があります。

-templateName *templateName*

(オプション) インスタンスが削除されるタスク・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場合は、nameSpace パラメーターも指定する必要があります。名前が同じで、`validFromUTC` 日付が異なるタスク・テンプレートが複数存在する場合は、validFromUTC パラメーターを使用して特定のテンプレートを指定しない限り、その名前を持つすべてのタスク・テンプレートのインスタンスが削除されます。

-nameSpace *nameSpace*

(オプション) 削除するタスク・テンプレートの名前空間を指定します。このオプションを指定する場合は、templateName パラメーターも指定する必要があります。名前が同じで、`validFromUTC` 日付が異なるタスク・テンプレートが複数存在する場合は、validFromUTC パラメーターを使用して特定のテンプレートを指定しない限り、その名前を持つすべてのタスク・テンプレートのインスタンスが削除されます。

-validFromUTC *timestamp*

タスク・テンプレートが有効になる日時。このオプションは、templateName および nameSpace オプションと一緒にしか使用できません。*timestamp* ストリングは協定世界時 (UTC) で指定する必要があり、その形式は「yyyy-MM-ddThh:mm:ss」(年、月、日、T、時、分、秒) です。例えば、`2009-11-20T12:00:00` などです。

-createdBy *userID*

(オプション) 特定のユーザー ID が作成した完了済みタスク・インスタンスのみを削除します。

-completedAfterLocal *timestamp*

(オプション) 指定されたサーバー地方時より後に完了したインスタンスのみを削除することを指定します。最上位のタスク・インスタンスが forwarded 状態にある場合は、その後続タスク・チェーンの最後のタスク・インスタンスが指定の時刻より後に完了していれば、最上位のタスク・インスタンスは指定の時刻より後に完了したと見なされます。*timestamp* ストリングの形式

は -validFromUTC と同じですが、このパラメーターでは時刻の部分はオプションです。日付のみを指定すると、時刻はデフォルトで 00:00:00 (サーバーの地方時) になります。

-completedAfterUTC *timestamp*
(オプション) 指定された UTC 時刻より後に完了したインスタンスのみを削除することを指定します。最上位のタスク・インスタンスが forwarded 状態にある場合は、その後続タスク・チェーンの最後のタスク・インスタンスが指定の時刻より後に完了していれば、最上位のタスク・インスタンスは指定の時刻より後に完了したと見なされます。*timestamp* ストリングの形式は -validFromUTC と同じですが、このパラメーターでは時刻の部分はオプションです。日付のみを指定すると、時刻はデフォルトで 00:00:00 (UTC) になります。

-completedBeforeLocal *timestamp*
(オプション) 指定されたサーバー地方時より前に完了したインスタンスのみを削除することを指定します。最上位のタスク・インスタンスが forwarded 状態にある場合は、その後続タスク・チェーンの最後のタスク・インスタンスが指定の時刻より前に完了していれば、最上位のタスク・インスタンスは指定の時刻より前に完了したと見なされます。*timestamp* ストリングの形式は -validFromUTC と同じですが、このパラメーターでは時刻の部分はオプションです。日付のみを指定すると、時刻はデフォルトで 00:00:00 (サーバーの地方時) になります。

-completedBeforeUTC *timestamp*
(オプション) 指定された UTC 時刻より前に完了したインスタンスのみを削除することを指定します。最上位のタスク・インスタンスが forwarded 状態にある場合は、その後続タスク・チェーンの最後のタスク・インスタンスが指定の時刻より前に完了していれば、最上位のタスク・インスタンスは指定の時刻より前に完了したと見なされます。*timestamp* ストリングの形式は -validFromUTC と同じですが、このパラメーターでは時刻の部分はオプションです。日付のみを指定すると、時刻はデフォルトで 00:00:00 (UTC) になります。

例えば、サーバー *myServer* のノード *myNode* で実行されるタスク・インスタンスのうち、finished 状態にあり、ユーザー Erich が作成したものを削除するには、次のコマンドを実行します。

```
wsadmin.sh -f deleteCompletedTaskInstances.py
        -node myNode -server myServer
        -finished
        -createdBy Erich
```

3. オプション: スクリプトによってサーバーで長期実行処理が発生する場合は、接続タイムアウトの長さが十分でなければ、アクションが完了せずにスクリプトが失敗する可能性があります。サーバーの SystemOut.log ファイルを調べて、スクリプトを再始動する必要があるかどうかを確認してください。タイムアウトが頻繁に発生する場合は、soap.client.props ファイルの com.ibm.SOAP.requestTimeout プロパティーの値を大きくするか、スクリプトのパラメーターを調整してサーバーで実行する処理の量を減らすことを検討してください。

タスクの結果

完了済みのタスク・インスタンスがデータベースから削除されました。

関連概念

318 ページの『Business Process Choreographer での時間帯の扱い方』
時刻が表示されるか、パラメーターとして渡されるとき、使用される時間帯は、使用されているクライアント、インターフェース、またはパラメーター名によって異なります。

レポート・データベースからのデータの削除

Business Process Choreographer Explorer のレポート・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すべてを選択的に削除するには、管理スクリプトを使用します。不要なデータを削除すると、レポート生成時のパフォーマンスが向上します。

始める前に

以下の条件を満たしている必要があります。

- このスクリプトは、接続モードで実行する必要があります。つまり、アプリケーション・サーバーが実行中でなければなりません。wsadmin `-conntype none` オプションは使用しないでください。
- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にオペレーター権限がない場合は、wsadmin `-user` および `-password` のオプションを付けて、オペレーター権限を持つユーザー ID を指定します。

このタスクについて

プロセス・インスタンスのレポート情報は、以下の 3 つの方法でレポート・データベースから削除することができます。

- 指定の時刻より前に終了状態 deleted になったプロセス・インスタンスのレポート作成用データを削除するには、パラメーター `-deletedBeforeUTC` または `-deletedBeforeLocal` のいずれかを指定してください。
- 特定のテンプレート・バージョンのプロセス・インスタンスのレポート作成用データを、その現在の状態に関係なく削除するには、パラメーター `-templateName` と `-validFromUTC` の両方を指定してください。
- 指定の時刻より前に指定の状態になった特定のテンプレート・バージョンのプロセス・インスタンスのレポート作成用データを削除するには、パラメーター `-force -templateName` *`template_name`* `-validFromUTC` *`timestamp`* `-state` *`state`* `-reachedBeforeUTC` *`timestamp`* および `-reachedBeforeUTC` *`timestamp`* または `-reachedBeforeLocal` *`timestamp`* を指定してください。ここで、`-templateName` *`template_name`* および `-validFromUTC` *`timestamp`* はオプションです。

これらの方法を使用するには、以下を実行します。

手順

1. 管理スクリプトがある 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. コマンドを入力して、データベースから、特定のプロセス・インスタンスのレポート作成用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.sh
          -f observerDeleteProcessInstanceData.py
          ( [-node node_name] -server server_name) | (-cluster cluster_name )
          [ -dataSource dataSource_JNDI_name ]
          [ -dbSchemaName dbSchemaName]
          (
            ( (-deletedBeforeLocal timestamp)|(-deletedBeforeUTC timestamp) )
            | ( -templateName template_name -validFromUTC timestamp )
            | ( -force [-templateName template_name -validFromUTC timestamp]
                -state state (-reachedBeforeLocal timestamp)|
                                              (-reachedBeforeUTC timestamp) ) ) )
     ```

     各部の意味は、次のとおりです。

     -node *node_name*
     この名前は、Business Process Choreographer Event Collector を構成するノードを示します。デフォルトはローカル・ノードです。このパラメーターは、server パラメーターを指定した場合はオプションです。 cluster パラメーターを指定した場合は、ノードを指定しないでください。

     -server *server_name*
     Business Process Choreographer Event Collector を構成するサーバーの名前。デフォルトでは、デフォルトのサーバーです。このパラメーターを指定する場合は、cluster パラメーターを指定しないでください。

     -cluster *cluster_name*
     Business Process Choreographer Event Collector を構成するクラスターの名前。このパラメーターを指定する場合は、server パラメーターを指定しないでください。

     -dataSource *datasource_JNDI_name*
     サーバーまたはクラスターは複数のレポート・データベースを所有できるため、コマンドで操作するデータベースをこのパラメーターで指定します。デフォルトは `jdbc/BPEDB` です。

     -dbSchemaName *dbSchemaName*
     このパラメーターは、レポート・データベースが特定のスキーマ名でセットアップされる場合に使用します。

     -deletedBeforeLocal
     指定された地方時より前に状態が deleted になったプロセス・インスタンスのレポート作成用データをすべて削除します。日時 *timestamp* の形式は「*yyyy-MM-dd*[T*hh*:*mm*:*ss*]」(年、月、日、T、時、分、秒) です。例えば、`2008-07-20T12:00:00` などです。年月日だけを指定する場合、時間と分と秒は `00:00:00` (サーバーの地方時) に設定されます。

     -deletedBeforeUTC *timestamp*
     協定世界時 (UTC) で指定された時刻より前に状態が deleted になったプロセス・インスタンスのレポート作成用データをすべて削除します。日時

timestamp の形式は「*yyyy-MM-dd*[T*hh*:*mm*:*ss*]」(年、月、日、T、時、分、秒) です。例えば、2008-07-20T12:00:00 などです。年月日だけを指定する場合、時間と分と秒は 00:00:00 (UTC) に設定されます。

-templateName *template_name*
指定のテンプレート・バージョンに属するインスタンスのレポート作成用データをすべて削除します。

-validFromUTC *timestamp*
templateName オプションを指定する場合は、そのテンプレートで考えられるさまざまなバージョンを区別するためにこのオプションが必要です。日時 *timestamp* は協定世界時 (UTC) で表記します。形式は「*yyyy-MM-dd*[T*hh*:*mm*:*ss*]」(年、月、日、T、時、分、秒) です。例えば、2008-07-20T12:00:00 などです。

-force
すべてのテンプレート、または指定の時刻より前に指定の状態になった指定のテンプレート・バージョンのプロセス・インスタンスのレポート作成用データをすべて強制的に削除します。このオプションを使用する場合は、さらにオプション -state と、-reachedBeforeLocal または -reachedBeforeUTC のいずれかを指定する必要があります。オプション -templateName および -validFromUTC は任意指定です。

-state *state*
状態 running、terminated、suspended、failed、finished、compensated のいずれかを指定します。

-reachedBeforeLocal *timestamp*
カットオフ時刻をサーバーの地方時で指定します。この時刻より前に、指定された状態になっていなければなりません。日時 *timestamp* の形式は「*yyyy-MM-dd*[T*hh*:*mm*:*ss*]」(年、月、日、T、時、分、秒) です。例えば、2008-07-20T12:00:00 などです。年月日だけを指定する場合、時間と分と秒は 00:00:00 (サーバーの地方時) に設定されます。

-reachedBeforeUTC *timestamp*
カットオフ時刻を UTC で指定します。この時刻より前に、指定された状態になっていなければなりません。日時 *timestamp* の形式は「*yyyy-MM-dd*[T*hh*:*mm*:*ss*]」(年、月、日、T、時、分、秒) です。例えば、2008-07-20T12:00:00 などです。年月日だけを指定する場合、時間と分と秒は 00:00:00 (UTC) に設定されます。

例えば、プロセス・テンプレート *my_template* のインスタンスのレポート作成用データのうち、2007 年 1 月 2 日の正午 (UTC) から有効になり、サーバー *my_server* のノード *my_node* で実行され、2007 年 7 月 20 日の正午 (サーバーの地方時) より前に開始されたものをすべて削除するには、以下のコマンドを実行します。

```
wsadmin.sh -f observerDeleteProcessInstanceData.py
        -node my_node -server my_server
        -force -templateName my_template -validFromUTC 2007-01-02T12:00:00
               -state running -reachedBeforeLocal 2007-07-20T12:00:00
```

タスクの結果

正常に実行されれば、ツールから、レポート作成用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。

次のタスク

スクリプトによってサーバーで長期実行処理が発生する場合は、接続タイムアウトの長さが十分でなければ、アクションが完了せずにスクリプトが失敗する可能性があります。サーバーの SystemOut.log ファイルを調べて、スクリプトを再始動する必要があるかどうかを確認してください。タイムアウトが頻繁に発生する場合は、soap.client.props ファイルの com.ibm.SOAP.requestTimeout プロパティーの値を大きくするか、スクリプトのパラメーターを調整してサーバーで実行する処理の量を減らすことを検討してください。

関連概念

318 ページの『Business Process Choreographer での時間帯の扱い方』
時刻が表示されるか、パラメーターとして渡されるとき、使用される時間帯は、使用されているクライアント、インターフェース、またはパラメーター名によって異なります。

無効になったプロセス・テンプレートの削除

管理スクリプトを使用して、Business Process Choreographer データベースから、無効になったビジネス・プロセス・テンプレートを削除します。

始める前に

以下の条件を満たしている必要があります。

- このスクリプトは、接続モードで実行する必要があります。つまり、テンプレートが削除されるアプリケーション・サーバーが稼働していなければなりません。
- オペレーター権限またはデプロイヤー権限のいずれかが必要です。
- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。

このタスクについて

deleteInvalidProcessTemplate.py スクリプトを使用して、データベースから、テンプレートおよび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、対応する有効なアプリケーションがいっさいない場合) 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、構成リポジトリーに保管され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。

手順

1. 管理スクリプトがある 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、無効になったプロセス・テンプレートを削除します。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin.sh -f deleteInvalidProcessTemplate.py
            (([-node nodeName] -server server_name) | (-cluster cluster_name))
            -templateName templateName
            -validFromUTC timestamp
   ```

   各部の意味は、次のとおりです。

   -cluster *cluster_name*
   Business Process Choreographer を構成するクラスターの名前。Business Process Choreographer がクラスターで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 *node_name*
   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 *server_name*
   Business Process Choreographer を構成するサーバーの名前。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。

   -templateName *templateName*
   除去するプロセス・テンプレートの名前。

   -validFromUTC *timestamp*
   テンプレートが有効になった日時 (協定世界時 (UTC) 形式)。ストリングの形式は、「yyyy-MM-ddThh:mm:ss」(年、月、日、T、時、分、秒) にする必要があります。例えば、2005-01-31T13:40:50 のようになります。

3. オプション: スクリプトによってサーバーで長期実行処理が発生する場合は、接続タイムアウトの長さが十分でなければ、アクションが完了せずにスクリプトが失敗する可能性があります。サーバーの SystemOut.log ファイルを調べて、スクリプトを再始動する必要があるかどうかを確認してください。タイムアウトが頻繁に発生する場合は、soap.client.props ファイルの com.ibm.SOAP.requestTimeout プロパティーの値を大きくするか、スクリプトのパラメーターを調整してサーバーで実行する処理の量を減らすことを検討してください。

関連概念

318 ページの『Business Process Choreographer での時間帯の扱い方』
時刻が表示されるか、パラメーターとして渡されるとき、使用される時間帯は、使用されているクライアント、インターフェース、またはパラメーター名によって異なります。

無効になったヒューマン・タスク・テンプレートの削除

管理スクリプトを使用して、Business Process Choreographer データベースから、無効になったヒューマン・タスク・テンプレートを削除します。

始める前に

以下の条件を満たしている必要があります。

- このスクリプトは、接続モードで実行する必要があります。つまり、テンプレートが削除されるアプリケーション・サーバーが稼働していなければなりません。
- オペレーター権限またはデプロイヤー権限のいずれかが必要です。
- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。

このタスクについて

deleteInvalidTaskTemplate.py スクリプトを使用して、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、対応する有効なアプリケーションがいっさいない場合) 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、構成リポジトリーに保管され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。

手順

1. 管理スクリプトがある 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、無効になったヒューマン・タスク・テンプレートを削除します。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin.sh -f deleteInvalidTaskTemplate.py
        (([-node nodeName] -server server_name) | (-cluster cluster_name))
        -templateName templateName
        -validFromUTC timestamp
        -nameSpace nameSpace
   ```

各部の意味は、次のとおりです。

-cluster *cluster_name*
Business Process Choreographer を構成するクラスターの名前。Business Process Choreographer がクラスターで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 *node_name*
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 *server_name*
Business Process Choreographer を構成するサーバーの名前。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。

-templateName *templateName*
除去するタスク・テンプレートの名前。

-validFromUTC *timestamp*
テンプレートが有効になった日時 (協定世界時 (UTC) 形式)。ストリングの形式は、「yyyy-MM-ddThh:mm:ss」(年、月、日、T、時、分、秒) にする必要があります。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名前空間。

3. オプション: スクリプトによってサーバーで長期実行処理が発生する場合は、接続タイムアウトの長さが十分でなければ、アクションが完了せずにスクリプトが失敗する可能性があります。サーバーの `SystemOut.log` ファイルを調べて、スクリプトを再始動する必要があるかどうかを確認してください。タイムアウトが頻繁に発生する場合は、`soap.client.props` ファイルの com.ibm.SOAP.requestTimeout プロパティーの値を大きくするか、スクリプトのパラメーターを調整してサーバーで実行する処理の量を減らすことを検討してください。

関連概念

318 ページの『Business Process Choreographer での時間帯の扱い方』
時刻が表示されるか、パラメーターとして渡されるとき、使用される時間帯は、使用されているクライアント、インターフェース、またはパラメーター名によって異なります。

管理スクリプトを使用した、未使用の担当者照会結果の除去

管理スクリプトを使用して、データベースから未使用の担当者照会結果を削除します。

始める前に

以下の条件を満たしている必要があります。

- サーバー接続が必要になるため、未使用の担当者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。

- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にオペレーター権限がない場合は、wsadmin `-user` および `-password` のオプションを付けて、オペレーター権限を持つユーザー ID を指定します。

このタスクについて

Business Process Choreographer は、評価された担当者照会のランタイム・データベースでユーザー名の名前を維持します。この担当者照会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担当者リストを削除できます。

手順

1. 管理スクリプトがある 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. 未使用の担当者リストを除去します。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin.sh -f cleanupUnusedStaffQueryInstances.py
        ( [-node node_name] -server server_name) | (-cluster cluster_name )
   ```

   各部の意味は、次のとおりです。

   -cluster *clusterName*
   Business Process Choreographer を構成するクラスターの名前。Business Process Choreographer がクラスターで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*
   Business Process Choreographer を構成するサーバーの名前。クラスター名が指定されていない場合は必須です。

タスクの結果

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。

照会テーブルの管理

Query Table Builder を使用して開発した Business Process Choreographer の照会テーブルを管理するには、administrative スクリプト、manageQueryTable.py スクリプトを使用します。すぐに使用可能な定義済み照会テーブルとは異なり、複合照会テーブルおよび補足照会テーブルを照会テーブル API で使用するには、事前にそれらの照会テーブルを WebSphere Process Server 上にデプロイする必要があります。

このタスクについて

照会テーブルのデプロイ時に、照会テーブル定義が Business Process Choreographer データベースに格納されます。WebSphere Process Server の現行バージョンでは、追加のデータベース成果物は作成されません。複合照会テーブルおよび補足照会テーブルに対して行う変更 (デプロイメント、更新、アンデプロイメントを含む) は、サーバーを再始動しなくても照会テーブル API に対して可視です。

照会テーブルは、稼働中のスタンドアロン・サーバー、または少なくとも 1 つのメンバーが稼働中のクラスターにデプロイされます。補足照会テーブルと複合照会テーブルのアンデプロイメントは稼働中のサーバーに対しても実行されます。補足照会テーブルでは、関連する物理データベース・オブジェクト (通常は、データベース・ビューまたはデータベース表) が存在しない場合は、照会テーブルを使用する前にそれらを作成する必要があります。

補足照会テーブルの場合、ユーザーまたは管理者が、関連したデータベース表またはビューを提供する責任があります。

複合照会テーブルの場合、情報は定義済み照会テーブルまたは補足照会テーブルに関連する既存のデータベース表またはビューで構成されます。WebSphere Process Server の現行バージョンでは、データは複写されません。

デプロイ済みの複合照会テーブルによって参照される補足照会テーブルは、更新またはアンデプロイしてはなりません。

manageQueryTable.py を使用して、複合照会テーブルと補足照会テーブルを更新し、それらの XML 定義を取得することができます。システムで使用可能な照会テーブルのリストを取得することもできます。補足照会テーブルでは、関連する物理データベース・オブジェクト (通常は、データベース・ビューまたはデータベース表) が存在しない場合は、照会テーブルを使用する前にそれらを作成する必要があります。

関連資料

『manageQueryTable.py スクリプト』
manageQueryTable.py スクリプトは、Business Process Choreographer の照会テーブルを、デプロイ、アンデプロイ、および更新します。照会テーブルと、照会テーブルの XML 定義をリストする場合にも使用されます。

manageQueryTable.py スクリプト

manageQueryTable.py スクリプトは、Business Process Choreographer の照会テーブルを、デプロイ、アンデプロイ、および更新します。照会テーブルと、照会テーブルの XML 定義をリストする場合にも使用されます。

目的

manageQueryTable.py スクリプトはバッチ・モードで実行されます。このスクリプトは、 Business Process Choreographer の照会テーブルを、デプロイ、アンデプロイ、または更新する場合に使用します。デプロイ済みの照会テーブルのリストや照会テーブルの XML 定義を取得することもできます。

場所

manageQueryTable.py スクリプトは、以下の Business Process Choreographer admin ディレクトリーにあります。

smpe_root/ProcessChoreographer/admin

スタンドアロン・サーバー環境でのスクリプトの実行

構成スクリプトは、 wsadmin コマンドで実行されます。スタンドアロン・サーバー環境の場合:

- このスクリプトは、接続モードで実行する必要があります。つまり、アプリケーション・サーバーが実行中でなければなりません。
- WebSphere 管理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。照会テーブルをデプロイ、アンデプロイ、または更新するには、指定するユーザーが管理者またはデプロイヤーの権限を持っていなければなりません。照会テーブルの XML 定義を表示したり、照会テーブルのリストを取得するには、指定するユーザーが、オペレーター、管理者、デプロイヤーのいずれかの権限を持っている必要があります。

Network Deployment 環境でのスクリプトの実行

構成スクリプトは、 wsadmin コマンドで実行されます。 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。
- このスクリプトは、接続モードで実行する必要があります。つまり、アプリケーション・サーバーまたは少なくとも 1 つのクラスター・メンバーとデプロイメント・マネージャーが実行中でなければなりません。
- WebSphere 管理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。照会テーブルをデプロイ、アンデプロイ、または更新するには、指定するユーザーが管理者またはデプロイヤーの権限を持っていなければなりません。照会テーブルの XML 定義を表示したり、照会テーブルのリストを取得するには、指定するユーザーが、オペレーター、管理者、デプロイヤーのいずれかの権限を持っている必要があります。

スクリプトの実行

以下のコマンドを入力します。

install_root/bin/wsadmin.sh -f manageQueryTable.py *parameters*

パラメーター

以下のパラメーターを指定できます。

```
[ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
( ( -deploy (qtdFile | jarFile)) |
  ( -undeploy queryTableName)  |
  ( -update definition (qtdFile | jarFile)) |
  ( -query names -kind (composite|predefined|supplemental)) |
  ( -query definition -name queryTableName))
```

-node *nodeName*

Business Process Choreographer を構成するノードの名前。このパラメーターは、サーバー名を指定する場合のオプションです。デフォルトはローカル・ノードです。

-server*serverName*

Business Process Choreographer を構成するサーバーの名前。このパラメーターは、クラスター名を指定しない場合は必須です。

-cluster*clusterName*

Business Process Choreographer を構成するクラスターの名前。このパラメーターは、Business Process Choreographer がクラスターで構成されている場合は必須です。

-deploy *qtdFile* | *jarFile*

デプロイする照会テーブル定義 XML ファイルのファイル名、またはその定義が格納されている JAR ファイルのファイル名 (完全修飾パスを含む)。このオプションは、照会テーブルのデプロイに使用します。

-undeploy *queryTableName*

照会テーブルの名前。このオプションは、照会テーブルのアンデプロイに使用します。

-update definition *qtdFile* | *jarFile*

更新する照会テーブル定義 XML ファイルのファイル名、またはその定義が格納されている JAR ファイルのファイル名 (完全修飾パスを含む)。このオプションは、既存の照会テーブルの更新に使用します。

JAR ファイルが提供されている場合、その JAR ファイルには複数の QTD ファイルと、各 QTD ファイルに対して複数のプロパティー・ファイルが含まれている可能性があります。QTD ファイルには、表示名と説明が記述されています。Query Table Builder を使用して、照会テーブル定義を JAR ファイルとしてエクスポートしてください。

-query names -kind {composite|predefined|supplemental}

照会テーブルのタイプ。composite (複合)、predefined (定義済み)、supplemental (補足) のいずれかです。このオプションは、特定タイプのデプロイ済み照会テーブルの名前をリストする場合に使用します。

-query definition -name *queryTableName*

照会テーブルの名前 (大文字)。このオプションは、デプロイ済みの補足照会テーブルまたは複合照会テーブルの XML 定義をリストする場合に使用します。

-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。

関連タスク

『複合照会テーブルおよび補足照会テーブルのデプロイ』
Business Process Choreographer で複合照会テーブルと補足照会テーブルを使用するには、まず manageQueryTable.py スクリプトを使用してそれらの照会テーブルをデプロイしなければなりません。照会テーブル API で照会テーブルを使用するには、関連する Business Process Choreographer コンテナーにそれらの照会テーブルをデプロイする必要があります。デプロイメント後に、照会テーブルを使用可能にするために、照会テーブルを開始したり、サーバーまたはクラスターを再始動したりする必要はありません。

346 ページの『複合照会テーブルおよび補足照会テーブルのアンデプロイ』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルを除去します。

348 ページの『複合照会テーブルおよび補足照会テーブルの更新』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルを更新します。照会テーブルの更新は、アプリケーションの稼働中でも実行できます。更新後に、サーバーまたはクラスターを再始動しなくても、照会テーブルは使用可能です。

350 ページの『照会テーブルのリストの取得』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer で使用可能な照会テーブルのリストを取得します。定義済み照会テーブル、補足照会テーブル、および複合照会テーブルのリストを取得できます。

340 ページの『照会テーブルの管理』
Query Table Builder を使用して開発した Business Process Choreographer の照会テーブルを管理するには、administrative スクリプト、manageQueryTable.py スクリプトを使用します。すぐに使用可能な定義済み照会テーブルとは異なり、複合照会テーブルおよび補足照会テーブルを照会テーブル API で使用するには、事前にそれらの照会テーブルを WebSphere Process Server 上にデプロイする必要があります。

352 ページの『照会テーブルの XML 定義の取得』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルの XML 定義を取得します。このスクリプトで、定義済み照会テーブルの XML 定義を取得することはできません。XML 形式の照会テーブルの定義は、公開されているインターフェースではありません。照会テーブルの定義を手動で変更する操作はサポートされていません。これを実行するには、照会テーブルの定義を Query Table Builder にロードし、そのツールで変更を適用します。

複合照会テーブルおよび補足照会テーブルのデプロイ

Business Process Choreographer で複合照会テーブルと補足照会テーブルを使用するには、まず manageQueryTable.py スクリプトを使用してそれらの照会テーブルをデプロイしなければなりません。照会テーブル API で照会テーブルを使用するには、関連する Business Process Choreographer コンテナーにそれらの照会テーブルをデプロイする必要があります。デプロイメント後に、照会テーブルを使用可能にするために、照会テーブルを開始したり、サーバーまたはクラスターを再始動したりする必要はありません。

始める前に

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。

- 照会テーブルのデプロイ先のアプリケーション・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。
- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID に管理者権限もデプロイヤー権限もない場合は、wsadmin `-user` および `-password` のオプションを付けて、管理者またはデプロイヤーの権限を持つユーザー ID を指定します。

このタスクについて

複合照会テーブルと補足照会テーブルを Business Process Choreographer 内にデプロイするには、次のステップを実行します。

手順

1. manageQueryTable.py スクリプトが配置された Business Process Choreographer サブディレクトリーに移動します。

   smpe_root/ProcessChoreographer/admin

2. 照会テーブルまたは JAR ファイルをデプロイするには、次のコマンドを実行します。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin.sh -f manageQueryTable.py
          [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
          -deploy qtdFile|jarFile
   ```

   各部の意味は、次のとおりです。

   -node *nodeName*
   Business Process Choreographer を構成するノードの名前。このパラメーターは、サーバー名を指定する場合のオプションです。デフォルトはローカル・ノードです。

   -server*serverName*
   Business Process Choreographer を構成するサーバーの名前。このパラメーターは、クラスター名を指定しない場合は必須です。

   -cluster*clusterName*
   Business Process Choreographer を構成するクラスターの名前。このパラメーターは、Business Process Choreographer がクラスターで構成されている場合は必須です。

   -deploy *qtdFile* | *jarFile*
   デプロイする照会テーブル定義 XML ファイルのファイル名、またはその定義が格納されている JAR ファイルのファイル名 (完全修飾パスを含む)。

   以下に例を挙げます。

   ```
   wsadmin.sh -f manageQueryTable.py -server server1 -deploy sample.qtd
   ```

関連タスク

348 ページの『複合照会テーブルおよび補足照会テーブルの更新』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルを更新します。照会テーブルの更新は、アプリケーションの稼働中でも実行できます。更新後に、サーバーまたはクラスターを再始動しなくても、照会テーブルは使用可能です。

350 ページの『照会テーブルのリストの取得』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer で使用可能な照会テーブルのリストを取得します。定義済み照会テーブル、補足照会テーブル、および複合照会テーブルのリストを取得できます。

352 ページの『照会テーブルの XML 定義の取得』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルの XML 定義を取得します。このスクリプトで、定義済み照会テーブルの XML 定義を取得することはできません。XML 形式の照会テーブルの定義は、公開されているインターフェースではありません。照会テーブルの定義を手動で変更する操作はサポートされていません。これを実行するには、照会テーブルの定義を Query Table Builder にロードし、そのツールで変更を適用します。

『複合照会テーブルおよび補足照会テーブルのアンデプロイ』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルを除去します。

関連資料

341 ページの『manageQueryTable.py スクリプト』
manageQueryTable.py スクリプトは、Business Process Choreographer の照会テーブルを、デプロイ、アンデプロイ、および更新します。照会テーブルと、照会テーブルの XML 定義をリストする場合にも使用されます。

複合照会テーブルおよび補足照会テーブルのアンデプロイ

manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルを除去します。

始める前に

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。

- アンデプロイする照会テーブルが存在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。
- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID に管理者権限もデプロイヤー権限もない場合は、wsadmin `-user` および `-password` のオプションを付けて、管理者またはデプロイヤーの権限を持つユーザー ID を指定します。
- アンデプロイする照会テーブルを参照するアプリケーションがインストールされていないこと、および実行されていないことを確認します。補足照会テーブルを

アンデプロイする場合は、補足照会テーブルが複合照会テーブルから接続される照会テーブルとして参照されていてはなりません。

このタスクについて

Business Process Choreographer 内の複合照会テーブルと補足照会テーブルをアンデプロイするには、次のステップを実行します。

手順

1. Business Process Choreographer 管理スクリプトがあるディレクトリーに移動します。

   ```
   smpe_root/ProcessChoreographer/admin
   ```

2. 照会テーブルをアンデプロイするには、次のコマンドを実行します。

   以下のコマンドを入力します。

   ```
   wsadmin.sh -f manageQueryTable.py
          [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
          -undeploy queryTableName
   ```

   各部の意味は、次のとおりです。

   -node *nodeName*
   Business Process Choreographer を構成するノードの名前。このパラメーターは、サーバー名を指定する場合のオプションです。デフォルトはローカル・ノードです。

   -server*serverName*
   Business Process Choreographer を構成するサーバーの名前。このパラメーターは、クラスター名を指定しない場合は必須です。

   -cluster*clusterName*
   Business Process Choreographer を構成するクラスターの名前。このパラメーターは、Business Process Choreographer がクラスターで構成されている場合は必須です。

   -undeploy *queryTableName*
   アンデプロイする照会テーブルの名前 (大文字)。

   以下に例を挙げます。

   ```
   wsadmin -f manageQueryTable.py -server server1 -undeploy COMPANY.SAMPLE
   ```

関連タスク

『複合照会テーブルおよび補足照会テーブルの更新』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルを更新します。照会テーブルの更新は、アプリケーションの稼働中でも実行できます。更新後に、サーバーまたはクラスターを再始動しなくても、照会テーブルは使用可能です。

350 ページの『照会テーブルのリストの取得』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer で使用可能な照会テーブルのリストを取得します。定義済み照会テーブル、補足照会テーブル、および複合照会テーブルのリストを取得できます。

352 ページの『照会テーブルの XML 定義の取得』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルの XML 定義を取得します。このスクリプトで、定義済み照会テーブルの XML 定義を取得することはできません。XML 形式の照会テーブルの定義は、公開されているインターフェースではありません。照会テーブルの定義を手動で変更する操作はサポートされていません。これを実行するには、照会テーブルの定義を Query Table Builder にロードし、そのツールで変更を適用します。

344 ページの『複合照会テーブルおよび補足照会テーブルのデプロイ』
Business Process Choreographer で複合照会テーブルと補足照会テーブルを使用するには、まず manageQueryTable.py スクリプトを使用してそれらの照会テーブルをデプロイしなければなりません。照会テーブル API で照会テーブルを使用するには、関連する Business Process Choreographer コンテナーにそれらの照会テーブルをデプロイする必要があります。デプロイメント後に、照会テーブルを使用可能にするために、照会テーブルを開始したり、サーバーまたはクラスターを再始動したりする必要はありません。

関連資料

341 ページの『manageQueryTable.py スクリプト』
manageQueryTable.py スクリプトは、Business Process Choreographer の照会テーブルを、デプロイ、アンデプロイ、および更新します。照会テーブルと、照会テーブルの XML 定義をリストする場合にも使用されます。

複合照会テーブルおよび補足照会テーブルの更新

manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルを更新します。照会テーブルの更新は、アプリケーションの稼働中でも実行できます。更新後に、サーバーまたはクラスターを再始動しなくても、照会テーブルは使用可能です。

始める前に

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。

- 照会テーブルがデプロイさ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。
- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。

- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID に管理者権限もデプロイヤー権限もない場合は、wsadmin `-user` および `-password` のオプションを付けて、管理者またはデプロイヤーの権限を持つユーザー ID を指定します。

このタスクについて

Business Process Choreographer 内の複合照会テーブルと補足照会テーブルを更新するには、次のステップを実行します。

手順

1. Business Process Choreographer 管理スクリプトがあるディレクトリーに移動します。

   ```
   smpe_root/ProcessChoreographer/admin
   ```

2. 照会テーブル定義 XML ファイルまたはその定義が格納されている JAR ファイルのいずれかを使用して、照会テーブルを更新します。プロパティー・ファイルが既にデプロイされている場合は上書きされます。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin.sh -f manageQueryTable.py
          [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
            -update definition qtdFile|jarFile
   ```

   各部の意味は、次のとおりです。

   -node *nodeName*
   Business Process Choreographer を構成するノードの名前。このパラメーターは、サーバー名を指定する場合のオプションです。デフォルトはローカル・ノードです。

   -server*serverName*
   Business Process Choreographer を構成するサーバーの名前。このパラメーターは、クラスター名を指定しない場合は必須です。

   -cluster*clusterName*
   Business Process Choreographer を構成するクラスターの名前。このパラメーターは、Business Process Choreographer がクラスターで構成されている場合は必須です。

   -update definition *qtdFile* | *jarFile*
   更新する照会テーブル定義 XML ファイルのファイル名、またはその定義が格納されている JAR ファイルのファイル名 (完全修飾パスを含む)。

   以下に例を挙げます。

   ```
   wsadmin -f manageQueryTable.py -server server1
   -update definition sample_v2.qtd
   ```

関連タスク

『照会テーブルのリストの取得』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer で使用可能な照会テーブルのリストを取得します。定義済み照会テーブル、補足照会テーブル、および複合照会テーブルのリストを取得できます。

352 ページの『照会テーブルの XML 定義の取得』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルの XML 定義を取得します。このスクリプトで、定義済み照会テーブルの XML 定義を取得することはできません。XML 形式の照会テーブルの定義は、公開されているインターフェースではありません。照会テーブルの定義を手動で変更する操作はサポートされていません。これを実行するには、照会テーブルの定義を Query Table Builder にロードし、そのツールで変更を適用します。

344 ページの『複合照会テーブルおよび補足照会テーブルのデプロイ』
Business Process Choreographer で複合照会テーブルと補足照会テーブルを使用するには、まず manageQueryTable.py スクリプトを使用してそれらの照会テーブルをデプロイしなければなりません。照会テーブル API で照会テーブルを使用するには、関連する Business Process Choreographer コンテナーにそれらの照会テーブルをデプロイする必要があります。デプロイメント後に、照会テーブルを使用可能にするために、照会テーブルを開始したり、サーバーまたはクラスターを再始動したりする必要はありません。

346 ページの『複合照会テーブルおよび補足照会テーブルのアンデプロイ』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルを除去します。

関連資料

341 ページの『manageQueryTable.py スクリプト』
manageQueryTable.py スクリプトは、Business Process Choreographer の照会テーブルを、デプロイ、アンデプロイ、および更新します。照会テーブルと、照会テーブルの XML 定義をリストする場合にも使用されます。

照会テーブルのリストの取得

manageQueryTable.py スクリプトを使用して、Business Process Choreographer で使用可能な照会テーブルのリストを取得します。定義済み照会テーブル、補足照会テーブル、および複合照会テーブルのリストを取得できます。

始める前に

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。

- 照会テーブルがデプロイさ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。
- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にオペレーター権限、管理者権限、デプロイヤー権限のいずれもない場合は、wsadmin `-user` および `-password` のオプションを付けて、オペレーター、管理者、またはデプロイヤーの権限を持つユーザー ID を指定します。

このタスクについて

Business Process Choreographer 内の照会テーブルのリストを取得するには、次のステップを実行します。

手順

1. Business Process Choreographer 管理スクリプトがあるディレクトリーに移動します。

   ```
   smpe_root/ProcessChoreographer/admin
   ```

2. コマンド・プロンプト・ウィンドウで照会テーブルをリストするには、次のコマンドを実行します。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin.sh -f manageQueryTable.py
          [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
          -query names
          -kind (composite|predefined|supplemental)
   ```

   各部の意味は、次のとおりです。

   -node *nodeName*
   Business Process Choreographer を構成するノードの名前。このパラメーターは、サーバー名を指定する場合のオプションです。デフォルトはローカル・ノードです。

   -server*serverName*
   Business Process Choreographer を構成するサーバーの名前。このパラメーターは、クラスター名を指定しない場合は必須です。

   -cluster*clusterName*
   Business Process Choreographer を構成するクラスターの名前。このパラメーターは、Business Process Choreographer がクラスターで構成されている場合は必須です。

   -kind (composite|predefined|supplemental)
   リストする照会テーブルのタイプ。composite (複合)、predefined (定義済み)、supplemental (補足) のいずれかです。選択した種類の照会テーブルがない場合は、none が返されます。

   以下に例を挙げます。

   ```
   wsadmin.sh -f manageQueryTable.py -server server1
              -query names -kind composite
   ```

関連タスク

348 ページの『複合照会テーブルおよび補足照会テーブルの更新』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルを更新します。照会テーブルの更新は、アプリケーションの稼働中でも実行できます。更新後に、サーバーまたはクラスターを再始動しなくても、照会テーブルは使用可能です。

『照会テーブルの XML 定義の取得』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルの XML 定義を取得します。このスクリプトで、定義済み照会テーブルの XML 定義を取得することはできません。XML 形式の照会テーブルの定義は、公開されているインターフェースではありません。照会テーブルの定義を手動で変更する操作はサポートされていません。これを実行するには、照会テーブルの定義を Query Table Builder にロードし、そのツールで変更を適用します。

344 ページの『複合照会テーブルおよび補足照会テーブルのデプロイ』
Business Process Choreographer で複合照会テーブルと補足照会テーブルを使用するには、まず manageQueryTable.py スクリプトを使用してそれらの照会テーブルをデプロイしなければなりません。照会テーブル API で照会テーブルを使用するには、関連する Business Process Choreographer コンテナーにそれらの照会テーブルをデプロイする必要があります。デプロイメント後に、照会テーブルを使用可能にするために、照会テーブルを開始したり、サーバーまたはクラスターを再始動したりする必要はありません。

346 ページの『複合照会テーブルおよび補足照会テーブルのアンデプロイ』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルを除去します。

関連資料

341 ページの『manageQueryTable.py スクリプト』
manageQueryTable.py スクリプトは、Business Process Choreographer の照会テーブルを、デプロイ、アンデプロイ、および更新します。照会テーブルと、照会テーブルの XML 定義をリストする場合にも使用されます。

照会テーブルの XML 定義の取得

manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルの XML 定義を取得します。このスクリプトで、定義済み照会テーブルの XML 定義を取得することはできません。XML 形式の照会テーブルの定義は、公開されているインターフェースではありません。照会テーブルの定義を手動で変更する操作はサポートされていません。これを実行するには、照会テーブルの定義を Query Table Builder にロードし、そのツールで変更を適用します。

始める前に

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。

- 照会テーブルがデプロイさ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。

- Business Process Choreographer がクラスター上に構成されている場合、少なくとも 1 つのクラスター・メンバーが実行している必要があります。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にオペレーター権限、管理者権限、デプロイヤー権限のいずれもない場合は、wsadmin `-user` および `-password` のオプションを付けて、オペレーター、管理者、またはデプロイヤーの権限を持つユーザー ID を指定します。

このタスクについて

Business Process Choreographer 内の複合照会テーブルと補足照会テーブルの XML 定義を取得するには、以下の手順を実行します。

手順

1. Business Process Choreographer 管理スクリプトがあるディレクトリーに移動します。

   ```
   smpe_root/ProcessChoreographer/admin
   ```

2. コマンド・プロンプト・ウィンドウで照会テーブルの XML 定義のリストを取得するには、次のコマンドを実行します。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin.sh -f manageQueryTable.py
          [ ([-node nodeName] -server serverName) | (-cluster clusterName) ]
          -query definition
          -name queryTableName
   ```

   各部の意味は、次のとおりです。

   -node *nodeName*
   Business Process Choreographer を構成するノードの名前。このパラメーターは、サーバー名を指定する場合のオプションです。デフォルトはローカル・ノードです。

   -server*serverName*
   Business Process Choreographer を構成するサーバーの名前。このパラメーターは、クラスター名を指定しない場合は必須です。

   -cluster*clusterName*
   Business Process Choreographer を構成するクラスターの名前。このパラメーターは、Business Process Choreographer がクラスターで構成されている場合は必須です。

   -name *queryTableName*
   XML 定義をリストする照会テーブルの名前 (大文字)。

   以下に例を挙げます。

   ```
   wsadmin.sh -f manageQueryTable.py -server server1
              -query definition -name COMPANY.SAMPLE
   ```

関連タスク

348 ページの『複合照会テーブルおよび補足照会テーブルの更新』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルを更新します。照会テーブルの更新は、アプリケーションの稼働中でも実行できます。更新後に、サーバーまたはクラスターを再始動しなくても、照会テーブルは使用可能です。

350 ページの『照会テーブルのリストの取得』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer で使用可能な照会テーブルのリストを取得します。定義済み照会テーブル、補足照会テーブル、および複合照会テーブルのリストを取得できます。

344 ページの『複合照会テーブルおよび補足照会テーブルのデプロイ』
Business Process Choreographer で複合照会テーブルと補足照会テーブルを使用するには、まず manageQueryTable.py スクリプトを使用してそれらの照会テーブルをデプロイしなければなりません。照会テーブル API で照会テーブルを使用するには、関連する Business Process Choreographer コンテナーにそれらの照会テーブルをデプロイする必要があります。デプロイメント後に、照会テーブルを使用可能にするために、照会テーブルを開始したり、サーバーまたはクラスターを再始動したりする必要はありません。

346 ページの『複合照会テーブルおよび補足照会テーブルのアンデプロイ』
manageQueryTable.py スクリプトを使用して、Business Process Choreographer 内の複合照会テーブルと補足照会テーブルを除去します。

関連資料

341 ページの『manageQueryTable.py スクリプト』
manageQueryTable.py スクリプトは、Business Process Choreographer の照会テーブルを、デプロイ、アンデプロイ、および更新します。照会テーブルと、照会テーブルの XML 定義をリストする場合にも使用されます。

テンプレートのリスト表示

bpcTemplates.jacl スクリプトには、インストールされたすべてのプロセス・テンプレートおよびタスク・テンプレートと、各テンプレートが有効になる時刻をリスト表示するオプションがあります。

手順

1. 管理スクリプトがある 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. すべてのテンプレートをリストします。

   ```
   install_root/bin/wsadmin.sh -f bpcTemplates.jacl -list [application_name]
   ```

   オプションのアプリケーション名 *application_name* を指定すると、そのアプリケーションに属するテンプレートのみがリストされます。指定しない場合は、デフォルトで、すべてのアプリケーションに属するすべてのテンプレートがリストされます。 テンプレートのリストには、各テンプレートが有効になる時刻が含まれています。以下に例を挙げます。

   ```
   Found 5 template(s):
   JavaSnippetsApp: Process template Process_A, valid from 2008-12-17T11:36:00 (UTC)
   JavaSnippetsApp: Process template Process_B, valid from 2008-12-17T11:36:00 (UTC)
   ```

```
OrderProcessingApp: Process template ChargingProcess, valid from 2009-04-03T19:52:54 (UTC)
OrderProcessingApp: Process template OrderProcess, valid from 2009-04-03T19:52:54 (UTC)
OrderProcessingApp: Process template ShippingProcess, valid from 2009-04-03T19:52:54 (UTC)
Done.
```

関連概念

318 ページの『Business Process Choreographer での時間帯の扱い方』
時刻が表示されるか、パラメーターとして渡されるとき、使用される時間帯は、使用されているクライアント、インターフェース、またはパラメーター名によって異なります。

第 7 章 Business Process Choreographer Explorer の概要

持っているユーザー・ロールに応じて、Business Process Choreographer Explorer を使用してビジネス・プロセスおよびヒューマン・タスクを管理するか、割り当てられたタスクを処理することができます。ビジネス・プロセスおよびタスクの実行中に、WebSphere Process Server はプロセス・インスタンスとそれに関連するアクティビティーの状態変更に関する情報が入ったイベントを発行できます。レポート作成機能を使用して、これらのイベントに基づく統計情報を取得し、プロセスおよびアクティビティーに関するレポートを作成できます。

このタスクについて

Business Process Choreographer Explorer を使用して、以下のタスクを実行できます。

- ビジネス管理者である場合は、ビジネス・プロセスのライフ・サイクルを管理し、ビジネス・プロセスを修復することができます。例えば、単一アクティビティーを再始動または強制的に完了させるか、ビジネス・プロセスを全体として補正できます。補正が失敗した場合、プロセス・インスタンスを再試行、スキップ、または停止できます。さらに、ビジネス・プロセスおよびアクティビティーのカスタム・プロパティーを追加および更新できます。
- ヒューマン・タスク管理者なら、ヒューマン・タスクのライフ・サイクルを管理したり、作業割り当てを管理したりすることができます。例えば、ユーザーに責任を割り当てたり、ユーザーの不在処理や代替ユーザーを管理したりすることができます。また、ヒューマン・タスクの優先順位およびビジネス・カテゴリーを変更したり、カスタム・プロパティーを追加または更新したりすることもできます。
- Business Process Choreographer Explorer のレポート機能を使用すると、プロセス・インスタンス、アクティビティー・インスタンス、またはインライン・ヒューマン・タスクの履歴をモニターできます。Business Process Choreographer Explorer の構成にレポート機能が含まれる場合は、独自のレポートを定義するか、ドリルダウン機能を使用することにより、特定のプロセス・インスタンス、アクティビティー・インスタンス、またはインライン・ヒューマン・タスクに関する詳細な情報を取得できます。また、さらに外部処理を行うために、レポート結果をエクスポートできます。
- 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。例えば、ビジネス・プロセス、サービス、およびヒューマン・タスクを開始したり、ヒューマン・タスクの作業、編集、保存、完了、または解放を行ったりすることができます。さらに、不在のフラグを立てて、代理人を定義できます。

それに加えて、Business Process Choreographer Explorer は、ビジネス・プロセスとそれに関連したアクティビティー、および注意が必要なヒューマン・タスクをディスカバーするために使用できる検索機能も提供します。例えば、それらのインスタンスの状況を検査したり、関連したインスタンスとテンプレートとの間をナビゲートしたり、また関連したアクティビティーおよびヒューマン・タスクを含むプロセ

ス状態のグラフィカル・ビューを取得することができます。

関連概念

『Business Process Choreographer Explorer ユーザー・インターフェース』
Business Process Choreographer Explorer は、ビジネス・プロセスおよびヒューマン・タスクを管理し、プロセスおよびアクティビティー・イベントに関するレポートを作成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。

Business Process Choreographer Explorer ユーザー・インターフェース

Business Process Choreographer Explorer は、ビジネス・プロセスおよびヒューマン・タスクを管理し、プロセスおよびアクティビティー・イベントに関するレポートを作成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・ヘルプにアクセスしたりすることができます。また、「**代理人**」オプションおよび「**代理人の定義**」オプションで、不在設定を指定することもできます。これらのオプションは、 Business Process Choreographer で Human Task Manager に対して代替が有効になっていて、WebSphere Application Server セキュリティーで Virtual Member Manager サービスが構成されている場合に選択できます。

代理人 このオプションは、ユーザーのタスクに対して代理人を指定するために選択します。

代理人の定義
このオプションは、ユーザーの不在設定を定義するために選択します。

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、ナビゲーション・ペインからビューを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

「ビュー」タブを選択すると、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、あるいは、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表示されます。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・ペインに対して事前定義ビューの追加および除去を行い、カスタム・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により、ナビゲーション・ペインのコンテンツ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。

「レポート」タブを選択すると、ナビゲーション・ペインには、作成するレポートの種類 (例えば、アクティビティー・インスタンスのデータを図表に表示できるレポート) を選択するために使用するリンクが表示されます。ランタイム・エンティティーの状態やイベント情報 (例えば、プロセスやアクティビティーのスナップショット・グラフ) を取得するには、事前に定義したリストや図表を使用します。「レポート」タブは、レポート作成が構成されている場合にのみ表示されます。レポート作成機能は、Business Process Choreographer Explorer の構成時に構成できますが、後で構成することもできます。

ページ・タイトル

「ビュー」タブを選択すると、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表示されます。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。ページについて詳しくは、それぞれのページの「**ヘルプ**」アイコンをクリックしてください。

「レポート」タブを選択すると、ワークスペースには、事前定義のリストや図表を表示したり、レポート定義を指定したり、レポートを表示したりするためのページが表示されます。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。ページについて詳しくは、それぞれのページの「**ヘルプ**」アイコンをクリックしてください。

Business Process Choreographer Explorer の「ビュー」タブ

Business Process Choreographer Explorer の「ビュー」タブを使用すると、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・インスタンスや作業割り当てなど) の管理に使用するビューにアクセスできます。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、独自の個別設定ビューを定義して、ナビゲーション・ペインに追加することもできます。さらに、システム管理者であれば、すべてのユーザーに使用可能なカスタマイズ・ビューを定義できます。

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。

- 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矢印をクリックして、その項目を拡大または縮小表示します。
- 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。
- 新規検索を定義します。

  オブジェクトの検索や個別設定ビューの定義を行うには、「**新規検索**」アイコン ( ) をクリックします。

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。「**ポップアップ・メニューの表示**」アイコン ( ) は、ポップアップ・メニューが使用可能であることを示します。

- 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、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。
- リスト内でのビューの位置を上下に移動するには、「**上へ**」アイコン（  ）または 「**下へ**」アイコン（  ）をクリックします。

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・ペイン内の事前定義ビュー

これらのビュー・グループはナビゲーション・ペインで使用できますが、最初はポップアップ・メニューに表示されません。「**カスタマイズ**」を使用してナビゲーション・ペインが変更されると、これらの事前設定ビューの前に「**事前定義ビュー**」アイコン  が表示され、ビューを上下に移動できます。

システム管理者によりナビゲーション・ペインに追加されたカスタマイズ・ビューおよび事前定義ビュー

ビジネス・ユーザーはビュー名をクリックして、ビューにナビゲートできます。システム管理者の場合、ポップアップ・メニューが使用可能です。

- これらの事前定義ビューは、「**事前定義ビュー**」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**」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**」アイコン  で示されます。これらのビューは、そのビューを作成したユーザーにのみ表示されます。ユーザーはビューを削除、編集、コピー、および移動することができます。

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。すべてのビューでは、追加フィルターとは関係なく、許可された項目が表示されます。例えば、強制終了したプロセスのうち、表示が許可されたものだけが表示されます。ビューのグループにビューが定義されていない場合、そのグループは表示されません。

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。

現在有効

このビューには、各プロセスの現在有効なバージョンのプロセス・テンプレートのリストが表示されます。すなわち、有効開始日がすでに過ぎているプロセスの、最新の開始済みバージョンです。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。

すべてのバージョン

このビューには、すべてのプロセス・バージョンのプロセス・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・バージョンのプロセス・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・グループには、以下のビューがあります。

停止したアクティビティー
このビューには、停止状態にあるアクティビティーが表示されます。

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。

ユーザーの予定
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。タスクの優先順位の変更と、タスクのビジネス・カテゴリーの変更もできます。

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。タスクの優先順位の変更と、タスクのビジネス・カテゴリーの変更もできます。

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。タスクの優先順位の変更と、タスクのビジネス・カテゴリーの変更もできます。

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。タスクの優先順位の変更と、タスクのビジネス・カテゴリーの変更もできます。

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。

Business Process Choreographer Explorer の「レポート」タブ

Business Process Choreographer Explorer の「レポート」タブを使用して、Business Process Choreographer で処理された特定のプロセスおよびアクティビティーについてのレポートを管理します。作成するレポートの種類 (プロセス・レポートやアクティビティー・レポートなど) を選択することができます。また、独自のレポート定義を保管して、ナビゲーション・ペインに追加することもできます。ドリルダウン方式の事前定義のリストおよび図表は、ランタイム・エンティティーの状態およびイベント情報を入手するために使用します。例えば、リスト、プロセスおよびアクティビティーのスナップショット・グラフ、期間グラフによるプロセス・インスタンスおよびアクティビティー・インスタンスが選択できます。「レポート」タブは、レポート作成が構成されている場合にのみ表示されます。レポート作成機能は、Business Process Choreographer Explorer の構成時に構成できますが、後で構成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。

- 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矢印をクリックして、その項目を拡大または縮小表示します。
- 事前定義のリストまたはグラフにナビゲートします。

  レポートするインスタンスの種類をクリックします。
- プロセス・レポートまたはアクティビティー・レポートのウィザードにナビゲートします。

  レポートのタイプ、レポートの内容、レポートのフィルター基準を指定するには、「**新規レポート**」アイコン () をクリックします。
- 保管されたプロセス・レポートまたはアクティビティー・レポートを実行します。

  レポート名をクリックして、レポートを実行します。
- 保管されたプロセスまたはアクティビティー・レポート定義のポップアップ・メニューを開きます。

  保存済みのレポート定義を操作するには、「**ポップアップ・メニューの表示**」アイコン () をクリックします。

  - レポート定義を削除するには、「**削除**」アイコン () をクリックします。
  - レポート定義を編集するには、「**編集**」アイコン () をクリックします。
  - レポート定義をコピーするには、「**コピー**」アイコン () をクリックします。

– レポート結果をエクスポートするには、「**エクスポート**」アイコン（  ）をクリックします。
– レポートを非同期で実行するには、「**非同期レポート (Asynchronous Report)**」アイコン（  ）をクリックします。
  - 非同期レポートが正常に完了すると、ナビゲーション・ペインに「**非同期レポート完了 (Asynchronous Report Completed)**」アイコン（  ）が表示されます。レポートの名前をクリックして、結果を表示します。
  - 非同期レポートが正常に完了しなかった場合は、「**非同期レポート失敗 (Asynchronous Report Failed)**」アイコン（  ）が表示されます。

ナビゲーション・ペインの事前定義リストおよびグラフ

ナビゲーション・ペインには、以下の事前定義リストおよびグラフのグループが含まれています。

リスト このグループには以下のリストが含まれています。

プロセス

このリストは、指定された時間フレーム中にプロセス・イベントを発行してプロセスを表示するために使用します。プロセスはプロセス状態に従ってリストされます。

アクティビティー

このリストは、選択したアクティビティーが指定された時間フレーム中に到達した状態を表示するために使用します。アクティビティーはアクティビティー状態に従ってリストされます。

ユーザー

このリストは、選択したユーザーが指定された時間フレーム中に実行したアクティビティー、およびアクティビティーが到達した状態を表示するために使用します。アクティビティーはその状態に従って表示されます。各アクティビティーに対応するユーザーが表示されます。

グラフ このグループには以下のグラフが含まれています。

プロセス・スナップショット

このグラフは、指定の時間にそれぞれの状態にあったプロセス・インスタンスの数を調べるために使用します。データは棒グラフまたは円グラフで表示できます。

期間によるプロセス

このグラフは、指定の期間中に指定の状態になったプロセス・インスタンスの数の分布を調べるために使用します。各インスタンスが、指定された状態になったタイム・スライスに表示されます。データは線グラフ、棒グラフ、または円グラフで表示できます。

アクティビティー・スナップショット

このグラフは、指定の時間にそれぞれの状態にあったアクティビティー・インスタンスの数を調べるために使用します。データは棒グラフまたは円グラフで表示できます。

期間によるアクティビティー

このグラフは、指定の期間中に指定の状態になったアクティビティー・インスタンスの数の分布を調べるために使用します。各インスタンスが、指定された状態になったタイム・スライスに表示されます。データは線グラフ、棒グラフ、または円グラフで表示できます。

プロセス・レポートとアクティビティー・レポート

ナビゲーション・ペインは以下のレポート・ウィザードにリンクします。レポート・ウィザードは「**新規レポート**」アイコン（  ）で示されます。

プロセス・レポート

プロセス・レポートは、プロセス・インスタンス・イベントを照会するために使用します。これらのイベントはプロセス・インスタンスの状態変更について記述します。レポートのデータを定義するには、レポート・ウィザードを使用します。レポート定義を保存したり取り出したりすることができます。

アクティビティー・レポート

アクティビティー・レポートを使用して、アクティビティー・インスタンス・イベントを照会します。これらのイベントはアクティビティー・インスタンスの状態変更について記述します。個々のレポートを指定するには、レポート・ウィザードを使用します。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。

Business Process Choreographer Explorer の開始

Business Process Choreographer Explorer は Web アプリケーションであり、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できます。 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。レポート作成機能を使用するには、Event Collector アプリケーションがインストールされ、動作している必要があります。

このタスクについて

Business Process Choreographer Explorer を開始するには、次のステップを実行します。

手順

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。また、URL を拡張して、プロセス、タスク、またはエスカレーションの詳細に直接移動することもできます。

```
http://app_server_host:port_no/context_root?oid_type=oid
```

以下に例を示します。

```
http://hostname:9080/bpc?piid=PI:90030109.7232ed16.d33c67f6.beb30076
```

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。

oid_type
オプション。表示するオブジェクトのタイプ。このパラメーターは、以下の値のいずれかをとることができます。

aiid アクティビティー・インスタンス ID

piid プロセス・インスタンス ID

ptid プロセス・テンプレート ID

tkiid タスク・インスタンス ID

tktid タスク・テンプレート ID

esiid エスカレーション・インスタンス ID

oid
オプション。オブジェクト ID の値。

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。

タスクの結果

オブジェクト ID を指定した場合は、そのオブジェクトの詳細ページが表示されます。オブジェクト ID を指定しなかった場合は、Business Process Choreographer Explorer の初期ページが表示されます。デフォルトでは、初期ページは「ユーザーの予定」ビューです。

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、特定の Business Process Choreographer Explorer インスタンス用のインターフェースを、このインスタンスに割り当てられるユーザー・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。 さらに、構成時に (または後で)、ユーザーは、レポート作成機能を追加して、プロセスとアクティビティーに関するレポートを作成したり、イベントに基づく統計情報を取得したりすることもできます。

このタスクについて

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 Choreographer 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。「ユーザーの予定」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。いずれかのシステム管理者ロールに割り当てられている場合は、ユーザーに表示される事前定義ビューと、ユーザーのログイン後に表示されるデフォルト・ビューの両方をカスタマイズできます。

このタスクについて

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・ステート・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・ステート・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、カスタマー・オーダーを処理するユーザーに、カスタマー・サービスの照会を処理するユーザーとは異なるインターフェースを提供する必要がある場合があります。Business Process Choreographer Explorer のインスタンスは、そのインスタンスに割り当てられているユーザーのワークフロー・パターンに合うようにカスタマイズできます。

一連のビューとデフォルトのログイン・ビューをカスタマイズし、Business Process Choreographer Explorer の新規ビューを定義するには、次のステップを実行します。

手順

1. ナビゲーション・ペイン内の一連のビューと、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、「カスタマイズ・ビューの検索および定義」ページ (ステップ 2 を参照) 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。

変更を保管すると、各事前定義ビューがナビゲーション・ペイン内に表示され、ビューの前にはアイコンが付きます。これらのアイコンを使用すれば、事前定義ビューをリスト内で上下に移動させることができます。

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. 新規ビューを定義します。

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. 「カスタマイズ・ビューの検索および定義」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. 「XXX の検索およびカスタマイズ・ビューの定義」ページ (XXX はビューのタイプ。例えば「プロセス・テンプレート」) で、ビューの照会テーブルを選択します。

ビュー定義のデフォルト照会テーブルが設定されます。別の照会テーブルを選択することもできます。または、ビュー定義で照会テーブルを使用しないようにすることもできます。

注: 照会テーブルを使用する場合、ここでビューに追加の検索基準を指定することはできません。すべての検索基準は、照会テーブル定義で定義する必要があります。

照会テーブルを使用しない場合は、検索基準を指定します。「プロセスの基準」タブ、「タスクの基準」タブ、および「プロパティー・フィルター」タブを使用して、検索結果を限定します (例えば、特定のプロセス・テンプレートに限定)。また、インスタンス・ビューの定義時に、「ユーザー役割」タブを使用して、検索結果をユーザー、グループ、または役割に限定することもできます。

d. 照会テーブルを使用していて、照会テーブル定義にパラメーターが含まれている場合は、必要な照会パラメーターを「照会プロパティー」タブで指定します。

指定するパラメーター名は、照会テーブル定義での名前と一致しなければなりません。パラメーターのデフォルト値を指定し、ビュー照会の実行時にデフォルト値を上書きできるかどうかを指定することもできます。

e. 「プロパティーの表示」タブを使用して、ビューに組み込む順序付けプロパティーや結果のしきい値などのリスト列とリスト・プロパティーを選択します。

また、「ビューの設定」で、ビューのアクション・バーに追加するアクションを指定できます。ビューに組み込むアクション、または実行しようとしている検索を選択するには、以下を実行します。

- 「使用可能なアクション」で、1 つまたは複数のアクションを選択して「**追加**」をクリックします。
- アクションを削除するには、「ビューのアクション」内でアクションを選択して、「**削除**」をクリックします。
- アクション・バー内のアクションの順序は、アクションを「ビューのアクション」内で上下に移動させて指定できます。

これが、タスク、プロセス、またはアクティビティー・インスタンス・ビューの場合は、「**ビューの設定**」をクリックして、システム管理者およびシステム・モニター用のビューに組み込まれる項目を指定します。

- システム管理者およびシステム・モニターであれば、検索結果を自分自身のインスタンスに制限できます。
  - ビュー内の検索基準に一致するすべての項目を表示するには、「**すべてのインスタンス**」を選択します。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どうかに関係なく、すべての項目が表示されます。
  - ログオン・ユーザーが作業項目を持っている項目のみを表示するには、「**個人用インスタンス**」を選択します。

f. 「**ビュー名**」フィールドにビューの表示名を入力して、「**検査して保存**」をクリックします。

エラーがないか確認するために検索が実行されます。エラーが発生せずに実行された場合は、ビューが保存されます。

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。ビューは、ナビゲーション・ペイン内で上下に移動させることができます。

ビジネス・ステート・マシンのプロセス・テンプレート用ビューの定義

ビジネス・ステート・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。

始める前に

カスタマイズ・ビューを作成するには、システム管理者のいずれかのロールが必要です。

このタスクについて

Business Process Choreographer Explorer で次のステップを実行します。

手順

1. タスクバーの「**ビューの定義**」をクリックします。
2. 「カスタマイズ・ビューの検索および定義」ページで、「**プロセス・テンプレートの検索およびカスタマイズ・ビューの定義**」を選択します。
3. 「**プロパティー・フィルター**」 → 「**カスタム・プロパティー・フィルター**」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` 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。
   b. 「**追加**」をクリックします。
   c. 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. 「**プロパティーの表示**」 → 「**リスト列**」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。
      - 「**表示名**」フィールドに、列の表示名を入力し、「**追加**」をクリックします。
   b. 選択した列のリストに他の列を追加するか、リストから列を除去します。
5. 「**ビュー名**」フィールドに照会の表示名を入力し、「**検査して保存**」をクリックします。

   エラーがないか確認するために検索が実行されます。エラーが発生せずに実行された場合は、ビューが保存されます。

タスクの結果

デフォルトでは、新規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・ステート・マシンのプロセス・インスタンス用ビューの定義

ビジネス・ステート・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。

始める前に

カスタマイズ・ビューを作成するには、システム管理者のいずれかのロールが必要です。

このタスクについて

Business Process Choreographer Explorer で次のステップを実行します。

手順

1. タスクバーの「**ビューの定義**」をクリックします。

2. 「カスタマイズ・ビューの検索および定義」ページで、「**プロセス・インスタンスの検索およびカスタマイズ・ビューの定義**」を選択します。
3. 「**カスタム・プロパティー・フィルター**」 → 「**カスタム・プロパティー・フィルター**」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` 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。
   b. 「**追加**」をクリックします。
   c. 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. 「**プロパティーの表示**」 → 「**リスト列**」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`、「**名前空間**」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・ステート・マシンのターゲット名前空間です。また、「**表示名**」フィールドに列の表示名を指定し、「**追加**」をクリックします。
      - 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、「**名前空間**」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・ステート・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・ステート・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*

        名前空間
        照会プロパティーの名前空間。ここで、`tns` は、*-process* のサフィックスが付けられたビジネス・ステート・マシンのターゲット名前空間です。
   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. 「**ビュー名**」フィールドに照会の名前を入力し、「**検査して保存**」をクリックします。

   エラーがないか確認するために検索が実行されます。エラーが発生せずに実行された場合は、ビューが保存されます。

タスクの結果

デフォルトでは、新規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、ユーザーのロールに関係なく、独自のビューを追加することができます。例えば、特定のタスクまたはプロセスの進行状況をモニターするために、新規のビューを追加することができます。ビューに表示される情報と、ビューで提供されるフィルター、ソート基準、およびアクションを指定できます。

このタスクについて

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。

手順

1. 新しいビューを定義する「ビュー」タブのナビゲーション・ペインのセクション (例:「プロセス・テンプレート」) で、「**新規検索**」アイコン () をクリックします。
2. 「XXX の検索およびカスタマイズ・ビューの定義」ページ (XXX はビューのタイプ。例えば「プロセス・テンプレート」) で、ビューの照会テーブルを選択します。

   ビュー定義のデフォルト照会テーブルが設定されます。別の照会テーブルを選択することもできます。または、ビュー定義で照会テーブルを使用しないようにすることもできます。

   注: 照会テーブルを使用する場合、ここでビューに追加の検索基準を指定することはできません。すべての検索基準は、照会テーブル定義で定義する必要があります。

   照会テーブルを使用しない場合は、検索基準を指定します。「プロセスの基準」タブ、「タスクの基準」タブ、および「プロパティー・フィルター」タブを使用して、検索結果を限定します (例えば、特定のプロセス・テンプレートに限定)。また、インスタンス・ビューの定義時に、「ユーザー役割」タブを使用して、検索結果をユーザー、グループ、または役割に限定することもできます。
3. 照会テーブルを使用していて、照会テーブル定義にパラメーターが含まれている場合は、必要な照会パラメーターを「照会プロパティー」タブで指定します。

   指定するパラメーター名は、照会テーブル定義での名前と一致しなければなりません。パラメーターのデフォルト値を指定し、ビュー照会の実行時にデフォルト値を上書きできるかどうかを指定することもできます。

4. 「プロパティーの表示」タブを使用して、ビューに組み込む順序付けプロパティーや結果のしきい値などのリスト列とリスト・プロパティーを選択します。

   また、「ビューの設定」で、ビューのアクション・バーに追加するアクションを指定できます。ビューに組み込むアクション、または実行しようとしている検索を選択するには、以下を実行します。

   - 「使用可能なアクション」で、1 つまたは複数のアクションを選択して「**追加**」をクリックします。
   - アクションを削除するには、「ビューのアクション」内でアクションを選択して、「**削除**」をクリックします。
   - アクション・バー内のアクションの順序は、アクションを「ビューのアクション」内で上下に移動させて指定できます。

   これが、タスク、プロセス、またはアクティビティー・インスタンス・ビューの場合は、「**ビューの設定**」をクリックして、システム管理者およびシステム・モニター用のビューに組み込まれる項目を指定します。システム管理者またはシステム・モニターであれば、検索結果を自分自身のインスタンスに限定できます。

   - ビュー内の検索基準に一致するすべての項目を表示するには、「**すべてのインスタンス**」を選択します。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どうかに関係なく、すべての項目が表示されます。
   - ログオン・ユーザーが作業項目を持っている項目のみを表示するには、「**個人用インスタンス**」を選択します。

5. 「**ビュー名**」フィールドにビューの表示名を入力して、「**検査して保存**」をクリックします。

   エラーがないか確認するために検索が実行されます。エラーが発生せずに実行された場合は、ビューが保存されます。「要約」タブを使用して、現在ビューに設定されている内容を確認します。

タスクの結果

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) コンポーネントに基づいた、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。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。スタイル・シートを変更して、新規コードをいっさい書き込まずに、特定の外観に合うようにユーザー・インターフェースを調整できます。

始める前に

スタイル・シートを変更するには、カスケーディング・スタイル・シートについて熟知している必要があります。

このタスクについて

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。

手順

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。

関連概念

358 ページの『Business Process Choreographer Explorer ユーザー・インターフェース』

Business Process Choreographer Explorer は、ビジネス・プロセスおよびヒューマン・タスクを管理し、プロセスおよびアクティビティー・イベントに関するレポートを作成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。

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。

- 『バナー』
- 376 ページの『フッター』
- 376 ページの『メニュー・バー』
- 376 ページの『ログイン・ページ』
- 376 ページの『ナビゲーター』
- 377 ページの『コンテンツ・パネル』
- 377 ページの『コマンド・バー』
- 377 ページの『リスト』
- 377 ページの『詳細パネル』
- 378 ページの『メッセージ・データ』
- 378 ページの『タブ付きペイン』
- 378 ページの『検索ページ』
- 379 ページの『エラー詳細』

このファイルは、次のディレクトリーにあります。

```
<profile_root>¥installedApps¥<node_name>¥<explorer_instance>¥bpcexplorer.war¥theme
```

バナー

スタイル名	説明
.banner	バナーの境界。
.banner_left	バナー内の境界。アプリケーションのタイトル・イメージを組み込むときに使用されます。
.banner_right	バナー内の境界。例えば、その他のロゴを表示する場合などに使用できます。

フッター

スタイル名	説明
.footer	フッターの境界。
.footer_left	フッター内の境界。例えば、アプリケーションの企業ロゴを表示する場合などに使用できます。
.footer_right	フッター内の境界。例えば、その他のロゴを表示する場合などに使用できます。

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。

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。

ナビゲーター

スタイル名	説明
.pageBodyNavigator	ナビゲーターを組み込む領域。
.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。

スタイル名	説明
.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。
.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。
.navigatorLink	ナビゲーター・ボックス内のリンク。
.expanded	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
.pageBodyContent	コンテンツを組み込む領域。
.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。
.panelTitle	表示コンテンツの表題。例えば、「ユーザーの予定」。
.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
.commandbar	コマンド・バー領域の周囲の分割コンテナー。
.button	コマンド・バー内のボタンに使用されるスタイル。

リスト

スタイル名	説明
.list	複数の行を含む表。
.listHeader	リストのヘッダー行に使用されているスタイル。
.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。

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。
.detailsProperty	プロパティー名のラベル。

スタイル名	説明
.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。
.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。
.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。
.selectedTab	アクティブなタブのヘッダー。
.tab	非アクティブなタブのヘッダー。
.tabPane	タブ付きペインを囲む分割コンテナー。
.tabbedPaneNested	検索ページで使用されるネストしたタブ付きペインを囲む分割コンテナー。
.tabHeaderSimple	ネストしたタブ付きペインのタブ・ヘッダー。
tabHeaderProcess	プロセス・フィルターのネストしたタブ付きペインのタブ・ヘッダー。
.tabHeaderTask	タスク・フィルターのネストしたタブ付きペインのタブ・ヘッダー。
.tabPaneSimple	ネストしたタブ付きペインを囲む分割コンテナー。

検索ページ

スタイル名	説明
.searchPane	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。
.summary	検索要約ペインを囲む分割コンテナー。
.summaryTitle	検索要約ペインのすべてのタイトルに適用される共通スタイル。
.summaryTitleProcess	検索要約ペインのプロセス関連セクションのタイトルに適用されるスタイル。
.summaryTitleTask	検索要約ペインのタスク関連セクションのタイトルに適用されるスタイル。

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	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。
.errorDetailsMessage	エラー・メッセージのテキスト・スタイル。

第 8 章 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・テンプレートとタスク・テンプレートを管理します。 Business Process Choreographer Explorer を使用して、プロセス・インスタンスとタスク・インスタンスを処理し、またビジネス・プロセスおよびヒューマン・タスクについて報告します。

プロセス管理をシステム管理者に制限する

デフォルトのインスタンス・ベース管理をオフにすると、プロセス・インスタンス、スコープ・インスタンス、およびアクティビティー・インスタンスを管理できるのは BPESystemAdministrator ロールのユーザーのみになります。さらに、プロセス・インスタンスを表示またはモニターできるのは、BPESystemMonitor ロールのユーザーのみになります。これにより、データベースの負荷および成長率が低下するので、パフォーマンスを向上させることができます。

始める前に

管理とモニターを BPESystemAdministrator ロールおよび BPESystemMonitor ロールに制限する前に、これらのロールが `BPEContainer` アプリケーションの適切なユーザー・セットにマップされていることを確認してください。

このタスクについて

パフォーマンスを向上させるために、モデル化されたタスクを含むすべての管理タスクを使用不可にするカスタム・プロパティーを設定できます。適切な管理ロールおよびモニター・ロールのメンバーのみがプロセス、アクティビティー、およびスコープを管理できます。

注: この設定を変更すると、新規に作成されたプロセス・インスタンス、アクティビティー・インスタンス、およびスコープ・インスタンスに対して管理アクションを実行できるユーザー ID に影響を与えます。これにより、管理アクションを実行する担当者または自動化プロセスが、適切なロールではないユーザー ID を使用したために、管理アクションの実行を拒否される可能性があります。

手順

1. Business Process Choreographer がクラスターに構成されているかサーバーに構成されているかに応じて、管理コンソールを使用して次のいずれかを実行します。

   クラスター
   「サーバー」 → 「クラスター」 → 「WebSphere Application Server クラスター」 → 「*cluster_name*」をクリックします。

   Server
   「サーバー」 → 「サーバー・タイプ」 → 「WebSphere Application Server」 → 「*server_name*」をクリックします。

2. 「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Business Flow Manager**」 → 「**カスタム・プロパティー**」をクリックします。
3. 「**新規**」をクリックして、新規カスタム・プロパティーを追加します。
4. 名前 `ProcessAdministration` と値 `useSystemAdminAuthorizationOnly` を入力します。

   注: このカスタム・プロパティーを削除すると、許可制限は取り消されますが、これは新規プロセス・インスタンス、アクティビティー、およびスコープのみが対象となります。
5. 変更を保存します。
6. Business Process Choreographer が構成されているクラスターのサーバーを再始動することにより、変更をアクティブにします。

タスクの結果

プロセスとアクティビティーの管理およびモニターは、BPESystemAdministrator ロールおよび BPESystemMonitor ロールのユーザーに制限されます。この変更は、新規インスタンスのみに影響を与えます。この変更が行われる前に作成された既存のインスタンスは、引き続き、作成時にアクティブであった管理設定に従います。

関連概念

代替プロセス管理許可モード
管理タスクを組み込むようにプロセス・テンプレートをモデル化した場合は、この管理モードによって、それらの管理タスクおよび関連アクション (エスカレーションやモニターなど) が非アクティブ化されます。このモードで実行すると、インスタンス・ベースの管理が使用不可になると共に、プロセス管理およびモニターがシステム管理者またはシステム・モニターであるユーザーに制限されるので、パフォーマンスが向上します。

関連情報

ビジネス・プロセスのための Java EE ロール
Business Process Choreographer の構成時に、Java EE ロールが設定されます。Java EE ロール・ベースの許可を設定するには、ユーザー・レジストリーが構成されており、アプリケーション・セキュリティーが有効になっている必要があります。

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。

このタスクについて

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。』
- 『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。』
- 384 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。』
- 384 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。』
- 384 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働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。

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。

始める前に

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。

- WebSphere 管理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。
- プロセス・テンプレートが停止または開始されるアプリケーション・サーバーは、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。

このタスクについて

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。

手順

1. 管理するモジュールを選択します。

   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」** → **「SCA モジュール」** → 「***module_name***」をクリックします。
2. **「追加プロパティー」**の下の SCA 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。
3. 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。

4. 停止状態のプロセス・テンプレートを開始します。

管理スクリプトによるプロセス・テンプレートの停止および開始

管理スクリプトは、プロセスおよびタスク・テンプレートを停止および開始するために、管理コンソールの代わりに使用できます。管理スクリプトを使用して、アプリケーションに属するすべてのテンプレートを停止および開始します。

始める前に

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。

- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にオペレーター権限も管理者権限もない場合は、wsadmin `-user` および `-password` のオプションを付けて、オペレーターまたは管理者の権限を持つユーザー ID を指定します。
- テンプレートが停止または開始されるアプリケーション・サーバーは、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。

このタスクについて

以下のステップでは、管理スクリプトを使用して、アプリケーションに属するプロセスおよびタスク・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。

手順

1. 管理スクリプトがある 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. プロセスおよびタスク・テンプレートを停止するには、次のようにします。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin -f bpcTemplates.jacl
                            -stop application_name
   ```

   各部の意味は、次のとおりです。

   -stop *application_name*
   指定されたアプリケーションに属するすべてのテンプレートが停止します。

   プロセスおよびタスク・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。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新規インスタンスを作成することはできません。

3. プロセスおよびタスク・テンプレートを開始するには、次のようにします。

   ```
   install_root/bin/wsadmin -f bpcTemplates.jacl
                            -start application_name
   ```

アプリケーションに属するテンプレートが開始します。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。

このタスクについて

最新の有効開始日付を持つ、インストールおよび開始されたプロセス・テンプレートはすべて、Business Process Choreographer Explorer のプロセス・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。

手順

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。

   「ビュー」タブのナビゲーション・ペインの「プロセス・テンプレート」の下で、**「現在有効」**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。

タスクの結果

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、人との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、潜在的な所有者が要求できる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「ユーザーの予定」ページのリストに表示されます。

プロセス・インスタンスが Microflow である場合は、Web ブラウザーにプロセス出力メッセージが自動的に表示されます。プロセスの完了後に自動的に削除されず、長期間にわたって実行するプロセスの場合、プロセス出力メッセージはプロセス・インスタンス・ビューで見ることができます。出力メッセージを表示するには、Business Process Choreographer Explorer でプロセス・リストからインスタンスを選択し、プロセス・インスタンス・ビューを開きます。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実装している場合、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。

このタスクについて

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。

手順

1. プロセス・インスタンスのリストを表示します。

   例えば、「ビュー」タブのナビゲーション・ペインの「プロセス・インスタンス」の下で、「**ユーザーが管理**」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。

   「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。アクティビティーは、状態に応じて、図の中で色分けされています。すべての状態に関連したアイコンがあります。例えば、完了したアクティビティーはチェック・マークで示されます。詳しくは、そのページのオンライン・ヘルプを参照してください。
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て、「**アクティビティーの詳細表示**」を選択します。

   プロセス状態ビューでアクティビティーをクリックすると、コンテキスト・メニューが開きます。このメニューでは、アクティビティーの詳細を表示したり、アクティビティーをスキップ (アクティビティーにスキップのマークを付ける) したり、アクティビティーをプロセス内の別のアクティビティーにジャンプさせるためのソースとして選択したりできます。case 条件の評価で問題があったために失敗した switch アクティビティーを修復することも可能です。

   使用可能なアクションが表示されます。必要なアクションを選択します。

アクティビティーの変数の表示と変更

プロセス・インスタンスのアクティビティー変数を Business Process Choreographer Explorer を使用して表示および変更します。

始める前に

アクティビティーのすべての変数を表示するには、少なくともスコープ・リーダーまたはプロセス・リーダーの権限を持つ必要があります。変数を変更するには、スコープ管理者またはプロセス管理者の権限を持つ必要があります。

このタスクについて

アクティビティーに可視のすべての変数にアクセスして、変数の値を変更することができます。

Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「ビュー」タブで、「プロセス・インスタンス」ページに移動します。以下のいずれかを行います。
   - 「**プロセス状態の表示**」をクリックします。次に、プロセス状態遷移で対象のアクティビティーをクリックして、「**アクティビティー変数の表示**」をクリックします。選択したアクティビティーに可視の各変数が表示されます。リストを使用して、このプロセス・インスタンス内の別のアクティビティーを選択して可視変数を表示します。
   - 「**アクティビティー変数**」をクリックします。リストを使用して、このプロセス・インスタンス内のアクティビティーを選択し、可視変数を表示します。
   - 「**アクティビティーのスキップ**」をクリックします。アクティビティーを 1 つ選択してから、「**変数の設定**」をクリックします。選択したアクティビティーに可視の各変数が表示されます。リストを使用して、このプロセス・インスタンス内の別のアクティビティーを選択し、可視変数を表示します。
2. 変数名を選択すると、実際の値が表示されます。
3. 値を変更して「**保管**」をクリックすると、変数の設定値が更新されます。

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。

始める前に

プロセス・インスタンスを中断および再開するには、プロセス管理者またはシステム管理者でなければなりません。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールのユーザーのみがこのアクションを実行できます。

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。

このタスクについて

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。

手順

1. プロセス・インスタンスのリストを表示します。

   例えば、「ビュー」タブのナビゲーション・ペインの「プロセス・インスタンス」の下で、「**ユーザーが管理**」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。
3. 以下のいずれかのオプションを選択して、プロセス・インスタンスを中断します。
   - プロセスを手動で再開するまで中断するには、「**中断**」を選択します。
   - 特定の時刻までプロセスを中断するには、「**次の期間までプロセスを中断**」を選択して日時を指定します。
   - 一定期間プロセスを中断するには、「**プロセスの中断期間**」を選択して期間を指定します。
4. 選択を確認するには、「**実行依頼**」をクリックします。

   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。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。

次のタスク

中断状態のプロセス・インスタンスを再開するには、プロセス・インスタンスを選択して「**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。

関連概念

代替プロセス管理許可モード
管理タスクを組み込むようにプロセス・テンプレートをモデル化した場合は、この管理モードによって、それらの管理タスクおよび関連アクション (エスカレーションやモニターなど) が非アクティブ化されます。このモードで実行すると、インスタンス・ベースの管理が使用不可になると共に、プロセス管理およびモニターがシステム管理者またはシステム・モニターであるユーザーに制限されるので、パフォーマンスが向上します。

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。

始める前に

プロセス・インスタンスを終了するには、プロセス管理者またはシステム管理者でなければなりません。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールのユーザーのみがこのアクションを実行できます。

このタスクについて

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。

手順

1. 管理できるプロセス・インスタンスを表示します。

   「ビュー」タブのナビゲーション・ペインの「プロセス・インスタンス」の下で、「**ユーザーが管理**」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。

関連概念

代替プロセス管理許可モード
管理タスクを組み込むようにプロセス・テンプレートをモデル化した場合は、この管理モードによって、それらの管理タスクおよび関連アクション (エスカレーションやモニターなど) が非アクティブ化されます。このモードで実行すると、インスタンス・ベースの管理が使用不可になると共に、プロセス管理およびモニターがシステム管理者またはシステム・モニターであるユーザーに制限されるので、パフォーマンスが向上します。

Business Process Choreographer Explorer によるプロセス・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。

始める前に

プロセス・インスタンスを削除するには、プロセス管理者またはシステム管理者でなければなりません。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールのユーザーのみがこのアクションを実行できます。

プロセス・インスタンスは、終了、失敗、強制終了、または補正のいずれかの状態になっている必要があります。

このタスクについて

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。

手順

1. 管理するプロセス・インスタンスを表示します。

「ビュー」タブのナビゲーション・ペインの「プロセス・インスタンス」の下で、「**ユーザーが管理**」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。

タスクの結果

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。

関連概念

代替プロセス管理許可モード
管理タスクを組み込むようにプロセス・テンプレートをモデル化した場合は、この管理モードによって、それらの管理タスクおよび関連アクション (エスカレーションやモニターなど) が非アクティブ化されます。このモードで実行すると、インスタンス・ベースの管理が使用不可になると共に、プロセス管理およびモニターがシステム管理者またはシステム・モニターであるユーザーに制限されるので、パフォーマンスが向上します。

Business Process Choreographer Explorer によるプロセス・インスタンスのマイグレーション

プロセス・インスタンスを、現在有効なバージョンのプロセスにマイグレーションできます。このタスクは、例えば、より新しいバージョンのプロセス・インスタンスが有効になった場合に実行できます。

始める前に

プロセス・インスタンスをマイグレーションするには、プロセス管理者またはシステム管理者でなければなりません。

このタスクについて

プロセスの新規バージョンをデプロイする場合、新規プロセス・インスタンスはこのバージョンのプロセスに基づいています。ただし、前のプロセス・テンプレートに基づく既存のプロセス・インスタンスは、それらが終了状態になるまで稼働し続けます。既存のプロセス・インスタンスを、現在有効なバージョンのプロセスにマイグレーションできます。現在有効なプロセス・テンプレートとは、開始された最も新しいバージョンのプロセスであり、その有効開始日は未来の日付ではありません。

migrateProcessInstances.py スクリプトを使用して、プロセス・インスタンスのマイグレーションおよびそれらの一括マイグレーションを行うこともできます。詳しくは、関連情報のセクションを参照してください。

プロセス・インスタンスをマイグレーションするには、Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「ビュー」タブで、「**すべてのバージョン**」をクリックして、インストールされたすべてのプロセス・テンプレートを表示します。

2. プロセス・テンプレートを 1 つ以上選択し、「**インスタンス**」をクリックして、関連付けられたプロセス・インスタンスを表示します。
3. 該当するプロセス・インスタンス・エントリーを選択し、「**マイグレーション**」をクリックして、それらを現在有効なバージョンのプロセスにマイグレーションします。現在有効なバージョンのプロセスに基づくプロセス・インスタンスはマイグレーションできないため、それらを選択しないでください。

タスクの結果

選択したプロセス・インスタンスが新規プロセス・テンプレートにマイグレーションされます。項目に対してエラーが発生した場合は、基本的なエラー情報が表示され、エラー詳細へのリンクを含むエラー標識も示されます。

作業権限の管理

プロセス・インスタンスの開始後に、そのプロセスに対する作業権限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作業負荷を効率よく分散させる場合など)。

このタスクについて

*作業項目*は、特定の理由でユーザーまたはユーザー・グループに割り当てられるビジネス・エンティティー (プロセス・インスタンスなど) 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、プロセス・インスタンスには、いくつかの作業項目を関連付けることができます。

場合によっては、プロセス・インスタンスの開始後にそれらのインスタンス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。追加のプロセス作業項目の作成や、不要に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。

プロセス作業項目の作成

プロセス・インスタンスの作業項目を使用して、そのプロセス・インスタンスの管理者権限または読者権限を管理します。例えば、プロセス・インスタンスのスターターのユーザー ID が削除されるのでユーザーがプロセス・インスタンスの所有権を要求する場合などには、新しい管理者のプロセス作業項目を作成する必要があります。また、ユーザーが変数の値を確認または変更する必要がある場合にも、作業項目の作成が必要になります。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。

始める前に

プロセス・インスタンスの作業項目を作成するには、プロセス管理者またはシステム管理者でなければなりません。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールのユーザーのみがこのアクションを実行できます。プロセス・インスタンスは、実行中、失敗中、または強制終了中のいずれかの実行状態でなければなりません。作業項目は、含まれているすべてのアクティビテ

ィー・インスタンスに自動的に伝搬されます。

このタスクについて

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。

手順

1. 管理するプロセス・インスタンスを表示します。

   「ビュー」タブのナビゲーション・ペインの「プロセス・インスタンス」の下で、「**ユーザーが管理**」をクリックします。
2. 作業項目の作成対象となる 1 つ以上のプロセス・インスタンスの横にあるチェック・ボックスを選択し、「**作業項目の作成**」をクリックします。 「プロセス作業項目の作成」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、作業項目の作成対象のユーザー ID を入力します。
   b. 「**理由**」リストから 1 つ以上のロールを選択します。

      理由が `Administrator` の場合、プロセス・インスタンスの管理者権限がユーザーに付与されます。`Reader` の場合は、読者権限が付与されます。
   c. 「**作成**」をクリックします。

タスクの結果

作業項目は、新規作業項目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。

関連タスク

396 ページの『プロセス作業項目の削除』
例えば、エラーのプロセス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プロセス作業項目が生成されている場合は、プロセス作業項目を削除することができます。

『プロセス管理者の場合のプロセス作業項目の転送』
プロセス作業項目の割り当てを変更することが必要になる場合があります。例えば、プロセス・インスタンスの所有権を他のユーザーが要求する必要があるため、そのユーザーにプロセス作業項目を転送しなければならないことがあります。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。

408 ページの『プロセス・インスタンスの所有権の移転』
プロセス・インスタンスの所有権は、プロセス管理者権限を持つユーザーにプロセス・インスタンスの所有権を取得させることによって、移転することができます。この所有権の移転は、例えば、プロセス・スターターが会社を辞めたような状況で行うことがあります。

プロセス管理者の場合のプロセス作業項目の転送

プロセス作業項目の割り当てを変更することが必要になる場合があります。例えば、プロセス・インスタンスの所有権を他のユーザーが要求する必要があるため、

そのユーザーにプロセス作業項目を転送しなければならないことがあります。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。

始める前に

プロセス作業項目を転送する場合、以下の条件が適用されます。

- プロセス管理者またはシステム管理者でなければなりません。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールのユーザーのみがこのアクションを実行できます。
- プロセス・インスタンスは、実行、失敗、または強制終了のいずれかの状態でなければなりません。
- 割り当て理由が `starter` である作業項目を転送することはできません。
- 割り当て理由が `administrator` または `reader` である作業項目を転送すると、その転送は、含まれているすべてのアクティビティー・インスタンスに自動的に伝搬されます。

このタスクについて

プロセス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。

手順

1. 管理できるプロセス・インスタンスを表示します。

   「ビュー」タブのナビゲーション・ペインの「プロセス・インスタンス」の下で、「**ユーザーが管理**」をクリックします。
2. プロセス・インスタンスの作業項目を表示します。

   「ユーザーが管理するプロセス・インスタンス」ページで、1 つ以上のプロセス・インスタンスの横にあるチェック・ボックスを選択し、「**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/グループ名 (New Owner / Group Name)**」フィールドで、作業項目の転送先のユーザー ID またはグループ名を入力します。ユーザーに割り当てられている作業項目は別のユーザーにのみ転送することができ、グループに割り当てられている作業項目は別のグループにのみ転送することができます。
   b. 1 つ以上の作業項目を選択し、「**転送**」をクリックします。

タスクの結果

新規の作業項目所有者を持つ転送済みプロセス作業項目が、作業項目のリストに表示されます。

関連概念

代替プロセス管理許可モード
管理タスクを組み込むようにプロセス・テンプレートをモデル化した場合は、この管理モードによって、それらの管理タスクおよび関連アクション (エスカレーションやモニターなど) が非アクティブ化されます。このモードで実行すると、インスタンス・ベースの管理が使用不可になると共に、プロセス管理およびモニターがシステム管理者またはシステム・モニターであるユーザーに制限されるので、パフォーマンスが向上します。

関連タスク

393 ページの『プロセス作業項目の作成』
プロセス・インスタンスの作業項目を使用して、そのプロセス・インスタンスの管理者権限または読者権限を管理します。例えば、プロセス・インスタンスのスターターのユーザー ID が削除されるのでユーザーがプロセス・インスタンスの所有権を要求する場合などには、新しい管理者のプロセス作業項目を作成する必要があります。また、ユーザーが変数の値を確認または変更する必要がある場合にも、作業項目の作成が必要になります。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。

『プロセス作業項目の削除』
例えば、エラーのプロセス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プロセス作業項目が生成されている場合は、プロセス作業項目を削除することができます。

408 ページの『プロセス・インスタンスの所有権の移転』
プロセス・インスタンスの所有権は、プロセス管理者権限を持つユーザーにプロセス・インスタンスの所有権を取得させることによって、移転することができます。この所有権の移転は、例えば、プロセス・スターターが会社を辞めたような状況で行うことがあります。

プロセス作業項目の削除

例えば、エラーのプロセス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プロセス作業項目が生成されている場合は、プロセス作業項目を削除することができます。

始める前に

プロセス作業項目を削除する場合、以下の条件が適用されます。

- プロセス管理者またはシステム管理者でなければなりません。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールのユーザーのみがこのアクションを実行できます。
- プロセス・インスタンスは、実行、失敗、または強制終了のいずれかの状態でなければなりません。
- 割り当て理由が `starter` である作業項目を削除することはできません。
- 割り当て理由が `administrator` である作業項目は、プロセス・インスタンスでこの割り当て理由が設定されている作業項目の中の最後の作業項目でない場合にのみ、削除することができます。

- 特定のユーザーの作業項目が全員に割り当てられている場合、その作業項目は削除できません。
- プロセス読者またはプロセス管理者の作業項目を削除すると、その削除は、含まれているすべてのアクティビティーに自動的に伝搬されます。

このタスクについて

プロセス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。

手順

1. 管理するプロセス・インスタンスを表示します。

   「ビュー」タブのナビゲーション・ペインの「プロセス・インスタンス」の下で、「**ユーザーが管理**」をクリックします。
2. プロセス・インスタンスの作業項目を表示します。

   「ユーザーが管理するプロセス・インスタンス」ページで、1 つ以上のプロセス・インスタンスを選択し、「**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。

タスクの結果

プロセス作業項目が削除されます。

関連概念

代替プロセス管理許可モード
管理タスクを組み込むようにプロセス・テンプレートをモデル化した場合は、この管理モードによって、それらの管理タスクおよび関連アクション (エスカレーションやモニターなど) が非アクティブ化されます。このモードで実行すると、インスタンス・ベースの管理が使用不可になると共に、プロセス管理およびモニターがシステム管理者またはシステム・モニターであるユーザーに制限されるので、パフォーマンスが向上します。

関連タスク

393 ページの『プロセス作業項目の作成』
プロセス・インスタンスの作業項目を使用して、そのプロセス・インスタンスの管理者権限または読者権限を管理します。例えば、プロセス・インスタンスのスターターのユーザー ID が削除されるのでユーザーがプロセス・インスタンスの所有権を要求する場合などには、新しい管理者のプロセス作業項目を作成する必要があります。また、ユーザーが変数の値を確認または変更する必要がある場合にも、作業項目の作成が必要になります。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。

394 ページの『プロセス管理者の場合のプロセス作業項目の転送』
プロセス作業項目の割り当てを変更することが必要になる場合があります。例えば、プロセス・インスタンスの所有権を他のユーザーが要求する必要があるため、そのユーザーにプロセス作業項目を転送しなければならないことがあります。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。

408 ページの『プロセス・インスタンスの所有権の移転』
プロセス・インスタンスの所有権は、プロセス管理者権限を持つユーザーにプロセス・インスタンスの所有権を取得させることによって、移転することができます。この所有権の移転は、例えば、プロセス・スターターが会社を辞めたような状況で行うことがあります。

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。

このタスクについて

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。

プロセスの障害動作は、プロセス・テンプレートの「**エラーの継続**」設定によって制御されます。「**エラーの継続**」が `no` に設定されていると、予期しない障害があった場合、影響を受けるアクティビティーが停止状態になります。

「**エラーの継続**」が `yes` に設定されている場合 (または、プロセスが 6.1.2 より前のバージョンの WebSphere Integration Developer で作成されたために設定されていない場合) は、予期しない障害が発生すると、デフォルトの障害ハンドラーが呼び出され、最終的にはプロセスが失敗状態で終了します。後者は、予期しない障害で、すぐ周囲のスコープ内に適切な障害ハンドラーが存在しないために発生します。現在の障害に対して定義された明示的な障害ハンドラーが存在しない場合に、

デフォルトの障害ハンドラーが呼び出されると、この障害ハンドラーにより現在のスコープが強制終了され、障害が周囲のスコープに伝搬されます。最終的に、これによってプロセスは失敗状態で終了します。

invoke アクティビティー、Java 断片アクティビティー、ヒューマン・タスク・アクティビティー、カスタム・アクティビティーについては、専用の「**エラーの継続**」設定をモデル化して、プロセスの設定をオーバーライドすることもできます。ただし、デフォルト値をプロセスの場合と同じ値のままにしておけば、それらのアクティビティー・タイプについても障害状態を修復できます。アクティビティー・レベルの設定によって制御されるのは、アクティビティーの実装によって障害が生成されたときの振る舞いのみです。結合条件の評価時または発信リンクの遷移条件の評価時に発生した障害は、プロセス・レベルの設定によって制御されます。例えば、アクティビティー・レベルの「**エラーの継続**」設定が yes に設定されていても、結合条件の評価が失敗した場合などには、invoke アクティビティーが停止状態になる可能性があります。

アクティビティーが停止した場合、プロセスは実行状態のままになります。この場合、Business Process Choreographer Explorer には、プロセスを修復してナビゲーションを続行するための、いくつかのオプションがあります。

手順

- 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、独自のプロセス・インスタンス検索を定義します。または、ナビゲーション・ペインで、「**アクティビティー・インスタンス**」の下の「**停止したアクティビティー**」をクリックし、次に失敗したアクティビティーの該当するプロセス・インスタンスをクリックします。
- 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。

次のタスク

これで、保留アクティビティーを修復できます。

失敗したプロセスの原因の分析

プロセスの失敗の原因となった例外について情報を確認します。プロセスが失敗状態にある場合、インスタンス自体は修復できません。しかし、問題の原因を修正して今後のインスタンスの失敗を防ぐことができる場合があります。

始める前に

プロセスが失敗状態である必要があります。

このタスクについて

プロセス・ナビゲーション中に発生した例外が、プロセスについて定義した障害でない場合、プロセスの失敗の原因となる可能性があります。

Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「ビュー」タブで、プロセスの「プロセス・インスタンス」ページに移動します。

   例えば、失敗状態のプロセスを検索するための新しいプロセス・インスタンスの検索を定義し、「**詳細**」をクリックして詳細を調べます。
2. 「**エラー詳細**」タブを選択して、プロセスの障害の詳細な情報を確認します。
3. 障害の原因を修復して、このプロセス・テンプレートが後続の障害を起こすのを回避します。

停止されたアクティビティーの変数の変更

アクティビティーの変数を確認し、アクティビティー停止の原因となっている場合はプロセス変数を修復します。

始める前に

プロセスは実行状態である必要があります。可視のアクティビティーの変数を表示するには、少なくともスコープ・リーダーまたはプロセス・リーダーの権限を持つ必要があります。変数を変更するには、スコープ管理者またはシステム管理者でなければなりません。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールのユーザーのみがこのアクションを実行できます。

このタスクについて

プロセスの存続期間中に、プロセスの動作を制御する変数の値が正しくないか欠落していることが原因で、問題が発生することがあります。アクティビティーについて可視のすべての変数にアクセスして、変数の値を変更してプロセスを修復することができます。この後、プロセス・ナビゲーションを続行できます。

Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「ビュー」タブで、「プロセス・インスタンス」ページに移動します。

   例えば、「重大なプロセス」ページでプロセス・インスタンスの名前をクリックします。「プロセス・インスタンス」ページで、「**アクティビティー**」タブをクリックして、停止したアクティビティーの名前をクリックします。

   もう一つの方法として、停止したアクティビティーを表示するために、ナビゲーション・ペインで「**アクティビティー・インスタンス**」の下の「**停止したアクティビティー**」をクリックします。次に、該当するアクティビティーをクリックします。
2. 「**変数**」ボタンをクリックして、アクティビティーについて可視のすべての変数のリストを取得します。
3. 単一の変数名を選択すると、値が表示されます。
4. 値を変更して「**保管**」をクリックすると、単一の変数の設定値が更新されます。

アクティビティーの再開

アクティビティーの変数を修復した場合などには、新しい入力データを使用してアクティビティーを再開できます。

始める前に

一般に、アクティビティーが停止状態であり、stopReason が STOP_REASON_IMPLEMENTATION_FAILED または STOP_REASON_FOLLOW_ON_NAVIGATION_FAILED の場合は、アクティビティーを再始動できます。アクティビティーのタイプによっては、アクティビティーが停止状態以外の状態であっても、アクティビティーを再始動できます。他の状態でのアクティビティーの再始動について詳しくは、アクティビティーの状態遷移図についての関連情報を参照してください。

このタスクについて

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。

手順

1. 「ビュー」タブで、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。

   例えば、「ユーザーが管理するプロセス・インスタンス」ページで、プロセス・インスタンスの名前をクリックします。「プロセス・インスタンス」ページで、「**アクティビティー**」タブをクリックして、再始動するアクティビティーの名前をクリックします。
2. 「**停止理由**」とアクティビティーの種類に応じて、 アクティビティーの再開に必要な入力データを指定できます。

   オプションで、プロセス「**エラーの継続**」設定がこのアクティビティーについてオーバーライドされるように指定することができます。アクティビティー再開時にエラーが発生した場合、アクティビティーを再度停止させるには、「**エラーの継続**」を選択解除します。
3. アクティビティーに有効期限が設定されている場合は、再始動するアクティビティーの有効期限の動作を指定します。
4. 「**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。また、障害の原因を修復できない場合、アクティビティーを強制的に完了させたい場合もあります。例えば、wait アクティビティーにおける wait 式の評価によってアクティビティーの停止が繰り返される場合に、アクティビティーを強制的に完了させる場合などです。

始める前に

一般に、アクティビティーが停止状態であり、stopReason が STOP_REASON_IMPLEMENTATION_FAILED、STOP_REASON_FOLLOW_ON_NAVIGATION_FAILED、または STOP_REASON_EXIT_CONDITION_FALSE の場合は、アクティビティーを強制完了できます。アクティビティーの種類によっては、アクティビティーが別の状態の場合にも、強制完了を利用できます。他の状態でのアクティビティーの強制完了について詳しくは、アクティビティーの状態遷移図についての関連情報を参照してください。

このタスクについて

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。

手順

1. 「ビュー」タブで、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。

   出力変数を保有するアクティビティー (invoke、ヒューマン・タスク、pick および receive アクティビティー) のデータのみを指定することができます。
3. もう一度**「強制完了」**をクリックします。

アクティビティーのスケジュール変更

Business Process Choreographer Explorer で新しい日時データを使用してアクティビティーのスケジュールを変更することができます。

始める前に

通常は、実行中、待機中、作動可能、または要求済みの状態にある pick、invoke、および staff アクティビティーのスケジュールを変更できます。さらに、timeout 式を評価できないために停止したアクティビティーを修復できます。

アクティビティーのスケジュールを変更するには、システム管理者であるか、アクティビティー、エンクロージング・スコープ、またはプロセスの管理者であることが必要です。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールのユーザーのみがこのアクションを実行できます。

このタスクについて

アクティビティーのスケジュールを変更するには、Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「ビュー」タブで、「アクティビティー・インスタンス・リスト」ページに移動します。スケジュール変更するアクティビティーを選択して、**「スケジュールを**

変更」をクリックします。あるいは、スケジュール変更するアクティビティーの名前をクリックし、「アクティビティー」ページで「**スケジュールを変更**」をクリックします。

2. 「アクティビティーのスケジュール変更」ページで、日時を指定して、アクティビティーのスケジュールを変更します。あるいは、アクティビティーのスケジュール変更を実行しないこと、またはアクティビティーのスケジュールを即時に変更することを指定できます。
3. 次に、「**OK**」をクリックします。

停止されたアクティビティーの修復

Business Process Choreographer Explorer の動的な性質により、プロセス・ナビゲーションで手動で介入できます。問題が発生したため (例えば、式の評価中) 停止されたアクティビティーを修復することができます。

始める前に

次のいずれかの状態の場合は、この手順を使用します。

- 結合条件の評価が失敗したため、アクティビティーは停止しました。**stopReason** は「*アクティブ化が失敗しました*」です。
- case 条件の評価が失敗したため、switch アクティビティーは停止しました。**stopReason** は「*実装が失敗しました*」です。
- ループ条件の評価が失敗したため、repeatUntil アクティビティーは停止しました。**stopReason** は「*実装が失敗しました*」です。
- ループ条件の評価が失敗したため、forEach アクティビティーは停止しました。**stopReason** は「*実装が失敗しました*」です。
- 遷移条件の評価が失敗したため、アクティビティーは停止しました。**stopReason** は「*後続のナビゲーションが失敗しました*」です。
- アクティビティー完了時の評価で出口条件の結果が `false` になったため、アクティビティーは停止しました。**stopReason** は「*出口条件が正しくありません*」です。

このタスクについて

通常、管理者は、アクティビティーを強制的に再試行しようとするか、またはアクティビティーを強制的に完了させようとします。これらのアクションで修復できないアクティビティーの障害については、Business Process Choreographer Explorer を使用してアクティビティーのナビゲーションをオーバーライドできます。問題の詳細については、「アクティビティー」ページの「**エラーの詳細**」をクリックしてください。修復処理を続行するには変数の値を変更することが必要になる場合があります。また、アクティビティーのジャンプに関するトピックで説明しているように、プロセス・インスタンスのあるアクティビティーから別のアクティビティーにジャンプすることもできます。アクティビティーのスキップ・オプションを使用して、失敗アクティビティーが後続のプロセス・インスタンス内でスキップされるようにマークを付けることができます。さらに、停止したアクティビティーの処理をスキップしたり、再処理時にスキップされるようにマークを付けたりすることもできます。

停止したアクティビティーを修復するには、Business Process Choreographer Explorer で該当するステップを実行します。

手順

1. 停止したアクティビティーを表示するには、ナビゲーション・ペインで、「**アクティビティー・インスタンス**」の下の「**停止したアクティビティー**」をクリックし、該当するアクティビティーをクリックします。
2. これで、保留アクティビティーのために適切な修復処理を行えます。
   - 結合条件の評価が失敗したため、アクティビティーは停止しました。**stopReason** は「*アクティブ化が失敗しました*」です。以下を実行します。
     a. 「ビュー」タブで、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、「**結合の修復**」をクリックします。
     b. 適切なオプションを選択して、処理を続行します。結合条件を再評価し、プロセス・インスタンスのナビゲーションを続行することを指定できます。あるいは、現在の分岐のナビゲーションを続行するかどうかを決定するため、アクティビティーの結合条件の値が true または false に設定されることを指定できます。

        値 `True` を指定した場合は、アクティビティーが開始します。値 `False` を指定した場合、動作は suppressJoinFailure プロセス属性によって変わります。この属性が `yes` に設定されている場合は、アクティビティーがスキップされ、このアクティビティーのすべての発信リンクの状況が `false` に設定されます。そうでない場合は、Business Process Execution Language の joinFailure 標準障害が指定されます。
     c. 「**続行**」をクリックして、アクティビティーのナビゲーションを強制します。
   - case 条件の評価が失敗したため、switch アクティビティーは停止しました。**stopReason** は「*実装が失敗しました*」です。以下を実行します。
     a. 「ビュー」タブで、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、「**強制ケース・ナビゲーション**」をクリックします。
     b. ナビゲーション中にたどる分岐を選択します。各分岐は、モデル内のそれぞれの位置に基づいて列挙されます。1 つの分岐のみを選択できます。
     c. 「**サブミット**」をクリックして、ケース・ナビゲーションを強制します。
   - ループ条件の評価が失敗したため、repeatUntil アクティビティーは停止しました。**stopReason** は「*実装が失敗しました*」です。以下を実行します。
     a. 「ビュー」タブで、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、「**次の反復を表示**」または「**ループの終了**」 をクリックし、アクティビティーのナビゲーションを強制します。
   - ループ条件の評価が失敗したため、forEach アクティビティーは停止しました。**stopReason** は「*実装が失敗しました*」です。以下を実行します。
     a. 「ビュー」タブで、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、「**forEach の修復**」をクリックします。
     b. 適切な値を指定して、処理を続行します。開始カウンターおよび最終カウンターの値を指定します。forEach アクティビティーに早期終了条件がある場合は、完了させる反復数の値も指定してください。

c. 「**続行**」をクリックして、アクティビティーのナビゲーションを強制します。

- 遷移条件の評価が失敗したため、アクティビティーは停止しました。**stopReason** は「*後続のナビゲーションが失敗しました*」です。以下を実行します。
  a. 「ビュー」タブで、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、「**強制ナビゲーション**」をクリックします。
  b. ナビゲーション中にたどるリンクの名前を選択します。表示されるリンクの名前は、プロセスのモデル化時に WebSphere Integration Developer 内で決定される名前です。任意の数のリンクを選択できます。
  c. 「**サブミット**」をクリックして、アクティビティーのナビゲーションを強制します。
- アクティビティー完了時の評価で出口条件の結果が `false` になったため、アクティビティーは停止しました。**stopReason** は「*出口条件が正しくありません*」です。以下を実行します。
  a. 「ビュー」タブで、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**」または「**強制完了**」をクリックします。
  b. アクティビティーの再始動または完了に必要なデータを指定します。
  c. 「**再始動**」または「**強制完了**」をクリックして、アクティビティーのナビゲーションを強制します。

関連概念

410 ページの『アクティビティーのジャンプ・ターゲット』
Business Process Choreographer Explorer を使用して、プロセス・インスタンスのあるアクティビティーからそのプロセス・インスタンスの別のアクティビティーにジャンプする際、選択可能なターゲット・アクティビティーのリストからターゲット・アクティビティーを選択できます。このトピックでは、ジャンプ・アクションの実行時にターゲット・アクティビティーとして機能するアクティビティーを選択する際に適用される制限事項について説明します。

関連タスク

411 ページの『アクティビティーのスキップ』
プロセス・インスタンスの処理に含まないように、アクティビティーをスキップできます。

『プロセス状態ビューによる停止されたアクティビティーの修復』
Business Process Choreographer Explorer でプロセス状態ビューを使用して、停止したアクティビティーを手動で修復することができます。

プロセス状態ビューによる停止されたアクティビティーの修復

Business Process Choreographer Explorer でプロセス状態ビューを使用して、停止したアクティビティーを手動で修復することができます。

始める前に

この手順は、結合条件、ループ条件、case 条件、遷移条件、または forEach カウンター値の評価が失敗したため、該当のアクティビティーが停止した場合に使用することができます。

このタスクを実行するには、プロセス管理者またはシステム管理者でなければなりません。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールのユーザーのみがこのアクションを実行できます。

このタスクについて

停止したアクティビティーをプロセス状態ビューを使用して修復するには、Business Process Choreographer Explorer で該当のステップを実行します。

手順

1. プロセス・インスタンスの停止したアクティビティーを表示するには、ナビゲーション・ペインで、「**プロセス・インスタンス**」 → 「**重大なプロセス**」をクリックします。その後、該当のプロセス・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。停止したアクティビティーまたはアクティビティー・ゲートウェイが 1 つ以上表示されます。あるいは、「**アクティビティー・インスタンス**」 → 「**停止したアクティビティー**」を選択します。
2. これで、発生した問題に応じて、アクティビティーを修復するための修復措置をとることができます。
   - 結合条件の評価が失敗したため、アクティビティーは停止しました。**stopReason** は「*アクティブ化が失敗しました*」です。以下を実行します。
     a. アクティビティー・ゲートウェイをクリックするか、着信リンクが 1 つだけの場合は、そのアクティビティーをクリックします。
     b. 「**結合の修復**」をクリックして、条件を再評価するか、評価の結果を強制的に true または false にするかを選択します。
   - While または Repeat Until アクティビティーのループ条件の評価が失敗しました。**stopReason** は「*実装が失敗しました*」です。以下を実行します。
     – アクティビティーをクリックして、ループを続行させる場合は「**次の反復を表示**」を、ループを終了させる場合は「**ループの終了**」を選択します。
   - switch アクティビティーの case 条件の評価が失敗しました。**stopReason** は「*実装が失敗しました*」です。以下を実行します。
     – アクティビティーをクリックして、「**強制ケース実行**」をクリックします。実行するケースを選択します。
   - 遷移条件の評価が失敗したため、アクティビティーは停止しました。**stopReason** は「*後続のナビゲーションが失敗しました*」です。以下を実行します。
     a. アクティビティー・ゲートウェイをクリックするか、発信リンクが 1 つだけの場合は、そのアクティビティーをクリックします。
     b. 「**後続のナビゲーションの修復**」をクリックします。選択可能な分岐のターゲット・ノードおよびリンクが強調表示されます。
     c. ターゲット・ノードまたはリンクをクリックし、「**この分岐の選択**」をクリックして、1 つ以上の分岐を選択します。
     d. 次に、ターゲット・ノード、リンク、またはソース・ノードをクリックし、「**強制ナビゲーション**」をクリックして、選択した分岐のナビゲーションを強制します。

- forEach アクティビティーのカウンター値の評価が失敗しました。**stopReason** は「*実装が失敗しました*」です。以下を実行します。
  a. アクティビティーをクリックし、「**ForEach の修復**」をクリックします。
  b. 開始カウンター値および終了カウンター値を入力し、実行する反復の数を必要に応じて入力して、ループを続行または終了します。

関連タスク

403 ページの『停止されたアクティビティーの修復』
Business Process Choreographer Explorer の動的な性質により、プロセス・ナビゲーションで手動で介入できます。問題が発生したため (例えば、式の評価中) 停止されたアクティビティーを修復することができます。

相関セットの修復

Business Process Choreographer Explorer を使用して、アクティビティーの相関セットを表示および変更することができます。さらに、ランタイム障害や修復アクション、あるいはプロセスのジャンプ操作が原因で不適切となったアクティビティー相関セットを修復することができます。

始める前に

アクティビティーに関連付けられた相関セットを修復するには、プロセス管理者またはシステム管理者でなければなりません。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールのユーザーのみがこのアクションを実行できます。

このタスクについて

例えば、アクティビティー相関セットの値が期待された値と一致しないためにアクティビティーが停止状態になっている場合は、その相関セットを修復します。

Business Process Choreographer Explorer で、次のステップを実行します。

手順

1. 「ビュー」タブで、アクティビティーの「アクティビティー」ページに移動し、「相関」タブを選択して、アクティビティーに関連付けられている相関セットを表示します。
2. 「**相関セットの修復**」をクリックして、相関セットを変更します。
3. 複数の相関セットが定義されている場合は、ドロップダウン・リストを使用して、変更する相関セットを選択します。
   - 相関セットが初期化されていない場合は、その相関セットの値を指定し、「**初期化 (Initialize)**」をクリックして値を保存します。
   - 相関セットが既に初期化されており、値を変更したい場合は、まず「**初期化解除 (Uninitialize)**」をクリックして、既存の値を削除します。次に、相関セットの新しい値を指定し、「**初期化 (Initialize)**」をクリックして値を保存します。

タスクの結果

すべての相関セットに正しいプロパティー値が含まれるようになります。

次のタスク

アクティビティーが停止している場合は、アクティビティーを再始動することによりプロセスの修復を続行できます。

プロセス・インスタンスの所有権の移転

プロセス・インスタンスの所有権は、プロセス管理者権限を持つユーザーにプロセス・インスタンスの所有権を取得させることによって、移転することができます。この所有権の移転は、例えば、プロセス・スターターが会社を辞めたような状況で行うことがあります。

始める前に

プロセス・インスタンスの所有権を移転するには、プロセス・インスタンス管理者またはビジネス・プロセスのシステム管理者がプロセス・インスタンスの所有権を要求します。プロセスの所有権の要求対象のプロセス・インスタンスは、いずれの状態でも構いません。

プロセス・インスタンスの所有権を要求するには、管理権限が必要です。システム管理者またはプロセス管理者は、理由に `administrator` と指定したプロセス作業項目を作成または転送することにより、この権限を割り当てることができます。

このタスクについて

プロセス・インスタンスの所有権を要求するには、Business Process Choreographer Explorer で以下のステップを実行します。

手順

1. プロセス・インスタンスのリストを表示します。

   例えば、「ビュー」タブのナビゲーション・ペインの「プロセス・インスタンス」の下で、「**ユーザーが管理**」をクリックします。
2. プロセスの所有権を要求します。

   プロセス・インスタンス (1 つまたは複数) の横にあるチェック・ボックスを選択して、「**要求の所有権**」をクリックします。

タスクの結果

これで、プロセス・インスタンスの所有権を取得し、プロセス・スターターとなり、またプロセス・インスタンスのプロセス管理者権限を取得しました。

関連タスク

394 ページの『プロセス管理者の場合のプロセス作業項目の転送』
プロセス作業項目の割り当てを変更することが必要になる場合があります。例えば、プロセス・インスタンスの所有権を他のユーザーが要求する必要があるため、そのユーザーにプロセス作業項目を転送しなければならないことがあります。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。

393 ページの『プロセス作業項目の作成』
プロセス・インスタンスの作業項目を使用して、そのプロセス・インスタンスの管理者権限または読者権限を管理します。例えば、プロセス・インスタンスのスターターのユーザー ID が削除されるのでユーザーがプロセス・インスタンスの所有権を要求する場合などには、新しい管理者のプロセス作業項目を作成する必要があります。また、ユーザーが変数の値を確認または変更する必要がある場合にも、作業項目の作成が必要になります。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。

396 ページの『プロセス作業項目の削除』
例えば、エラーのプロセス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プロセス作業項目が生成されている場合は、プロセス作業項目を削除することができます。

アクティビティーのジャンプ

プロセス・インスタンスのあるアクティビティーからそのプロセス・インスタンスの別のアクティビティーにジャンプすることができます。ソース・アクティビティーが完了してからターゲット・アクティビティーにジャンプするように選択できます。

始める前に

このアクションを実行するには、システム管理者であるか、あるいはソース・アクティビティーとターゲット・アクティビティーが属するスコープまたは親スコープのプロセス管理者またはスコープ管理者である必要があります。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールのユーザーのみがこのアクションを実行できます。

このタスクについて

あるアクティビティーから別のアクティビティーにジャンプするには、Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「ビュー」タブで、プロセス・インスタンスの「プロセス状態」ページに移動します。
2. プロセス状態遷移で適切なアクティビティーをクリックします。

   ジャンプ・アクションは、プロセス状態遷移の「**詳細レベル**」スライダーが高レベルになっている場合のみ使用可能であることに注意してください。

3. 別のアクティビティーに移動するには、「**別のアクティビティーにジャンプ**」をクリックします。

   このオプションは、実行状態 (例えば、作動可能、要求済み、実行中、停止、待機中) のアクティビティーについてだけ選択できます。

   プロセス状態遷移が再度表示され、ターゲット・アクティビティーに使用できるアクティビティーのみが選択可能になります。ターゲット・アクティビティーについては、アクティビティーのジャンプ・ターゲットについての関連情報を参照してください。
4. 実行するアクションを選択するためにターゲット・アクティビティーを選択します。

   使用可能なアクションは、ソース・アクティビティーにより異なります。
5. 実行するアクションを選択します。
   - ソース・アクティビティーを完了してからターゲット・アクティビティーにジャンプするには、「**ソース・アクティビティーを完了してジャンプ**」をクリックします。

     「**ソース・アクティビティーを完了してジャンプ**」オプションがターゲット・アクティビティーで使用可能なのは、ソース・アクティビティーが要求済み状態のヒューマン・タスク・アクティビティーである場合のみです。このオプションを選択すると、ソース・アクティビティーの完了後にターゲット・アクティビティーにジャンプします。
   - ソース・アクティビティーを強制完了してからターゲット・アクティビティーにジャンプするには、「**ソース・アクティビティーを強制完了してジャンプ**」をクリックします。提供したデータでアクティビティーを完了するには、「**強制完了してジャンプ**」をクリックします。

     「**ソース・アクティビティーを強制完了してジャンプ**」オプションがターゲット・アクティビティーで使用可能となるのは、ソース・アクティビティーが作動可能、要求済み、または停止状態のヒューマン・タスク・アクティビティーである場合です。そのオプションは、実行中または停止状態の invoke アクティビティー、待機中または停止状態の receive アクティビティーまたは wait アクティビティー、および他のすべての停止状態の基本アクティビティーでも使用できます。このオプションを選択すると、ソース・アクティビティーの強制完了後にターゲット・アクティビティーにジャンプします。
   - アクティビティーをスキップして別のアクティビティーにジャンプするには、「**ソース・アクティビティーをスキップしてジャンプ**」をクリックします。
   - ジャンプ・アクションを取り消すには、「**ジャンプの取り消し**」をクリックします。

アクティビティーのジャンプ・ターゲット:

Business Process Choreographer Explorer を使用して、プロセス・インスタンスのあるアクティビティーからそのプロセス・インスタンスの別のアクティビティーにジャンプする際、選択可能なターゲット・アクティビティーのリストからターゲット・アクティビティーを選択できます。このトピックでは、ジャンプ・アクション

の実行時にターゲット・アクティビティーとして機能するアクティビティーを選択する際に適用される制限事項について説明します。

プロセス・インスタンスのナビゲーション中は、あるアクティビティーから同じ sequence 内または同じ循環 flow 内で直接ネストされたアクティビティーにのみジャンプできます。また、ソース・アクティビティーとターゲット・アクティビティーが一連の flow リンクによって接続されていて、中間に存在するすべてのアクティビティーに接続される他のリンクがない場合は、flow 内でジャンプすることもできます。

- sequence アクティビティー内でアクティビティーのジャンプを実行できます。これは、ジャンプのソース・アクティビティーとターゲット・アクティビティーが同じ sequence 内に存在する必要があり、両方とも他の構造化アクティビティー内でネストされていないことを意味します。
- flow アクティビティー内でアクティビティーのジャンプを実行できます。この場合、ジャンプのソース・アクティビティーとターゲット・アクティビティーは flow アクティビティー内で直接ネストでき、ソースからターゲットへの制御フロー内にパスがただ 1 つしか存在しないはずです。
- また、スコープ内に含まれたアクティビティーが 1 つしかない場合は、スコープの外側にジャンプできます。例えば、接続されたハンドラーで invoke アクティビティーからジャンプすることができます。
- また、循環 flow 内でアクティビティーのジャンプを実行することもできます。これによって、ジャンプのソース・アクティビティーとターゲット・アクティビティーが循環 flow 内で直接ネストされ、他の構造化アクティビティー内ではネストされません。

関連タスク

403 ページの『停止されたアクティビティーの修復』
Business Process Choreographer Explorer の動的な性質により、プロセス・ナビゲーションで手動で介入できます。問題が発生したため (例えば、式の評価中) 停止されたアクティビティーを修復することができます。

アクティビティーのスキップ

プロセス・インスタンスの処理に含まないように、アクティビティーをスキップできます。

このタスクについて

スキップするアクティビティーにマークを付けるには、 Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「ビュー」タブで、プロセス・インスタンスの「プロセス状態」ページに移動します。
2. プロセス状態遷移で適切なアクティビティーをクリックします。

   スキップ・アクションおよびジャンプ・アクションは、プロセス状態遷移の「**詳細レベル**」スライダーが高レベルになっている場合のみ使用可能であることに注意してください。
3. 以下のいずれかのスキップ・アクションを実行します。

- スキップするアクティビティーにマークを付けるには、「**アクティビティーのスキップ**」をクリックします。これにより、アクティビティーは「スキップの要求」アイコン  で示されます。スキップされたアクティビティーは、「スキップ」アイコン  で示されます。

  このアクションを実行するには、ソース・アクティビティーとターゲット・アクティビティーが属するスコープあるいは親スコープのプロセス管理者、またはスコープ管理者である必要があります。

  「**アクティビティーのスキップ**」アクションは、あらゆるアクティビティーの状態で使用可能です。終了状態にあるアクティビティーはスキップ対象のマークが付きますが、アクティビティーの状態はナビゲーションによって再度到達されるまで変更されないままです。そのため、アクティビティーが既に終了状態にある場合、アクティビティーは再度アクティブ化されると即時にスキップされます。
- スキップするアクティビティーのマークを解除するには、「**スキップの取り消し**」をクリックします。これによって、前に選択済みのアクティビティーのスキップ要求が取り消されます。
- アクティビティーのスキップおよび別のアクティビティーにジャンプする別の方法としては、「**別のアクティビティーにジャンプ**」をクリックします。

  プロセス状態遷移が再度表示され、ターゲット・アクティビティーに使用できるアクティビティーのみが選択可能になります。使用可能なオプションはソース・アクティビティーにより異なることに注意してください。

  アクティビティーをスキップして別のアクティビティーにジャンプするには、「**ソース・アクティビティーをスキップしてジャンプ**」をクリックします。

関連タスク

403 ページの『停止されたアクティビティーの修復』
Business Process Choreographer Explorer の動的な性質により、プロセス・ナビゲーションで手動で介入できます。問題が発生したため (例えば、式の評価中) 停止されたアクティビティーを修復することができます。

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。やり直しアクションは、Microflow のトランザクションに参加していないアクションを実行するアクティビティーにのみ必要です。

始める前に

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。

このタスクについて

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。

手順

1. 失敗した補正アクションのリストを表示します。

   「ビュー」タブのナビゲーション・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。

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。

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、インストール済みの各タスク・テンプレートを個別に開始および停止します。

始める前に

WebSphere 管理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。

このタスクについて

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。

手順

1. 管理するモジュールを選択します。

   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**」 → 「**SCA モジュール**」 → 「***module_name***」をクリックします。
2. 「**追加プロパティー**」の下の SCA 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タスク・テンプレートを選択します。
3. 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。
4. 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。

管理スクリプトによるタスク・テンプレートの停止および開始

管理スクリプトは、タスクおよびプロセス・テンプレートを停止および開始するために、管理コンソールの代わりに使用できます。管理スクリプトを使用して、アプリケーションに属するすべてのテンプレートを停止および開始します。

始める前に

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。

- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にオペレーター権限も管理者権限もない場合は、wsadmin `-user` および `-password` のオプションを付けて、オペレーターまたは管理者の権限を持つユーザー ID を指定します。
- テンプレートが停止または開始されるアプリケーション・サーバーは、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。

このタスクについて

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、テンプレートが開始状態になります。

手順

1. 管理スクリプトがある 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. タスクおよびプロセス・テンプレートを停止するには、次のようにします。

   以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin -f bpcTemplates.jacl
                            -stop application_name
   ```

   各部の意味は、次のとおりです。

   -stop *application_name*
   指定されたアプリケーションに属するすべてのテンプレートが停止します。

   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. タスクおよびプロセス・テンプレートを開始します。

```
install_root/bin/wsadmin -f bpcTemplates.jacl
                         -start application_name
```

アプリケーションに属するテンプレートが開始します。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。

このタスクについて

最新の有効開始日付を持つ、インストールおよび開始されたタスク・テンプレートはすべて、Business Process Choreographer Explorer のタスク・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。

手順

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。

   「ビュー」タブのナビゲーション・ペインの「タスク・テンプレート」の下で、「**ユーザーのタスク・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。

タスクの結果

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。

このタスクについて

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。

手順

1. 自分に割り当てられているタスクを表示します。

   「ビュー」タブで、「**タスク・インスタンス**」 → 「**ユーザーの予定**」をクリックします。

   このアクションにより、「ユーザーの予定」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。

タスクの結果

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。

タスク・インスタンスの中断と再開

Business Process Choreographer Explorer を使用して、タスク・インスタンスを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・インスタンスが失敗する原因になっている問題を修正できます。タスクの前提条件が満たされている場合は、タスク・インスタンスの実行を再開できます。

始める前に

タスク・インスタンスを中断して再開するには、タスク管理者権限が必要です。

タスク・インスタンスを中断するには、タスク・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。タスクを再開するには、タスク・インスタンスが中断された状態である必要があります。

タスクの中断は、WebSphere Application Server の単純カレンダーを使用するヒューマン・タスクの場合にのみサポートされます。

このタスクについて

タスク・インスタンスを中断するには、Business Process Choreographer Explorer で次のステップを完了します。

手順

1. 管理できるタスク・インスタンスを表示します。

   「ビュー」タブのナビゲーション・ペインの「タスク・インスタンス」の下で、「**ユーザーが管理**」をクリックします。

2. 「タスク・インスタンス」ページで、「**中断**」をクリックします。
3. 以下のいずれかのオプションを選択して、タスク・インスタンスを中断します。
   - タスクを手動で再開するまで中断するには、「**中断**」を選択します。
   - 特定の時刻までタスクを中断するには、「**次の期間までタスクを中断**」を選択して日時を指定します。
   - 一定の期間にわたってタスクを中断するには、「**タスクの中断期間**」を選択して期間を指定します。
4. 選択を確認するには、「**実行依頼**」をクリックします。 タスク・インスタンスは中断状態になります。

次のタスク

中断状態のタスク・インスタンスを再開するには、「**再開**」をクリックします。

タスク・インスタンスの再始動

Business Process Choreographer Explorer を使用して、タスク・インスタンスを再始動することができます。このタスク・インスタンスの再始動は、例えば、既に実行中であるが期待どおりに進行していないヒューマン・タスク、または失敗や期限切れなどの予期しない、または望ましくない終了状態に達したタスクに対して実行します。また、それらを再始動する前にタスクの入力メッセージの値を変更することができます。再度同じ作業を開始するために再使用したいタスクを再始動することができます。このタスクには、完了しているヒューマン・タスク、例えば、呼び出しタスクやコラボレーション・タスクも含まれます。このタスクは通常、変更した入力メッセージを使用して再始動します。

始める前に

タスク・インスタンスは、コラボレーション・タスク、呼び出しタスク、または予定タスクのいずれかです。タスク・インスタンスは、非アクティブを除くいずれかの状態にあります。また、以下の条件があります。

- 呼び出しタスクは、実行状態であることはできません。
- 予定タスクは、終了状態 (つまり、終了、失敗、強制終了、または期限切れ) であることはできません。予定タスクが転送済みの場合は、後続タスクは終了状態であることはできません。
- インライン予定タスクは、作動可能状態であることはできません。

タスク・インスタンスは、エスカレート済み、中断、またはサブタスクの待機中であることが可能です。呼び出し元は、タスク・インスタンスのスターター、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・インスタンスを再始動すると、担当者解決が新規に実行され、すべてのタイマーがリセットされます。すべてのサブタスクまたは後続タスクが削除されます。エスカレーションもすべて取り消され、非アクティブ状態にリセットされます。呼び出しタスクの場合、ログオン・ユーザーが再始動されるタスク・インスタンスのスターターになります。

このタスクについて

タスク・インスタンスを再始動するには、Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「ビュー」タブで、タスクの「タスク」ページに移動して「**再始動**」をクリックします。

   例えば、「ユーザーが管理するタスク・インスタンス」ページで、タスク・インスタンスのチェック・ボックスを選択して「**再始動**」をクリックします。
2. 「**再始動**」をクリックして、提供した情報でタスクを再度開始します。

タスク・インスタンスのスケジュール変更

Business Process Choreographer Explorer で新しい日時データを使用してタスク・インスタンスのスケジュールを変更することができます。

始める前に

タスクの有効期限時刻、削除時刻、および期限を更新、スケジュール変更、停止、および再始動することができます。タスクが非アクティブ状態のとき、つまりタスクが作成されてからタスクが開始するまでの間、タスク・オリジネーターは、スケジュール変更を実行しないか、スケジュールを即時に変更するためにのみ、時間のスケジュール変更を更新できます。これにより、タスク開始の時刻設定のスケジューリングが影響を受けます。タスクが作成されてからタスクが開始するまでの間、タスク・オリジネーターはタスク期間を変更できます。

タスクが作動可能、要求済み、または実行中の状態のときは、期限および有効期限時刻の設定をスケジュール変更できます。

タスクの期限設定を変更するには、タスクの所有者、スターター、オリジネーター、編集者、または管理者でなければなりません。有効期限時刻の設定を変更するには、タスクのオリジネーターまたは管理者でなければなりません。

このタスクについて

タスクのスケジュールを変更するには、Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「ビュー」タブで、タスクの「タスク・インスタンス」ページに移動して「**スケジュールを変更**」をクリックします。

   例えば、「ユーザーが管理するプロセス・インスタンス」ページで、プロセス・インスタンスの名前をクリックします。「プロセス・インスタンス」ページで、「**タスク**」タブをクリックして、スケジュール変更するタスクの名前をクリックします。「タスク・インスタンス」ページで「**スケジュールを変更**」をクリックします。

2. 「タスクのスケジュール変更」ページで、変更する時刻設定を選択し、タスクのスケジュール変更の日時を指定します。あるいは、タスクのスケジュール変更を実行しないこと、またはタスクのスケジュールを即時に変更することを指定できます。
3. 「**OK**」をクリックして、タスクのスケジュールを変更します。

ヒューマン・タスクの優先順位の管理

ヒューマン・タスクの優先順位を使用すると、タスクをフィルターに掛けたり、タスクのリストをソートしたりすることができます。

このタスクについて

タスク・インスタンスの優先順位を変更するには、Business Process Choreographer Explorer で次のステップを実行します。

手順

1. タスク・インスタンスのリストを表示します。

   例えば、「ビュー」タブのナビゲーション・ペインの「タスク・インスタンス」の下で、「**ユーザーの予定**」をクリックします。
2. 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、「**優先順位の変更**」をクリックします。
3. 値を入力し、「**実行依頼**」をクリックします。

   タスク・インスタンスの優先順位が、新しい値に設定されます。

次のタスク

タスクのリストを優先順位でソートするには、テーブル・ヘッダーの矢印をクリックします。

作業割り当ての管理

タスクまたはプロセス・インスタンスの開始後に、そのタスクまたはプロセス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)。

このタスクについて

*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・インスタンスまたはプロセス・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、タスクまたはプロセス・インスタンスの開始後にそれらのインスタンス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。不在時の不在設定を指定したほうがよい場合があります。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)。

このタスクについて

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。

手順

1. 所有するタスクを表示します。

   「ビュー」タブのナビゲーション・ペインの「タスク・インスタンス」グループで「**ユーザーの予定**」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。
   a. 「**新規所有者/グループ名 (New Owner / Group Name)**」フィールドで、作業項目の転送先のユーザー ID またはグループ名を入力します。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。ユーザーに割り当てられている作業項目は別のユーザーにのみ転送することができ、グループに割り当てられている作業項目は別のグループにのみ転送することができます。
   b. 1 つ以上の作業項目を選択し、「**転送**」をクリックします。

タスクの結果

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。

関連タスク

423 ページの『不在設定の指定』
ある程度の時間にわたってオフィスから離れる場合は、タスクの代理人を指定します。

424 ページの『ユーザーの不在設定の指定』
ユーザーがタスクを操作できない場合 (病欠など)、そのユーザーのタスクの代理人を指定します。

425 ページの『タスク作業項目の作成』
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タスク作業項目を作成できます。また、担当者ディレク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、タスク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。

『スターター、オリジネーター、またはタスクの管理者の場合のタスク作業項目の転送』
タスクで作業を始めた後にタスク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、タスク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。

426 ページの『タスク作業項目の削除』
例えば、エラーのタスク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タスク作業項目が生成されている場合は、タスク作業項目を削除することができます。

スターター、オリジネーター、またはタスクの管理者の場合のタスク作業項目の転送

タスクで作業を始めた後にタスク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、タスク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。

始める前に

タスク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、割り当て理由に従ってタスクは次の状態のいずれかになっている必要があります。

ロール	理由	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	所有者	要求	潜在的所有者、管理者。
スターター	スターター	期限切れ、強制終了、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	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。

ロール	理由	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。
オリジネーター	潜在的スターター	非アクティブ	任意のユーザー・ロール。
管理者	スターター	期限切れ、強制終了、終了、失敗、または実行中	スターター。
管理者	潜在的スターター	非アクティブ	潜在的スターター。
管理者	読者または管理者	非アクティブ状態以外のすべての状態	読者、管理者。
管理者	潜在的な所有者または編集者	作動可能、または要求済み	潜在的な所有者、または編集者。

このタスクについて

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。

手順

1. 管理できるタスク・インスタンスを表示します。

   「ビュー」タブのナビゲーション・ペインの「タスク・インスタンス」の下で、**「ユーザーが管理」**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。

   「ユーザーが管理するタスク・インスタンス」ページで、1 つ以上のタスクの横にあるチェック・ボックスを選択し、**「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/グループ名 (New Owner / Group Name)」**フィールドで、作業項目の転送先のユーザー ID またはグループ名を入力します。ユーザーに割り当てられている作業項目は別のユーザーにのみ転送することができ、グループに割り当てられている作業項目は別のグループにのみ転送することができます。
   b. 1 つ以上の作業項目を選択し、**「転送」**をクリックします。

タスクの結果

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。

関連タスク

424 ページの『ユーザーの不在設定の指定』
ユーザーがタスクを操作できない場合 (病欠など)、そのユーザーのタスクの代理人を指定します。

425 ページの『タスク作業項目の作成』
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タスク作業項目を作成できます。また、担当者ディレク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、タスク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。

420 ページの『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)。

426 ページの『タスク作業項目の削除』
例えば、エラーのタスク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タスク作業項目が生成されている場合は、タスク作業項目を削除することができます。

不在設定の指定

ある程度の時間にわたってオフィスから離れる場合は、タスクの代理人を指定します。

始める前に

このタスクを実行する場合は、代替のために Virtual Member Manager 担当者ディレクトリー・プロバイダーが必要になります。Business Process Choreographer 内の Human Task Manager で代替を使用可能にする必要もあります。代替を使用可能にすると、「**代理人**」オプションがタスクバーに表示されます。

このタスクについて

適用された代替ポリシーに従って、1 人以上の代理人が、ユーザーの不在時にユーザーの作業割り当てを受け取ることができます。代替ポリシーはタスク・テンプレートごとに異なってもかまいません。Business Process Choreographer Explorer で次のステップを実行します。

手順

1. タスクバーの「**代理人**」をクリックします。
2. 「代理人」ページで不在設定を指定し、「**保管**」をクリックします。
   a. 不在設定を使用可能にするには、「**不在にしています**」チェック・ボックスを選択します。
   b. 「**代理人**」フィールドに、代理人のユーザー ID を入力し、「**追加**」をクリックします。
   c. オプション: 必要に応じて、さらに代理人を追加します。 適用された代替ポリシーに従って、1 人以上の代理人が、ユーザーの不在時にユーザーの作業割り当てを受け取ることができます。代替ポリシーはタスク・テンプレートごとに異なってもかまいません。

d. オプション: リストから代理人を削除するには、代理人のユーザー ID を選択して、「**削除**」をクリックします。 複数の代理人を選択するには、Ctrl キーを押して操作します。

3. TaskSystemAdministrator に担当者の照会結果を最新表示するように依頼します。

タスクの結果

「**不在にしています**」チェック・ボックスを選択している間は、代理人が作業割り当てを受信します。

次のタスク

「**不在にしています**」チェック・ボックスを選択する前に割り当てられた作業割り当ては、別途転送する必要があります。

関連タスク

420 ページの『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)。

ユーザーの不在設定の指定

ユーザーがタスクを操作できない場合 (病欠など)、そのユーザーのタスクの代理人を指定します。

始める前に

このタスクを実行するには、TaskSystemAdministrator 権限が必要です。また、代替のために Virtual Member Manager 担当者ディレクトリー・プロバイダーも必要です。Business Process Choreographer 内の Human Task Manager で代替を使用可能にする必要があります。そのようにすると、「**代理人の定義**」オプションがタスクバーに表示されます。

このタスクについて

Business Process Choreographer Explorer で次のステップを実行します。

手順

1. タスクバーの「**代理人の定義**」をクリックします。
2. 「代理人の定義」ページで、不在設定を指定して「**保管**」をクリックします。
   a. 不在設定を指定するユーザーのユーザー ID を入力します。
   b. 不在設定を使用可能にするには、「**ユーザーが不在です**」 チェック・ボックスを選択します。
   c. 「**ユーザーの代理人**」フィールドに代理人として指名するユーザー ID を入力し、「**追加**」をクリックします。
   d. オプション: 必要に応じて、さらに代理人を追加します。 適用された代替ポリシーに従って、1 人以上の代理人が、ユーザーの不在時にユーザーの作業割り当てを受け取ることができます。代替ポリシーはタスク・テンプレートごとに異なってもかまいません。

e. オプション: リストから代理人を削除するには、代理人のユーザー ID を選択して、「**削除**」をクリックします。 複数の代理人を選択するには、Ctrl キーを押して操作します。

3. 担当者照会結果を最新表示します。

タスクの結果

「**ユーザーが不在です**」チェック・ボックスを選択している間は、そのユーザーの作業割り当てを代理人が受信します。

次のタスク

「**ユーザーが不在です**」チェック・ボックスを選択する前に不在ユーザーに割り当てられた作業割り当ては、別途転送する必要があります。

関連タスク

421 ページの『スターター、オリジネーター、またはタスクの管理者の場合のタスク作業項目の転送』
タスクで作業を始めた後にタスク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、タスク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。

420 ページの『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)。

タスク作業項目の作成

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タスク作業項目を作成できます。また、担当者ディレク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、タスク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。

始める前に

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。

このタスクについて

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。

手順

1. 管理するタスク・インスタンスを表示します。

「ビュー」タブのナビゲーション・ペインの「タスク・インスタンス」の下で、**「ユーザーが管理」**をクリックします。

2. 作業項目の作成対象となる 1 つ以上のタスク・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。 「タスク作業項目の作成」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。

タスクの結果

作業項目は、新規作業項目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。

関連タスク

421 ページの『スターター、オリジネーター、またはタスクの管理者の場合のタスク作業項目の転送』
タスクで作業を始めた後にタスク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、タスク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。

420 ページの『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)。

『タスク作業項目の削除』
例えば、エラーのタスク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タスク作業項目が生成されている場合は、タスク作業項目を削除することができます。

タスク作業項目の削除

例えば、エラーのタスク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タスク作業項目が生成されている場合は、タスク作業項目を削除することができます。

始める前に

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・インスタンスの作業項目を削除できます。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、タスク・インスタンスが強制終了または期限切れ状態の場合にも作業項目を削除できます。

このタスクについて

タスク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。

手順

1. 管理するタスク・インスタンスを表示します。

   「ビュー」タブのナビゲーション・ペインの「タスク・インスタンス」の下で、「**ユーザーが管理**」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。

   「ユーザーが管理するタスク・インスタンス」ページで、1 つ以上のタスク・インスタンスを選択し、「**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。

タスクの結果

タスク作業項目が削除されます。

関連タスク

425 ページの『タスク作業項目の作成』
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タスク作業項目を作成できます。また、担当者ディレク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、タスク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。

421 ページの『スターター、オリジネーター、またはタスクの管理者の場合のタスク作業項目の転送』
タスクで作業を始めた後にタスク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、タスク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。

420 ページの『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)。

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。

このタスクについて

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。

Business Process Choreographer Explorer で次のステップを実行します。

手順

エスカレーションを表示するには、「ビュー」タブのナビゲーション・ペインの「タスク・インスタンス」の下にある「**ユーザー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。

始める前に

以下の規則が、エスカレーション E メールに適用されます。

- 担当者ディレクトリー・プロバイダーは、Lightweight Directory Access Protocol (LDAP) や Virtual Member Manager などの E メール・アドレスの仕様をサポートする必要があります。
- 「**全員**」、「**ユーザーなし**」、「**グループ**」、および「**ユーザー ID 別のユーザー (Users by user ID)**」の担当者割り当て基準はサポートされません。例えば、「**ユーザー ID 別のユーザー・レコード (User records by user ID)**」を代わりに使用してください。

手順

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にある「**個人ディレクトリー (JNDI 名)**」フィールドの値を編集します。

      このフィールドの値を以下のいずれかに設定します。

      - `bpe/staff/samplevmmconfiguration`
      - `bpe/staff/samplevmmconfiguration`
      - 選択した担当者ディレクトリー構成名 (JNDI 名)

   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**」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。

タスク固有情報を組み込む変数をテキストに挿入するには、「**変数の追加**」をクリックして、該当する変数をリストから選択します。エディターでは、変数は「%」文字に囲まれて表示されますが、E メールが送信されるランタイム環境での実行中に変数が評価されるときに置換されます。

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。認証が使用可能にされている場合、SMTP ホストのユーザー ID およびパスワードを設定します。

管理コンソールで、「**リソース**」 → 「**メール**」 → 「**メール・セッション**」 → 「***HTMMailSession_nodeName_serverName***」をクリックしてこの設定を確認するか、または Business Process Choreographer がクラスター上で構成されている場合は「**リソース**」 → 「**メール**」 → 「**メール・セッション**」 → 「***HTMMailSession_clusterName***」をクリックします。 SMTP ホストはセル・レベルで定義されます。

b. Human Task Manager の構成時に指定した送信側の E メール・アドレス (「**送信者の E メール・アドレス**」) が有効な E メール・アドレスであることを確認します。

管理コンソールで、「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックしてこの設定を確認するか、または Business Process Choreographer がクラスター上で構成されている場合には「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックします。「**構成**」タブの「ビジネス・インテグレーション」セクションの下で、「**Business Process Choreographer**」 → 「**Human Task Manager**」をクリックします。

次のタスク

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。

Business Process Choreographer Explorer でのカスタム・プロパティーの作成および編集

プロセス・インスタンス、アクティビティー・インスタンス、またはタスク・インスタンスに追加のプロパティーを指定するには、新規カスタム・プロパティーを作成します。

このタスクについて

インスタンスのカスタム・プロパティーを作成するには、Business Process Choreographer Explorer で次のステップを実行します。

手順

1. プロセス・インスタンス、アクティビティー・インスタンス、またはタスク・インスタンスのリストを表示させてインスタンスの名前をクリックすると、詳細ページが開きます。

   例えば、タスク・インスタンスのリストを開くには、「ビュー」タブのナビゲーション・ペインの「タスク・インスタンス」の下で、**「ユーザーの予定」**をクリックします。
2. 「カスタム・プロパティー」タブで、**「追加」**を追加します。
3. **「プロパティー名」**フィールドにカスタム・プロパティーの名前を入力し、**「プロパティー値」**フィールドには値を入力します。
4. オプション: カスタム・プロパティーを追加するには、ステップ 2 に進みます。
5. オプション: 新しいカスタム・プロパティーを削除するには、カスタム・プロパティーの横にある**「削除」**アイコンをクリックします。
6. オプション: カスタム・プロパティーのプロパティー名または値を変更するには、カスタム・プロパティーをクリックして新しい値を入力します。
7. **「保管」**をクリックします。 カスタム・プロパティーを保存した後にプロパティー名の変更およびカスタム・プロパティーの削除を行うことはできません。

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、プロセス、アクティビティー、またはタスクの状態が変わると、イベントが生成されることがあります。これらのイベントは保管され、Business Process Choreographer Explorer によるレポート作成に利用できます。例えば、プロセスのパフォーマンス問題の分析レポートや、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。

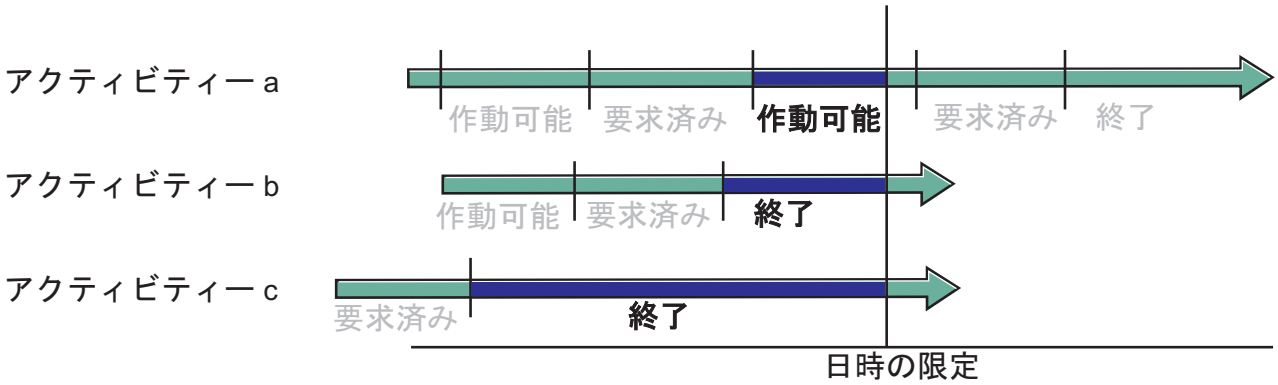
このタスクについて

Business Process Choreographer Explorer の「レポート」タブで、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。「レポート」タブは、レポート作成が構成されている場合にのみ表示されます。レポート作成機能は、Business Process Choreographer Explorer の構成時に構成できますが、後で構成することもできます。また、Event Collector アプリケーションをインストールおよび構成する必要もあります。

スナップショット・レポート

Business Process Choreographer Explorer のスナップショット・レポートを使用して、特定の日時におけるアクティビティーまたはプロセスの状態を判断します。

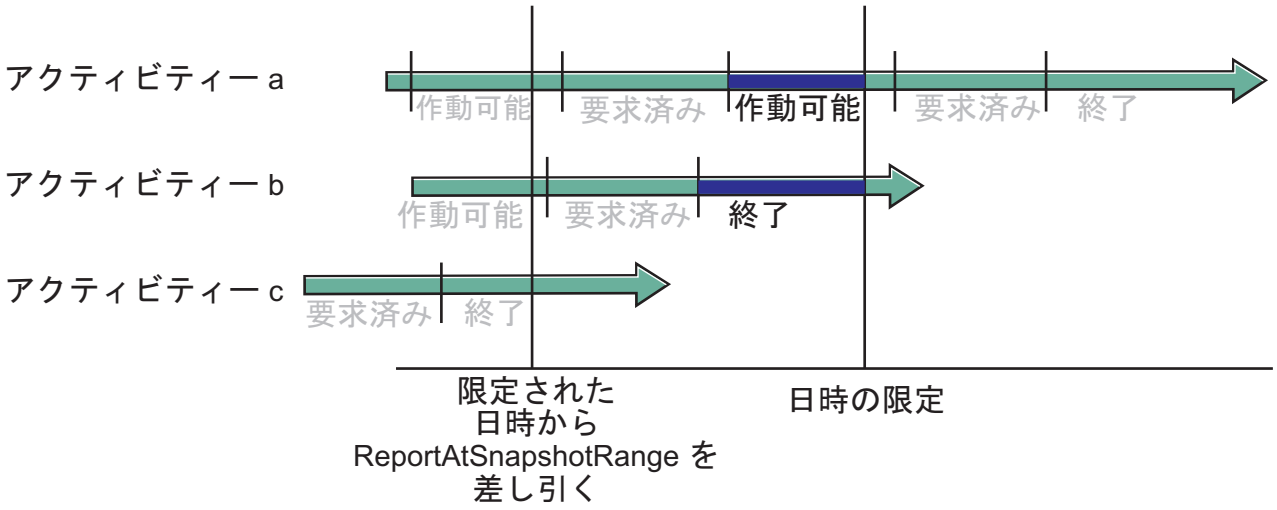
例えば、深夜に実行されているプロセス・インスタンスの数を知りたいとします。Business Process Choreographer Explorer は、プロセスまたはアクティビティーのインスタンスごとに、指定された日時より前の最後のイベントを検出して結果の状態を評価します。以下の状態遷移は、スナップショット・レポートの対象となるイベントをどのように絞り込むかを示しています。



このスナップショットでは、作動可能状態の 1 つのアクティビティー (アクティビティー a) と、終了状態の 2 つのアクティビティー (アクティビティー b および c) が含まれます。

構成パラメーター ReportAtSnapshotRange

レポート・データベースに、対象となる期間が長いプロセス・インスタンス・データが含まれている場合は、スナップショットを取得するのに時間がかかります。関係がなくなったイベントを照会しないようにするには、ReportAtSnapshotRange 構成パラメーターを使用します。指定された日時から ReportAtSnapshotRange 構成パラメーターの値を差し引いた日時よりも新しいイベントのみが、レポートの対象と見なされます。以下の状態遷移は、ReportAtSnapshotRange パラメーターが設定されているときに、スナップショット・レポートの対象としてイベントが条件を満たすときの様子を示しています。



このスナップショットでは、作動可能状態の 1 つのアクティビティー (アクティビティー a) と、終了状態の 1 つのアクティビティー (アクティビティー b) が含まれます。レポートでは、アクティビティー c の状況は戻されません。

レポート作成サイクル

スナップショット・レポートのレポート作成サイクルを定義することができます。このオプションを使用して、複数の日付に対する繰り返しスナップショットを含む

レポートを作成します。例えば、3 月中に開始したプロセス数を毎日レポート作成するとします。この場合は、毎日別々にレポート作成する必要はありません。代わりに、開始日を 3 月 1 日に、開始日以後のスナップショットの数を 31 に、スナップショット間の時間を 1 日に定義することができます。結果のレポートには、タイム・スライス数を示す追加列が含まれています。各タイム・スライスの値は対象日を示します。

関連タスク

435 ページの『事前定義のリストおよび図表の使用』
Business Process Choreographer Explorer の事前定義のリストおよび図表では、ランタイム・エンティティーの状態およびイベント情報を入手するためにドリルダウン方式が採用されています。ドリルダウン処理の各ステップで、対象とする情報の種類を詳細に定義します。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。

436 ページの『事前定義スナップショット・グラフの作成』
指定の日時におけるプロセス・インスタンスまたはアクティビティー・インスタンスの状態の分布を表示させるには、Business Process Choreographer Explorer の事前定義のスナップショット・グラフを使用します。

439 ページの『例: 事前定義のグラフの使用』
このシナリオでは、Business Process Choreographer Explorer での事前定義のグラフの使用例を示します。

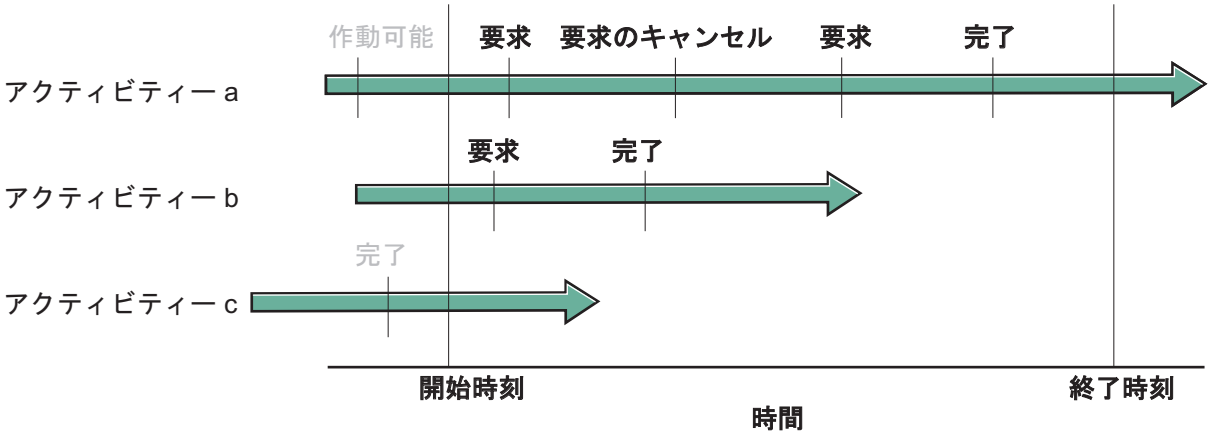
446 ページの『ユーザー定義スナップショット・レポートの作成』
指定の日時に状態情報のスナップショットを取得するユーザー定義レポートを Business Process Choreographer Explorer で定義できます。レポート作成期間内の定期的な (各月の初日の午前 0 時などの) 状態スナップショットを含むレポートを作成することもできます。

期間レポート

Business Process Choreographer Explorer の期間レポートを使用して、一定期間に特定のアクティビティー・イベントまたはプロセス・イベントが発生する頻度を判別します。

期間ビューで、報告期間の開始日と終了日を指定します。レポートの対象は、その 2 つの日付の間に入る期間です。例えば、その日に要求のあったスタッフ・アクティビティーの数を知りたいとします。

以下の状態遷移は、期間レポートに対してイベントが限定される様子を示しています。次の例に示される期間を包含するレポートには、アクティビティー a のイベント 4 つとアクティビティー b のイベント 2 つから成る、6 つのアクティビティー・イベントが含まれます。アクティビティー c は、報告期間が始まる前に完了したため、レポートで報告されるイベントとはなりません。



したがって、この期間内に完了したイベントの数を照会すると、結果は 2 になります。

レポート作成サイクル

期間レポートのレポート作成サイクルを定義することができます。このオプションを使用して、複数の期間を包含するレポートを作成します。例えば、直前の 12 カ月間に開始したプロセスの数を月ごとにレポート作成するとします。この場合は、月ごとに別々にレポート作成する必要はありません。代わりに、開始日を 1 月 1 日に、開始日以後のタイム・スライスの数を 12 に、タイム・スライスの長さを 1 カ月に定義することができます。結果のレポートには、タイム・スライス数を示す追加列が含まれています。各タイム・スライスの値は対象月を示します。

関連タスク

435 ページの『事前定義のリストおよび図表の使用』
Business Process Choreographer Explorer の事前定義のリストおよび図表では、ランタイム・エンティティーの状態およびイベント情報を入手するためにドリルダウン方式が採用されています。ドリルダウン処理の各ステップで、対象とする情報の種類を詳細に定義します。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。

437 ページの『事前定義の期間グラフの作成』
ある期間に指定の状態になったプロセス・インスタンスまたはアクティビティー・インスタンスの数の分布を表示させるには、Business Process Choreographer Explorer の事前定義の期間グラフを使用します。各インスタンスが、指定された状態になったタイム・スライスに表示されます。

439 ページの『例: 事前定義のグラフの使用』
このシナリオでは、Business Process Choreographer Explorer での事前定義のグラフの使用例を示します。

450 ページの『ユーザー定義期間レポートの作成』
Business Process Choreographer Explorer で、一定の期間に発生したプロセス・イベントまたはアクティビティー・イベントに関するユーザー定義レポートを作成できます。レポート作成サイクルに応じて、複数の期間にわたるレポートを作成することもできます。

時間処理

レポートで、Business Process Choreographer Explorer がタイム・スタンプと期間を処理する方法を検討します。

タイム・スタンプ

データベースでは、タイム・スタンプが協定世界時 (UTC) 形式で保管されます。入力および表示されるタイム・スタンプは、常にユーザー・インターフェースが実行されているロケーションの地方時です。これは、スナップショット・レポートにレポート作成サイクルを指定し、そのレポート作成サイクルが夏時間調整の時刻調整を含む場合、時刻変更後に日時が 1 時間ずれることを意味します。

例えばスナップショット・レポートに、冬時間の間は最初のスナップショットを午前 8:00 に取得し、次のスナップショットを 24 時間ごとに取得するレポート作成サイクルを指定すると、夏時間調整時刻の間は午前 9:00 にスナップショットが取得されます。

月および年の期間

レポートにレポート作成サイクルを指定するときに、例えばタイム・スライスを月または年単位の長さで設定する場合は、個々のタイム・スライスの長さがカレンダーに応じて変化します。これにより、それぞれのタイム・スライスが 1 年のいずれかの月を表すレポートを指定できます。

関連タスク

『事前定義のリストおよび図表の使用』
Business Process Choreographer Explorer の事前定義のリストおよび図表では、ランタイム・エンティティーの状態およびイベント情報を入手するためにドリルダウン方式が採用されています。ドリルダウン処理の各ステップで、対象とする情報の種類を詳細に定義します。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。

事前定義のリストおよび図表の使用

Business Process Choreographer Explorer の事前定義のリストおよび図表では、ランタイム・エンティティーの状態およびイベント情報を入手するためにドリルダウン方式が採用されています。ドリルダウン処理の各ステップで、対象とする情報の種類を詳細に定義します。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。

このタスクについて

事前定義のリストおよび図表としては、以下のものが使用可能です。

- リスト
- プロセスおよびアクティビティー・スナップショット・グラフ
- 期間別のプロセス・インスタンスおよびアクティビティー・インスタンスのグラフ

関連概念

434 ページの『時間処理』
レポートで、Business Process Choreographer Explorer がタイム・スタンプと期間を処理する方法を検討します。

430 ページの『スナップショット・レポート』
Business Process Choreographer Explorer のスナップショット・レポートを使用して、特定の日時におけるアクティビティーまたはプロセスの状態を判断します。

432 ページの『期間レポート』
Business Process Choreographer Explorer の期間レポートを使用して、一定期間に特定のアクティビティー・イベントまたはプロセス・イベントが発生する頻度を判別します。

関連タスク

438 ページの『例: 事前定義のリストの使用』
このシナリオでは、Business Process Choreographer Explorer での事前定義のリストの使用例を示します。

439 ページの『例: 事前定義のグラフの使用』
このシナリオでは、Business Process Choreographer Explorer での事前定義のグラフの使用例を示します。

事前定義リストを使用したレポートの作成

指定の期間内に発生したプロセス・イベントまたはアクティビティー・イベントの数についてのレポートを状態別に分類して作成するには、Business Process

Choreographer Explorer の事前定義のリストを使用します。リストを使用して、特定のインスタンスのイベントにドリルダウンすることもできます。また、各状態のレポート結果をエクスポートできます。

始める前に

「レポート」タブは、レポート作成が構成されている場合にのみ表示されます。レポート作成機能は、Business Process Choreographer Explorer の構成時に構成できますが、後で構成することもできます。

このタスクについて

Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「レポート」タブのナビゲーション・ペインでリストのタイプを選択します。

   事前定義のリストは、プロセス・インスタンス、アクティビティー・インスタンス、またはユーザーに関連したアクティビティーについて使用可能です。
2. 対象とする期間の開始日および終了日を入力し、「**継続**」をクリックします。

   リストのタイプに応じて、プロセス・テンプレートのリスト、アクティビティー・テンプレート、またはユーザーのリストと関連したインスタンスの数が表示されます。
3. 対象とするインスタンスのチェック・ボックスを選択し、「**インスタンス・スナップショット**」をクリックします。

   選択したインスタンスのイベントが、タブ付きのペインに表示されます。各ページには、特定の状態にあるインスタンスが表示されます。
4. オプション: 特定のインスタンスのすべてのイベントおよび詳細情報を参照するには、インスタンス名をクリックします。
5. オプション: レポートされたデータを CSV 形式でエクスポートするには、「**エクスポート**」をクリックします。生成されたエクスポート・データを開くか保存するかを選択し、「**OK**」をクリックします。 現在表示中の状態についてレポートするデータがエクスポートされます。

関連概念

434 ページの『時間処理』
レポートで、Business Process Choreographer Explorer がタイム・スタンプと期間を処理する方法を検討します。

事前定義スナップショット・グラフの作成

指定の日時におけるプロセス・インスタンスまたはアクティビティー・インスタンスの状態の分布を表示させるには、Business Process Choreographer Explorer の事前定義のスナップショット・グラフを使用します。

始める前に

「レポート」タブは、レポート作成が構成されている場合にのみ表示されます。レポート作成機能は、Business Process Choreographer Explorer の構成時に構成できますが、後で構成することもできます。

このタスクについて

Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「レポート」タブのナビゲーション・ペインの「**グラフ**」の下で、スナップショットのタイプを選択します。

   事前定義のスナップショット・グラフは、プロセス・インスタンスおよびアクティビティー・インスタンスについて使用可能です。
2. 検索基準を入力して、「**継続**」をクリックします。

   検索基準を満たすオブジェクト・テンプレートのリストが表示されます。
3. 対象とするテンプレートのチェック・ボックスを選択し、「**選択されたアクションを続行**」をクリックします。

   グラフのタイプを変更すると、棒グラフまたは円グラフとして結果を表示させることができます。

関連概念

430 ページの『スナップショット・レポート』
Business Process Choreographer Explorer のスナップショット・レポートを使用して、特定の日時におけるアクティビティーまたはプロセスの状態を判断します。

434 ページの『時間処理』
レポートで、Business Process Choreographer Explorer がタイム・スタンプと期間を処理する方法を検討します。

関連タスク

439 ページの『例: 事前定義のグラフの使用』
このシナリオでは、Business Process Choreographer Explorer での事前定義のグラフの使用例を示します。

事前定義の期間グラフの作成

ある期間に指定の状態になったプロセス・インスタンスまたはアクティビティー・インスタンスの数の分布を表示させるには、Business Process Choreographer Explorer の事前定義の期間グラフを使用します。各インスタンスが、指定された状態になったタイム・スライスに表示されます。

始める前に

「レポート」タブは、レポート作成が構成されている場合にのみ表示されます。レポート作成機能は、Business Process Choreographer Explorer の構成時に構成できますが、後で構成することもできます。

このタスクについて

事前定義の期間グラフの使用例としては、事前定義のグラフを使用して、直近 12 カ月に完了したプロセス・インスタンスの分布を表示するような場合があります。これを行うには、Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「レポート」タブのナビゲーション・ペインの「**グラフ**」の下で、期間グラフのタイプを選択します。

   事前定義の期間グラフは、プロセス・インスタンスおよびアクティビティー・インスタンスについて使用可能です。
2. 検索基準を入力して、「**継続**」をクリックします。

   期間の開始日を入力して、タイム・スライスの数、各タイム・スライスの長さ、およびレポートさせる状態を指定します。例えば、直近 12 カ月の各月に完了したインスタンスについてレポートさせるには、タイム・スライスの数として 12 を指定し、各タイム・スライスの長さとして 1 カ月を指定します。

   検索基準を満たすオブジェクト・テンプレートのリストが表示されます。
3. 対象とするテンプレートのチェック・ボックスを選択し、「**選択されたアクションを続行**」をクリックします。

   グラフのタイプを変更すると、棒グラフ、線グラフ、または円グラフとして結果を表示させることができます。

関連概念

432 ページの『期間レポート』
Business Process Choreographer Explorer の期間レポートを使用して、一定期間に特定のアクティビティー・イベントまたはプロセス・イベントが発生する頻度を判別します。

434 ページの『時間処理』
レポートで、Business Process Choreographer Explorer がタイム・スタンプと期間を処理する方法を検討します。

関連タスク

439 ページの『例: 事前定義のグラフの使用』
このシナリオでは、Business Process Choreographer Explorer での事前定義のグラフの使用例を示します。

例: 事前定義のリストの使用

このシナリオでは、Business Process Choreographer Explorer での事前定義のリストの使用例を示します。

このタスクについて

工場では品目 Item1、Item2、および Item3 を生産します。製造プロセスおよび出荷プロセスは、WebSphere Process Server で SOA プロセスとしてモデル化され、実行されます。各顧客オーダーは、該当するプロセス・テンプレートの専用のプロセス・インスタンスによって表されます。製品を顧客に出荷すると、出荷プロセスは

最終状態である「終了」に到達します。顧客がオーダーを取り消した場合は、対応するプロセス・インスタンスが終了し、強制終了された状態になります。

先月 Item1、Item2、または Item3 のオーダーを取り消した顧客の数を調べるために、強制終了状態になったプロセス・インスタンスの数を知りたいとします。さらに、取り消されたときにオーダー処理がどの段階まで進行していたかも知りたいとします。

取り消されたプロセスの数を表示するビューを作成し、取り消し時のプロセスの状態を表示させるには、事前定義のリストを使用します。これを行うには、Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「レポート」タブのナビゲーション・ペインの「**リスト**」の下で、「**プロセス**」を選択します。
2. 「検索基準」ページで、対象期間の開始日および終了日を入力し、「**継続**」をクリックします。 「プロセス・テンプレート」ページに、観察期間内にプロセスを生成したすべてのプロセス・テンプレートがリストされます。各プロセス・テンプレートについて、開始および終了したプロセス・インスタンスの数を表示させることができます。
3. 「プロセス・テンプレート」ページで、リストにあるすべてのテンプレートを選択し、「**インスタンス・スナップショット (Instance snapshot)**」をクリックします。 観察期間内に到達した状態別にプロセス・インスタンスがグループ化され、そのすべてが「プロセス・インスタンス」ページにリストされます。
4. 「プロセス・インスタンス」ページで「**強制終了**」タブを選択すると、観察期間内の取り消しの総数が表示されます。
5. リストをテンプレート名でソートして、プロセス・テンプレートごとの取り消し回数を評価します。
6. 詳細を表示させるには、強制終了されたプロセス・インスタンスの名前をクリックして「プロセス・インスタンスの詳細」ページを表示させます。 インスタンスの処理時刻および経過時間を確認します。

関連概念

434 ページの『時間処理』
レポートで、Business Process Choreographer Explorer がタイム・スタンプと期間を処理する方法を検討します。

関連タスク

435 ページの『事前定義のリストおよび図表の使用』
Business Process Choreographer Explorer の事前定義のリストおよび図表では、ランタイム・エンティティーの状態およびイベント情報を入手するためにドリルダウン方式が採用されています。ドリルダウン処理の各ステップで、対象とする情報の種類を詳細に定義します。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。

例: 事前定義のグラフの使用

このシナリオでは、Business Process Choreographer Explorer での事前定義のグラフの使用例を示します。

このタスクについて

工場では品目 Item1 および Item2 を生産します。製造プロセスおよび出荷プロセスは、WebSphere Process Server で SOA プロセスとしてモデル化され、実行されます。各顧客オーダーは、該当するプロセス・テンプレートの専用のプロセス・インスタンスによって表されます。

最近、Item3 によって生産ラインを拡張しました。新しい Item3 オーダー・テンプレートを持っており、先月の生産ラインでの進捗を知りたいと考えています。指標として、直近 30 日以内の生産オーダーの数を知りたいとします。

直近 30 日以内に処理された生産オーダーの数を視覚化するために、対象期間における OrderItem3 プロセス・テンプレートに関連したすべてのプロセス・インスタンスを表示するグラフ・ビューを指定します。これを行うには、Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「レポート」タブのナビゲーション・ペインの「**グラフ**」の下で、「**期間によるプロセス**」を選択して、直近 30 日以内のプロセス・インスタンスの統計的分布を表示させます。
2. 検索基準を指定します。
   a. 観察期間の開始日を入力します。
   b. タイム・スライスの数を 30 に設定します。
   c. タイム・スライスの長さを 1 日に設定します。
   d. 「**次の状態にフォーカス**」リストで「**実行中**」を選択し、「**継続**」をクリックします。

   「プロセス・テンプレートの選択」ページが開き、観察期間内に生成されたプロセス・インスタンスに関連するすべてのプロセス・テンプレートのリストが表示されます。
3. OrderItem3 テンプレートを選択し、そのプロセス・テンプレートに関連したすべてのプロセス・インスタンスを表示させ、「**選択されたアクションを続行**」をクリックします。
4. 「プロセス・インスタンス・スナップショット」ページに、指定の時間にそれぞれの状態にあったすべてのプロセス・インスタンスが表示されます。
5. 先月のプロセスの進捗を視覚化するには、折れ線グラフまたは棒グラフを使用します。

次のタスク

観察期間内に「実行中」の状態になったすべてのプロセス・インスタンスがレポートに表示されます。

関連概念

434 ページの『時間処理』
レポートで、Business Process Choreographer Explorer がタイム・スタンプと期間を処理する方法を検討します。

430 ページの『スナップショット・レポート』
Business Process Choreographer Explorer のスナップショット・レポートを使用して、特定の日時におけるアクティビティーまたはプロセスの状態を判断します。

432 ページの『期間レポート』
Business Process Choreographer Explorer の期間レポートを使用して、一定期間に特定のアクティビティー・イベントまたはプロセス・イベントが発生する頻度を判別します。

関連タスク

435 ページの『事前定義のリストおよび図表の使用』
Business Process Choreographer Explorer の事前定義のリストおよび図表では、ランタイム・エンティティーの状態およびイベント情報を入手するためにドリルダウン方式が採用されています。ドリルダウン処理の各ステップで、対象とする情報の種類を詳細に定義します。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。

436 ページの『事前定義スナップショット・グラフの作成』
指定の日時におけるプロセス・インスタンスまたはアクティビティー・インスタンスの状態の分布を表示させるには、Business Process Choreographer Explorer の事前定義のスナップショット・グラフを使用します。

437 ページの『事前定義の期間グラフの作成』
ある期間に指定の状態になったプロセス・インスタンスまたはアクティビティー・インスタンスの数の分布を表示させるには、Business Process Choreographer Explorer の事前定義の期間グラフを使用します。各インスタンスが、指定された状態になったタイム・スライスに表示されます。

ユーザー定義レポートの作成

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、Business Process Choreographer Explorer を使用してレポート定義を格納して再利用したり、レポート結果をエクスポートしたりすることができます。

このタスクについて

プロセス・レポートでは、プロセス・インスタンスの属性と、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、アクティビティーの属性と、アクティビティーが関連付けられているプロセス・インスタンスの情報を取得できます。1 回限りの照会を定義するほか、レポート定義を保存して必要なときに実行できるようにすることができます。レポートを実行するたびにレポート定義の値を変更するには、パラメーターを指定します。

関連概念

434 ページの『時間処理』
レポートで、Business Process Choreographer Explorer がタイム・スタンプと期間を処理する方法を検討します。

430 ページの『スナップショット・レポート』
Business Process Choreographer Explorer のスナップショット・レポートを使用して、特定の日時におけるアクティビティーまたはプロセスの状態を判断します。

432 ページの『期間レポート』
Business Process Choreographer Explorer の期間レポートを使用して、一定期間に特定のアクティビティー・イベントまたはプロセス・イベントが発生する頻度を判別します。

関連タスク

453 ページの『例: ユーザー定義レポートの使用』
このシナリオでは、Business Process Choreographer Explorer を使用したユーザー定義レポートの使用例を示します。

Business Process Choreographer Explorer レポートの属性

属性を使用して、Business Process Choreographer Explorer にレポートの内容を定義し、結果をフィルターに掛けます。使用可能な属性は、レポート・タイプに応じて決まります。

レポート内容として定義される各属性は、レポート内の列の名前です。また、属性を使用して照会の結果をフィルターに掛けます。レポートに含めていなかった属性のフィルター基準を定義することもできます。

属性	説明	スナップショット・レポート	期間レポート
アクティビティーが完了した	アクティビティー・インスタンスが、失敗、完了、スキップ、強制終了、または期限切れのいずれかの終了状態になったときの時刻。	X	X
アクティビティー・イベント	アクティビティー・イベントのイベント・コード。	X	X
アクティビティー・イベント・カウント	アクティビティー・インスタンスによって発行されたアクティビティー・イベントの数。	X	X
アクティビティー・インスタンス ID	アクティビティー・インスタンス ID。	X	X
アクティビティーの種類	アクティビティー・インスタンスの種類。	X	X
アクティビティー最終ユーザー名	このアクティビティーでアクションを開始した最後のユーザーの名前。	X	X
アクティビティー名	アクティビティー・インスタンスの名前。	X	X
アクティビティーが開始した	アクティビティー・インスタンスが開始された時刻。	X	X
アクティビティー状態	イベント後のアクティビティー・インスタンスの状態。	X	X

属性	説明	スナップショット・レポート	期間レポート
アクティビティー・テンプレート ID	アクティビティー・テンプレート ID。	X	X
アクティビティーの平均継続時間	アクティビティー・インスタンスすべての平均継続時間 (秒)。	X	X
プロセスの平均継続時間	プロセス・インスタンスすべての平均継続時間 (秒)。	X	X
イベント時間	イベントが発生した時刻。	X	X
例外テキスト	例外によってアクティビティー・イベントがトリガーされた場合、例外メッセージはイベント・データの一部になる可能性があり、それからこのフィールドに保管されます。	X	X
指定した状態のアクティビティーの数	指定された状態になっているアクティビティー・インスタンスの数。	X	
アクティビティー・イベントの数	指定された期間内に発生したアクティビティー・イベントの数。		X
プロセス・イベントの数	指定された期間内に発生したプロセス・イベントの数。		X
指定した状態のプロセスの数	指定された状態になっているプロセス・インスタンスの数。	X	
プロセス・アクティビティー・カウント	最低でも 1 つのイベントを発行したプロセス・インスタンスのアクティビティーの数。	X	X
プロセス・アクティビティー・イベント・カウント	プロセス・インスタンスに属するアクティビティー・イベントの数。	X	X
プロセスが完了した	プロセス・インスタンスが、補正、補正の失敗、失敗、完了、または強制終了のいずれかの終了状態になったときの時刻。	X	X
プロセス削除時刻	プロセスが Business Process Choreographer データベースから削除された時刻。	X	X
プロセス・イベント	プロセス・インスタンス・イベントのイベント・コード。	X	X
プロセス・イベント・カウント	プロセス・インスタンスによって発行されたプロセス・イベントの数。	X	X
プロセス・インスタンス ID	プロセス・インスタンス ID。	X	X
プロセス最終ユーザー名	このプロセスでアクションを開始した最後のユーザーの名前。	X	X
プロセスが開始された	プロセス・インスタンスが開始された時刻。	X	X
プロセス状態	イベント後のプロセス・インスタンスの状態。	X	X
プロセス・テンプレート ID	プロセス・テンプレート ID。	X	X

属性	説明	スナップショット・レポート	期間レポート
プロセス・テンプレート名	プロセス・インスタンスに関連付けられているプロセス・テンプレート。	X	X
プロセス作業時間	プロセス・インスタンスの所要時間。この値は、プロセスに含まれている完了した基本アクティビティーすべての作業時間の合計です。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。	X	X
スナップショット番号	レポート作成サイクルが定義されているスナップショット・レポートでは、この属性によってレポート作成サイクルの特定のスナップショットが識別されます。	X	
タイム・スライス番号	レポート作成サイクルが定義されている期間レポートでは、この属性によってレポート作成サイクルの特定のタイム・スライスが識別されます。		X
ユーザー名	イベントに関連付けられているユーザーのユーザー ID。	X	X
有効開始日	プロセス・テンプレートが有効になる時刻。	X	X

Business Process Choreographer Explorer のビジネス・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。これらのイベントのサブセットは、Business Process Choreographer Explorer で使用可能です。

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。

- 『プロセス・イベント』
- 445 ページの『アクティビティー・イベント』

Business Process Choreographer Explorer には、イベントで送信されたビジネス・データは表示されません。

プロセス・イベント

以下の表に、Business Process Choreographer Explorer を使用して報告できるすべてのプロセス・イベントを示します。

コード	説明
21000	プロセスが開始された
21001	プロセスが中断された
21002	プロセスが再開された
21004	プロセスが完了した

コード	説明
21005	プロセスが強制終了された
21019	プロセスが再始動した
42001	プロセスが失敗した
42003	プロセスが補正中
42004	プロセスが補正された
42009	プロセスが強制終了中
42010	プロセスが失敗する
42046	プロセス補正が失敗した
42079	プロセスがマイグレーションされた

さらに、21020 イベント・コード (プロセス削除時刻) は、Business Process Choreographer Explorer レポート作成機能でプロセス・インスタンスの「プロセス削除 (process deleted)」属性を更新するために処理されます。このイベントは、リストしたプロセス・イベントと同じように照会することはできません。

アクティビティー・イベント

以下の表に、Business Process Choreographer Explorer を使用して報告できるすべてのアクティビティー・イベントを示します。

コード	説明
21006	アクティビティーが作動可能
21007	アクティビティーが開始した
21011	アクティビティーが完了した
21021	アクティビティー要求が取り消された
21022	アクティビティーが要求された
21027	アクティビティーが強制終了された
21080	アクティビティーが失敗した
21081	アクティビティーの期限切れ
42005	アクティビティーがスキップされた
42015	アクティビティーが停止した
42031	アクティビティーが強制再試行された
42032	アクティビティーが強制完了した
42036	アクティビティーがメッセージを受信
42063	アクティビティーがジャンプした
42065	アクティビティーが要求に応じてスキップされた
42070	アクティビティーが出口条件をスキップした

パフォーマンス関連の属性

Business Process Choreographer Explorer を使用してレポート定義を実行するために必要な時間はさまざまです。レポート生成のパフォーマンスを向上させるには、レポート定義を最適化することができます。いくつかの一般的な規則に基づいて、レポートの属性がパフォーマンスに与える影響を評価できます。

フィルターを指定する

適切なフィルターを使用して検索されるデータの量を制限します。日付、またはアクティビティー・インスタンスやプロセス・インスタンスのその他のプロパティーによってレポート結果を制限することを検討します。スナップショット・レポートの場合は、ReportAtSnapshotRange 構成パラメーターを適切な値に設定します。

期間レポートとスナップショット・レポート

スナップショット・レポートは、期間レポートよりパフォーマンスが低下する傾向があります。

レポート作成サイクルが定義されているレポート

レポート作成サイクルが定義されているレポートは、パフォーマンスが低下する傾向があり、特に、多くの期間またはスナップショットが照会に定義されている場合には顕著になります。

集約　イベントの合計数などの集約、またはインスタンスの平均継続時間では、大量のデータを処理する必要が生じることがあり、そのためにパフォーマンスが低下します。

表示される結果の数

レポートの一部の結果にのみ関心がある場合は、結果に出力されるエントリーの数を制限するしきい値を指定できます。これにより、データベースとユーザー・インターフェースの間で転送されるデータ量が減少します。

ただし、ソート順を定義する場合は、データをソートできるようにするため、まずすべての結果データをデータベースに収集する必要があります。この場合、表示される結果の数を減らしてもパフォーマンスは向上しません。代わりに、適切なフィルター式を設定してください。

イベントおよびインスタンスの情報

レポート・データベースでは、イベントに関連する情報はイベント・データベース表に保管されるのに対し、アクティビティーとプロセスのインスタンスに関連する情報はインスタンス・データベース表に保管されます。インスタンス関連情報とイベント固有情報の両方を含むレポートを作成する場合は、必要な情報を取得するためにそれらの表が結合されます。1 つのタイプの情報のみを含むレポートを作成する場合、表は結合されません。したがって、1 つのタイプの情報のみを含むレポートのパフォーマンスは通常、インスタンス関連とイベント固有の両方の情報を照会するレポートのパフォーマンスよりも優れています。

ユーザー定義スナップショット・レポートの作成

指定の日時に状態情報のスナップショットを取得するユーザー定義レポートを Business Process Choreographer Explorer で定義できます。レポート作成期間内の定期的な (各月の初日の午前 0 時などの) 状態スナップショットを含むレポートを作成することもできます。

始める前に

「レポート」タブは、レポート作成が構成されている場合にのみ表示されます。レポート作成機能は、Business Process Choreographer Explorer の構成時に構成できますが、後で構成することもできます。

このタスクについて

Business Process Choreographer Explorer で次のステップを実行します。レポート・ウィザードに、レポートを定義するためのガイドが表示されます。

手順

1. 「レポート」タブのナビゲーション・ペインで、「**新規レポート**」アイコン () をプロセス・レポートまたはアクティビティー・レポートのいずれかについてクリックします。
2. 「レポート・タイプの選択」ページで、「**スナップショット・レポート**」をクリックして「**次へ**」をクリックします。
3. 「スナップショット・タイプの選択」ページで、いつスナップショットを取得するかを指定して「**次へ**」をクリックします。
   - 現在の状況を表示させるには、「**現在のスナップショットを取得**」をクリックします。スナップショット日時は、レポートを実行するごとに評価されます。

     「内容の指定 (Specify Content)」ページが表示されます。ステップ 5 に進みます。
   - 特定の日時 (6 月 10 日の午前 8:00 など) におけるプロセスまたはアクティビティーの状況を表示させるには、「**特定の日時にスナップショットを取得**」をクリックします。

     「スナップショット設定の指定」ページが表示されます。ステップ 4 に進みます。
   - レポート作成期間内の定期的な状況を表示させるには、「**レポート作成のサイクルに従って繰り返しスナップショットを取得**」をクリックします。

     「スナップショット設定の指定」ページが表示されます。ステップ 4 に進みます。
4. スナップショットの設定を指定して、「**次へ**」をクリックします。

   特定の日時にスナップショットを取得する場合は、日時の設定を指定します。将来の日時を指定できます。レポートを実行するときに毎回設定を変更するには、「**これらの設定をパラメーターとして使用**」 チェック・ボックスを選択します。

   レポートにレポート作成サイクルを指定する場合:
   a. レポート作成サイクルの開始日と終了日のいずれを設定するかを選択し、「**次へ**」をクリックします。
   b. レポート作成サイクルの開始日を設定するには、最初のスナップショットをいつ取得するかを指定します。レポート作成サイクルの終了日を設定するには、最後のスナップショットをいつ取得するかを指定します。
   c. レポート作成サイクルの期間を定義するには、スナップショットの数と、スナップショットの取得間隔を設定します。
   d. レポートを実行するときに毎回レポート作成サイクルの設定を変更するには、「**これらの設定をパラメーターとして使用**」チェック・ボックスを選択します。

5. 「レポート内容の指定」ページで、レポートに含める情報を指定し、「**次へ**」をクリックします。

   レポートにレポート作成サイクルが指定されている場合は、属性のリストに既にスナップショット数属性が含まれています。この属性を削除することはできません。

   a. 「**追加**」をクリックし、レポートに組み込める属性のリストを表示させます。これらの属性はレポートの列見出しになります。 属性の位置によってレポートでの列の順序が決まります。各属性には、列内での結果のソート方法を指定できます。複数の属性にソート順を指定した場合は、属性の順序に従って結果がソートされます。レポートでの結果のソート順を変更するには、属性を並べ替えてください。

      - 属性を変更するには、「**編集**」アイコン () をクリックします。
      - 属性を削除するには、「**削除**」アイコン () をクリックします。
      - レポート内での属性の位置を変更するには、「**上へ**」アイコン () または「**下へ**」アイコン () をクリックします。

   b. パフォーマンス上の理由などで結果に含める項目の数を制限するには、結果の最大数を指定する値を「**しきい値**」フィールドに入力します。

      デフォルトのしきい値は `20` です。結果を制限しない場合は、値を `-1` に設定します。

      レポートを実行するときに毎回しきい値を変更するには、「**しきい値をパラメーターとして使用**」チェック・ボックスを選択します。

6. オプション: 「内容のフィルタリングの指定」ページで、属性のフィルター基準を設定します。

   属性がとることができる値を制限してレポートを特定の目的に限定するには、フィルター基準を使用します。レポートには、指定されたすべてのフィルター基準を満たすプロセスおよびアクティビティーのみが入れられます。集約である「レポート内容の指定」ページで属性を指定した場合は、フィルター基準のリストに、この属性のフィルター基準が既に含まれています。このフィルターを削除することはできません。

   a. 「**追加**」をクリックし、フィルター基準を指定できる属性のリストを表示させます。
      - 複雑な値のタイプの場合は (タイム・スタンプなど)、「**入力ヘルパー (Input Helper)**」アイコン () をクリックしてフィールドに入力します。
      - レポートを実行するときに毎回フィルター基準の値を変更するには、「**パラメーター**」チェック・ボックスを選択します。

   b. 「**次へ**」をクリックします。

   「要約」ページが表示されます。このページには、レポート定義が表示されます。

7. 「要約」ページで、以下のいずれかを実行します。
   - レポート定義がパラメーターを含まない場合は、「**実行**」をクリックします。

     生成されたレポートが表示されます。
   - レポート定義がパラメーターを含む場合は、「**次へ**」をクリックします。

     パラメーターの値は変更できます。「**実行**」をクリックします。生成されたレポートが表示されます。

   レポート結果が予想と異なる場合は、「**編集**」をクリックしてレポートの設定を変更できます。
8. オプション: レポートの結果をエクスポートします。

   レポートされたデータを CSV 形式でエクスポートするには、「**エクスポート**」をクリックします。生成されたエクスポート・データを開くかデータをハード・ディスクに保存するかを選択し、「**OK**」をクリックします。

   レポート・リストに項目が含まれる場合にのみ「**エクスポート**」ボタンが表示されます。
9. オプション: レポート定義を保管します。

   このレポートを 2 回以上実行する場合は (毎月 10 日に完了したプロセス・インスタンスを表示する月次レポートなど)、「**保管**」をクリックしてレポート名を入力します。 ナビゲーション・ペインにレポートが表示されます。

関連概念

430 ページの『スナップショット・レポート』
Business Process Choreographer Explorer のスナップショット・レポートを使用して、特定の日時におけるアクティビティーまたはプロセスの状態を判断します。

434 ページの『時間処理』
レポートで、Business Process Choreographer Explorer がタイム・スタンプと期間を処理する方法を検討します。

関連タスク

453 ページの『例: ユーザー定義レポートの使用』
このシナリオでは、Business Process Choreographer Explorer を使用したユーザー定義レポートの使用例を示します。

関連資料

442 ページの『Business Process Choreographer Explorer レポートの属性』
属性を使用して、Business Process Choreographer Explorer にレポートの内容を定義し、結果をフィルターに掛けます。使用可能な属性は、レポート・タイプに応じて決まります。

444 ページの『Business Process Choreographer Explorer のビジネス・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。これらのイベントのサブセットは、Business Process Choreographer Explorer で使用可能です。

445 ページの『パフォーマンス関連の属性』
Business Process Choreographer Explorer を使用してレポート定義を実行するために必要な時間はさまざまです。レポート生成のパフォーマンスを向上させるには、レポート定義を最適化することができます。いくつかの一般的な規則に基づいて、レポートの属性がパフォーマンスに与える影響を評価できます。

ユーザー定義期間レポートの作成

Business Process Choreographer Explorer で、一定の期間に発生したプロセス・イベントまたはアクティビティー・イベントに関するユーザー定義レポートを作成できます。レポート作成サイクルに応じて、複数の期間にわたるレポートを作成することもできます。

始める前に

「レポート」タブは、レポート作成が構成されている場合にのみ表示されます。レポート作成機能は、Business Process Choreographer Explorer の構成時に構成できますが、後で構成することもできます。

このタスクについて

Business Process Choreographer Explorer で次のステップを実行します。レポート・ウィザードに、レポートを定義するためのガイドが表示されます。

手順

1. 「レポート」タブのナビゲーション・ペインで、「**新規レポート**」アイコン（  ）をプロセス・レポートまたはアクティビティー・レポートのいずれかについてクリックします。
2. 「レポート・タイプの選択」ページで「**期間レポート**」をクリックし、「**次へ**」をクリックします。
3. 「期間タイプの選択」ページで期間のタイプを指定し、「**次へ**」をクリックします。

   例えばプロセスの場合は、以下の期間タイプのいずれかを選択できます。
   - 指定の日付から現在までのイベントを表示させるには、「**現在までのすべてのプロセスについてレポート**」をクリックします。
   - 指定した期間のイベントを表示させるには、「**指定された期間内のプロセスについてレポート**」をクリックします。
   - レポート作成期間内のイベントを一定の間隔で表示させるには、「**レポート作成のサイクルに従ってプロセスについてレポート**」をクリックします。

   「日時の指定」ページが表示されます。
4. 日時設定を指定し、「**次へ**」をクリックします。

   現在までのすべてのプロセスに関するレポートの場合は、開始日を指定します。終了日はレポートを実行するたびに生成されます。指定した期間のプロセスに関するレポートの場合は、開始日および終了日を指定します。日付には将来の日付を指定できます。レポートを実行するときに毎回設定を変更するには、「**これらの設定をパラメーターとして使用**」 チェック・ボックスを選択します。

   レポートにレポート作成サイクルを指定する場合:
   a. レポート作成サイクルの開始日と終了日のいずれを設定するかを選択し、「**次へ**」をクリックします。
   b. レポート作成サイクルの開始日を設定するには、最初のタイム・スライスの開始日を指定します。レポート作成サイクルの終了日を設定するには、最後のタイム・スライスの終了日を指定します。
   c. レポート作成サイクルの期間を定義するには、タイム・スライスの総数と、各タイム・スライスの長さを設定します。
   d. レポートを実行するときに毎回レポート作成サイクルの設定を変更するには、「**これらの設定をパラメーターとして使用**」チェック・ボックスを選択します。
5. 「レポート内容の指定」ページで、レポートに含める情報を指定し、「**次へ**」をクリックします。

   レポートにレポート作成サイクルが指定されている場合は、属性のリストに既にタイム・スライス数属性が含まれています。この属性を削除することはできません。
   a. 「**追加**」をクリックし、レポートに組み込める属性のリストを表示させます。これらの属性はレポートの列見出しになります。 属性の位置によってレポートでの列の順序が決まります。各属性には、列内での結果のソート方法

を指定できます。複数の属性にソート順を指定した場合は、属性の順序に従って結果がソートされます。レポートでの結果のソート順を変更するには、属性を並べ替えてください。

- 属性を変更するには、「**編集**」アイコン () をクリックします。
- 属性を削除するには、「**削除**」アイコン () をクリックします。
- レポート内での属性の位置を変更するには、「**上へ**」アイコン () または「**下へ**」アイコン () をクリックします。

b. パフォーマンス上の理由などで結果に含める項目の数を制限するには、結果の最大数を指定する値を「**しきい値**」フィールドに入力します。

デフォルトのしきい値は `20` です。結果を制限しない場合は、値を `-1` に設定します。

レポートを実行するときに毎回しきい値を変更するには、「**しきい値をパラメーターとして使用**」チェック・ボックスを選択します。

6. オプション: 「内容のフィルタリングの指定」ページで、属性のフィルター基準を設定します。

属性がとることができる値を制限してレポートを特定の目的に限定するには、フィルター基準を使用します。集約である「レポート内容の指定」ページで属性を指定した場合は、フィルター基準のリストに、この属性のフィルター基準が既に含まれています。このフィルターを削除することはできません。

a. 「**追加**」をクリックし、フィルター基準を指定できる属性のリストを表示させます。
   - 複雑な値のタイプの場合は (タイム・スタンプなど)、「**入力ヘルパー (Input Helper)**」アイコン () をクリックしてフィールドに入力します。
   - レポートを実行するときに毎回フィルター基準の値を変更するには、「**パラメーター**」チェック・ボックスを選択します。

b. 「**次へ**」をクリックします。

「要約」ページが表示されます。このページには、レポート定義が表示されます。

7. 「要約」ページで、以下のいずれかを実行します。
   - レポート定義がパラメーターを含まない場合は、「**実行**」をクリックします。

     生成されたレポートが表示されます。
   - レポート定義がパラメーターを含む場合は、「**次へ**」をクリックします。

     パラメーターの値は変更できます。「**実行**」をクリックします。生成されたレポートが表示されます。

レポート結果が予想と異なる場合は、「**編集**」をクリックしてレポートの設定を変更できます。

8. オプション: レポートの結果をエクスポートします。

   レポートされたデータを CSV 形式でエクスポートするには、「**エクスポート**」をクリックします。生成されたエクスポート・データを開くかデータをハード・ディスクに保存するかを選択し、「**OK**」をクリックします。

   レポート・リストに項目が含まれる場合にのみ「**エクスポート**」ボタンが表示されます。
9. オプション: レポート定義を保管します。

   このレポートを定期的に実行する場合 (月次レポートなど)、「**保管**」をクリックしてレポート名を入力します。 ナビゲーション・ペインにレポートが表示されます。

関連概念

432 ページの『期間レポート』
Business Process Choreographer Explorer の期間レポートを使用して、一定期間に特定のアクティビティー・イベントまたはプロセス・イベントが発生する頻度を判別します。

434 ページの『時間処理』
レポートで、Business Process Choreographer Explorer がタイム・スタンプと期間を処理する方法を検討します。

関連タスク

『例: ユーザー定義レポートの使用』
このシナリオでは、Business Process Choreographer Explorer を使用したユーザー定義レポートの使用例を示します。

関連資料

442 ページの『Business Process Choreographer Explorer レポートの属性』
属性を使用して、Business Process Choreographer Explorer にレポートの内容を定義し、結果をフィルターに掛けます。使用可能な属性は、レポート・タイプに応じて決まります。

444 ページの『Business Process Choreographer Explorer のビジネス・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。これらのイベントのサブセットは、Business Process Choreographer Explorer で使用可能です。

445 ページの『パフォーマンス関連の属性』
Business Process Choreographer Explorer を使用してレポート定義を実行するために必要な時間はさまざまです。レポート生成のパフォーマンスを向上させるには、レポート定義を最適化することができます。いくつかの一般的な規則に基づいて、レポートの属性がパフォーマンスに与える影響を評価できます。

例: ユーザー定義レポートの使用

このシナリオでは、Business Process Choreographer Explorer を使用したユーザー定義レポートの使用例を示します。

このタスクについて

工場では品目 Item1、Item2、および Item3 を生産します。製造プロセスおよび出荷プロセスは、WebSphere Process Server で SOA プロセスとしてモデル化され、実行されます。各顧客オーダーは、該当するプロセス・テンプレートの専用のプロセス・インスタンスによって表されます。製品を顧客に出荷すると、出荷プロセスは最終状態である「終了」に到達します。顧客がオーダーを取り消した場合は、対応するプロセス・インスタンスが終了し、強制終了された状態になります。

オーダーを取り消した顧客の 1 人が、応答時間が長すぎたと訴えています。このオーダーの処理にそれほど時間がかかった理由を知りたいとします。

「強制終了」の状態にあり、かつ作業時間が 2 日を超えているプロセス・インスタンス用のユーザー定義レポートを作成します。さらに、強制終了されたプロセス・インスタンスでの問題もレポートで明らかにするものとします。これを行うには、Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 顧客のオーダーに属するプロセス・インスタンス・データを取得します。

   顧客名、アドレス、およびオーダー番号はビジネス・データの一部であるため、プロセス・メッセージに含まれます。しかし、ビジネス・オブジェクトは Common Event Infrastructure (CEI) イベントの一部ではないため、Business Process Choreographer Explorer ではビジネス・オブジェクトの内容を使用できません。しかし、「強制終了」の状態にあり、かつ作業時間が 2 日を超えているプロセス・インスタンスを探していることは理解しています。

   a. 「レポート」タブのナビゲーション・ペインの「**プロセス・レポート**」の下で、「**新規レポートの作成**」を選択します。
   b. プロセス・インスタンスの状態に注目しているため、レポート・タイプとして「**スナップショット・レポート**」を選択します。
   c. 「スナップショット・タイプの選択」ページで、「**特定の日時にスナップショットを取得**」を選択します。オーダーが取り消された直後の日時を、スナップショット日付の条件として指定します。
   d. 「レポート内容の指定」のページで、「**プロセス・インスタンス ID**」、「**プロセス作業時間**」、「**開始したプロセス**」、および「**完了したプロセス**」をレポートの内容に追加します。
   e. 「内容のフィルタリングの指定」ページで、フィルターの内容として「**プロセスの作業時間が 2 日を超える (Process work time greater 2 days)**」および「**プロセス状態が強制終了に等しい (Process state equal Terminated)**」を指定し、レポートを実行します。
   f. 「レポートの要約」ページで、プロセス・インスタンス ID、開始日、および完了日を確認し、顧客のオーダーに対応するプロセス・インスタンスを探します。 レポートの結果が予想と異なる場合 (プロセス・インスタンスのリストが長すぎる場合など) は、「**戻る**」をクリックして検索基準を変更します。
   g. ステップ 2 ではプロセス・インスタンス ID が必要になるため、この ID をクリップボードにコピーします。

2. 特定のプロセス・インスタンスでの問題を明らかにする情報を入手します。

a. ナビゲーション・ペインの「**プロセス・レポート**」セクションで、「**新規レポートの作成**」を選択します。
b. レポート・タイプとして「**スナップショット・レポート**」を選択します。

   「期間レポート」のタイプは使用しないでください。ここではスナップショット・レポートに関連した属性を対象としています。違いを確認するには、まったく同じ属性で期間レポートを定義して実行してみてください。
c. 「スナップショット・タイプの選択」ページで、「**特定の日時にスナップショットを取得**」を選択します。オーダーが取り消された直後の日時を、スナップショット日付の条件として指定します。
d. 「レポート内容の指定」のページで、「**プロセス・インスタンス ID**」、「**アクティビティー名**」、「**開始したアクティビティー**」、および「**完了したアクティビティー**」をレポートの内容に追加します。
e. 「内容のフィルタリングの指定」ページで、フィルターの内容として「**プロセス・インスタンス ID が *your_customer's_process_instance_ID* に等しい (Process instance ID equal *your_customer's_process_instance_ID*)**」を指定し、レポートを実行します。 ほとんどの時間が費やされているアクティビティーがレポートで明らかにされます。
f. オプション: 遅れの根本原因を調べるために詳細な情報が必要な場合は、レポートを編集して再実行します。
g. レポート定義を保管します。

3. 以上を終えて、今後は同様の状況を避けることにします。 毎営業日の終わりにレポートを作成して、リソースの制約や障害などが原因で制限時間を超える危険性があるアクティブなオーダー・プロセスをすべてリストさせます。
   a. 保管したレポート定義を編集します。「スナップショット・タイプの選択」ページで、スナップショット・タイプを「**現在のスナップショットを取得**」に変更し、フィルターの内容「**プロセス・インスタンス ID が *your_customer's_process_instance_ID* に等しい (Process instance ID equal *your_customer's_process_instance_ID*)**」および式「**プロセスの作業時間が 1 日を超える (Process work time greater 1 day)**」を削除します。
   b. 変更したレポートを実行して、新しいフィルター基準を満たすプロセス・インスタンスがないことを確認します。
   c. レポートを保存して、毎営業日の終わりに実行できるようにします。

関連概念

434 ページの『時間処理』
レポートで、Business Process Choreographer Explorer がタイム・スタンプと期間を処理する方法を検討します。

430 ページの『スナップショット・レポート』
Business Process Choreographer Explorer のスナップショット・レポートを使用して、特定の日時におけるアクティビティーまたはプロセスの状態を判断します。

432 ページの『期間レポート』
Business Process Choreographer Explorer の期間レポートを使用して、一定期間に特定のアクティビティー・イベントまたはプロセス・イベントが発生する頻度を判別します。

関連タスク

441 ページの『ユーザー定義レポートの作成』
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、Business Process Choreographer Explorer を使用してレポート定義を格納して再利用したり、レポート結果をエクスポートしたりすることができます。

446 ページの『ユーザー定義スナップショット・レポートの作成』
指定の日時に状態情報のスナップショットを取得するユーザー定義レポートを Business Process Choreographer Explorer で定義できます。レポート作成期間内の定期的な (各月の初日の午前 0 時などの) 状態スナップショットを含むレポートを作成することもできます。

450 ページの『ユーザー定義期間レポートの作成』
Business Process Choreographer Explorer で、一定の期間に発生したプロセス・イベントまたはアクティビティー・イベントに関するユーザー定義レポートを作成できます。レポート作成サイクルに応じて、複数の期間にわたるレポートを作成することもできます。

関連資料

442 ページの『Business Process Choreographer Explorer レポートの属性』
属性を使用して、Business Process Choreographer Explorer にレポートの内容を定義し、結果をフィルターに掛けます。使用可能な属性は、レポート・タイプに応じて決まります。

444 ページの『Business Process Choreographer Explorer のビジネス・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。これらのイベントのサブセットは、Business Process Choreographer Explorer で使用可能です。

445 ページの『パフォーマンス関連の属性』
Business Process Choreographer Explorer を使用してレポート定義を実行するために必要な時間はさまざまです。レポート生成のパフォーマンスを向上させるには、レポート定義を最適化することができます。いくつかの一般的な規則に基づいて、レポートの属性がパフォーマンスに与える影響を評価できます。

保存したユーザー定義レポート定義の使用

レポート定義を Business Process Choreographer Explorer に保存した場合は、必要な

ときにレポートを実行したり、レポート定義を編集したり、レポート定義のコピーを使用して類似したレポートを作成したりすることができます。また、レポートを非同期に実行したり、レポート結果をエクスポートしたりできます。

保存したユーザー定義レポート定義の実行

保存したレポート定義は、必要なときに Business Process Choreographer Explorer を使用して実行できます。レポートがパラメーターを含む場合は、レポートを実行するたびに対象値を設定できます。

始める前に

「レポート」タブは、レポート作成が構成されている場合にのみ表示されます。レポート作成機能は、Business Process Choreographer Explorer の構成時に構成できますが、後で構成することもできます。

このタスクについて

Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 保存したレポート定義を実行するには、「レポート」タブのナビゲーション・ペインでレポートの名前をクリックします。
   - レポート定義がパラメーターを含まない場合は、生成されたレポートが表示されます。
   - レポート定義がパラメーターを含む場合は、「レポートの実行」ページが表示されます。パラメーターの値は変更できます。「**実行**」をクリックします。

     生成されたレポートが表示されます。
2. オプション: レポートの結果をエクスポートします。

   レポートされたデータを CSV 形式でエクスポートするには、「**エクスポート**」をクリックします。生成されたエクスポート・データを開くかデータをハード・ディスクに保存するかを選択し、「**OK**」をクリックします。

保存したユーザー定義レポート定義の非同期実行

Business Process Choreographer Explorer に保存したレポートを非同期に実行して、照会の実行中に作業を続行できます。

始める前に

「レポート」タブは、レポート作成が構成されている場合にのみ表示されます。レポート作成機能は、Business Process Choreographer Explorer の構成時に構成できますが、後で構成することもできます。

このタスクについて

Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 保存したレポート定義を非同期に実行するには、「レポート」タブのナビゲーション・ペインの「**ポップアップ・メニューの表示**」アイコン () をクリックし、「非同期検索 (Asynchronous Search)」アイコン () をクリックします。
2. レポート定義がパラメーターを含む場合は、「レポートの実行」ページが表示されます。パラメーターの値は変更できます。「実行」をクリックします。
   - 非同期検索が正常に完了した後、「非同期検索の完了 (Asynchronous Search Completed)」アイコン () がナビゲーション・ペインに表示されます。レポートの名前をクリックして、検索結果を表示します。
   - 非同期検索が正常に完了しない場合、「非同期検索の失敗 (Asynchronous Search Failed)」アイコン () が表示されます。

ポップアップ・メニューを使用したレポート結果のエクスポート

Business Process Choreographer Explorer に保存されたユーザー定義レポートの場合、レポートを実行しないでさらに外部処理を行うためにレポート結果をエクスポートできます。

始める前に

「レポート」タブは、レポート作成が構成されている場合にのみ表示されます。レポート作成機能は、Business Process Choreographer Explorer の構成時に構成できますが、後で構成することもできます。

このタスクについて

このオプションは、パラメーターを含まない保存されたユーザー定義レポート定義にのみ使用可能です。 Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 保存されたレポート定義のレポート結果をエクスポートするには、「レポート」タブのナビゲーション・ペインの「**ポップアップ・メニューの表示**」アイコン () をクリックし、「エクスポート」アイコン () をクリックします。
2. 生成されたエクスポート・データを開くか保存するかを選択し、「**OK**」をクリックします。 レポートするデータがエクスポートされます。

エクスポート・クライアントを使用したレポート結果のエクスポート

保存されたユーザー定義レポートの場合、エクスポート・クライアント・コマンド行ツールを使用して、レポートを実行し、さらに外部処理を行うためにレポート結果をエクスポートできます。

始める前に

このオプションは、パラメーターを含まない保存されたユーザー定義レポート定義にのみ使用可能です。

エクスポート・クライアント・ツール *wps_install_root*/ProcessChoreographer/util/bpcobserverexporter.jar は、ローカル・ワークステーションにインストールする必要があります。

手順

レポートを実行し、レポート結果をエクスポートするには、コマンド行を使用してエクスポート・クライアントを開始します。
以下のコマンドを入力します。java -jar bpcobserverexporter.jar *options*
コマンド行に -option value -option value ... の形式でオプションを直接指定するか、プロパティー・ファイルの名前を指定することができます。プロパティー・ファイルの場合、オプションの形式は、option=value です。コマンド行で指定されるオプションは、プロパティー・ファイルで指定されるものより優先します。
有効なオプションは以下のとおりです。

表 29. エクスポート・クライアントに有効なオプション

オプション	説明
help	使用法の情報を表示します。
verbose	結果のエクスポート時に、デバッグに使用できる追加情報を表示します。
unicode	結果を UTF-8 エンコード方式でエクスポートします。デフォルトは、ローカル・オペレーティング・システム・エンコード方式です。
o	既存のファイルを上書きします。ファイルが既に存在する場合、デフォルトはエラーです。
properties	これは、追加オプションを含む完全修飾ファイル名を定義します。
url	Business Process Choreographer Explorer が稼働している完全な URL。デフォルトは *http://localhost:9080* です。
out	これは、エクスポート結果を保管するための完全修飾ファイル名を定義します。デフォルトは *report name.csv* です。
userid	セキュリティーが有効にされている場合、有効なユーザー ID が必要です。
password	セキュリティーが有効にされている場合、有効なパスワードが必要です。
reportname	保存されたレポート定義の名前が必要です。エクスポート・クライアントを使用したエクスポートは、パラメーターを含まない、保存されたユーザー定義レポート定義に対してのみ機能します。

保存したユーザー定義レポート定義の編集およびコピー

Business Process Choreographer Explorer に保存したレポート定義の設定を変更したり、レポート定義のコピーを使用して類似したレポートを作成したりすることができます。

始める前に

「レポート」タブは、レポート作成が構成されている場合にのみ表示されます。レポート作成機能は、Business Process Choreographer Explorer の構成時に構成できますが、後で構成することもできます。

このタスクについて

Business Process Choreographer Explorer で次のステップを実行します。

手順

1. 「レポート」タブのナビゲーション・ペインの「**ポップアップ・メニューの表示**」アイコン () をクリックし、以下のいずれかを実行します。
   - レポート定義を編集するには、「**編集**」アイコン () をクリックします。
   - レポート定義をコピーするには、「**コピー**」アイコン () をクリックします。

   「要約」ページが開きます。このページには、時間設定、レポートの内容、およびレポートのフィルター設定が表示されます。

   対応する設定を変更するには、各要約セクションの下にあるリンクをクリックします。レポート・タイプを変更することはできません。
2. オプション: 時間設定を編集するには、「**レポートの日付およびレポート作成サイクルの設定を変更**」をクリックします。

   定義したレポートのタイプに応じて、「スナップショット・タイプの選択」ページまたは「期間タイプの選択」ページが開きます。
3. オプション: レポートの内容を変更するには、「**結果の内容を変更**」をクリックします。

   「レポート内容の指定」ページが開きます。

   レポートにレポート作成サイクルが指定されている場合は、定義したレポートのタイプに応じて、スナップショット数属性またはタイム・スライス属性属性が属性のリストに含まれます。この属性を削除することはできません。
4. オプション: フィルター設定を変更するには、「**フィルタリングを変更**」をクリックします。

   「内容のフィルタリングの指定」ページが開きます。
5. 「要約」ページで、以下のいずれかを実行します。
   - レポート定義がパラメーターを含まない場合は、「**実行**」をクリックします。

     生成されたレポートが表示されます。

- レポート定義がパラメーターを含む場合は、「**次へ**」をクリックします。

  パラメーターの値は変更できます。「**実行**」をクリックします。生成されたレポートが表示されます。

  レポート結果が予想と異なる場合は、「**編集**」をクリックしてレポートの設定を変更できます。

6. 「レポートの結果」ページで、「**保管**」をクリックします。レポート定義のコピーを作成する場合は、新しいレポートの名前を入力して「**保管**」を再度クリックします。

   ナビゲーション・ペインに新規レポートが表示されます。

関連概念

434 ページの『時間処理』
レポートで、Business Process Choreographer Explorer がタイム・スタンプと期間を処理する方法を検討します。

関連資料

442 ページの『Business Process Choreographer Explorer レポートの属性』
属性を使用して、Business Process Choreographer Explorer にレポートの内容を定義し、結果をフィルターに掛けます。使用可能な属性は、レポート・タイプに応じて決まります。

444 ページの『Business Process Choreographer Explorer のビジネス・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。これらのイベントのサブセットは、Business Process Choreographer Explorer で使用可能です。

445 ページの『パフォーマンス関連の属性』
Business Process Choreographer Explorer を使用してレポート定義を実行するために必要な時間はさまざまです。レポート生成のパフォーマンスを向上させるには、レポート定義を最適化することができます。いくつかの一般的な規則に基づいて、レポートの属性がパフォーマンスに与える影響を評価できます。

保存したユーザー定義レポート定義の削除

ナビゲーション・ペインをきれいに整理して管理しやすくするには、Business Process Choreographer Explorer 内の古いレポート定義や不要なレポート定義を削除します。

始める前に

「レポート」タブは、レポート作成が構成されている場合にのみ表示されます。レポート作成機能は、Business Process Choreographer Explorer の構成時に構成できますが、後で構成することもできます。 削除したレポート定義を復元することはできません。

このタスクについて

Business Process Choreographer Explorer で次のステップを実行します。

手順

レポート定義を削除するには、「レポート」タブのナビゲーション・ペインの「**ポップアップ・メニューの表示**」アイコン（  ）をクリックし、「**削除**」アイコン（  ）をクリックします。

タスクの結果

レポート名がナビゲーション・ペインから削除されます。

第 4 部 モジュールの開発とデプロイ

第 9 章 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。

このタスクについて

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・タスクと対話するためのプログラミング・インターフェースの比較

ビジネス・プロセスおよびヒューマン・タスクと対話するクライアント・アプリケーションの作成には、Enterprise JavaBeans (EJB)、Web サービス、 Java Message Service (JMS) および Representational State Transfer (REST) サービスなどの汎用プログラミング・インターフェースを使用できます。これらのインターフェースは、それぞれ特性が異なります。

どのプログラミング・インターフェースを選択するかは、クライアント・アプリケーションに求める機能、既存のエンド・ユーザー・クライアント・インフラストラクチャーがあるかどうか、ヒューマン・ワークフローを処理するかどうかなど、いくつかの要因によって決まります。使用するインターフェースを決定するためのヒントとして、 EJB、Web サービス、JMS、および REST プログラミング・インターフェースの特性を比較した表を以下に示します。

	EJB インターフェース	Web サービス・インターフェース	JMS メッセージ・インターフェース	REST インターフェース
機能	このインターフェースは、ビジネス・プロセスとヒューマン・タスクの両方に使用可能です。このインターフェースは、一般的な方法でプロセスおよびタスクを処理するクライアントを作成するために使用します。	このインターフェースは、ビジネス・プロセスとヒューマン・タスクの両方に使用可能です。このインターフェースは、既知の一式のプロセスおよびタスクに対するクライアントを作成するために使用します。	このインターフェースはビジネス・プロセスの場合に限って使用可能です。このインターフェースは、既知の一式のプロセスに対するメッセージング・クライアントを作成するために使用します。	このインターフェースは、ビジネス・プロセスとヒューマン・タスクの両方に使用可能です。このインターフェースは、既知の一式のプロセスおよびタスクに対する Web 2.0 スタイルのクライアントを作成するために使用します。
データ処理	ビジネス・オブジェクト・メタデータにアクセスするためのスキーマのリモート成果物ロードをサポートします。 EJB クライアント・アプリケーションが接続先 WebSphere Process Server と同じセルで実行されている場合は、プロセスおよびタスクのビジネス・オブジェクトに必要なスキーマをクライアントで使用可能にする必要はなく、リモート成果物ローダー (RAL) を使用してサーバーからロードできます。 クライアント・アプリケーションを完全な WebSphere Process Server サーバー・インストールで実行する場合は、RAL をクロス・セルで使用することもできます。ただし、クライアント・アプリケーションを WebSphere Process Server クライアント・インストールで実行する場合は、RAL をクロス・セル・セットアップで使用することはできません。	入力データ、出力データ、および変数に対するスキーマ成果物が、クライアント上で、適切な形式で提供されていなければなりません。	入力データ、出力データ、および変数に対するスキーマ成果物が、クライアント上で、適切な形式で提供されていなければなりません。	入力データ、出力データ、および変数に対するスキーマ成果物が、クライアント上で、適切な形式で提供されていなければなりません。
クライアント環境	WebSphere Process Server インストールまたは WebSphere Process Server クライアント・インストール。	Web サービスの呼び出しをサポートする任意のランタイム環境 (Microsoft .NET 環境など)。	SCA JMS インポートを使用する SCA モジュールなど、JMS クライアントをサポートする任意のランタイム環境。	REST クライアントをサポートする任意のランタイム環境。

	EJB インターフェース	Web サービス・インターフェース	JMS メッセージ・インターフェース	REST インターフェース
セキュリティー	Java Platform, Enterprise Edition (Java EE) セキュリティー。	Web サービスのセキュリティー。	WebSphere Process Server インストール用の Java Platform, Enterprise Edition (Java EE) セキュリティー。JMS クライアント・アプリケーションが API メッセージを書き込むキューを、例えば WebSphere MQ のセキュリティー・メカニズムを使用する方法で、保護することもできます。	REST メソッドを呼び出すクライアント・アプリケーションは、適切な HTTP 認証メカニズムを使用する必要があります。

操作は複数のプロトコルによって公開できます。同一の操作を異なる複数のプロトコルで使用する場合は、以下の一般的な考慮事項に留意してください。

- Web サービス・インターフェースおよび REST インターフェースでは、PIID、AIID、および TKIID などのオブジェクト ID はすべて string 型で表されます。タイプ・セーフなオブジェクト ID を予期するのは EJB API インターフェースのみです。
- 操作の過負荷は EJB メソッドのみで使用され、WSDL 操作では使用されません。ある場合は複数の WSDL 操作が存在し、また別の場合は、省略 (`minOccurs="0"`) またはヌル値 (`nillable="true"`) によってパラメーターのすべてのバリエーションを許可する 1 つの WSDL 操作のみが存在します。
- 一部の EJB メソッドでは、XML 名前空間とローカル名は別個のパラメーターとして渡されます。多くの場合、WSDL 操作は QName XML スキーマ・タイプを使用してこれらのパラメーターを渡します。
- EJB インターフェースでの callWithReplyContext 操作や WSDL インターフェースでの callAsync 操作などの、長時間にわたる WSDL 要求応答操作との非同期対話は、JMS インターフェースでは call 操作によって表されます。
- EJB インターフェースは、一連の API オブジェクトを返します。これらのオブジェクトは、組み込まれているフィールドの getter メソッドおよび setter メソッドを公開します。Web サービス・インターフェースおよび REST インターフェースは、クライアントに対して複合タイプ (XML または JSON) の文書を返します。
- ヒューマン・タスクに対して機能する一部の Human Task Manager サービスは、ヒューマン・タスクを呼び出すアクティビティーに対して機能する Business Flow Manager サービスとしても使用可能です。

第 10 章 ビジネス・プロセスおよびタスク・データに対する照会

長時間実行ビジネス・プロセスおよびヒューマン・タスクのインスタンス・データはデータベースに永続的に格納され、照会によってアクセスできます。また、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・テンプレートのテンプレート・データには、QUERY インターフェースを使用してアクセスできます。

Business Process Choreographer では、EJB 照会インターフェース、照会 API、および照会テーブル API を使用できます。

プロセスまたはタスクの関連データにアクセスするクライアントによっては、インターフェースの 1 つ以上に対応している可能性があります。Business Process Choreographer では、タスクおよびプロセス・リスト・データを照会するために REST および Web サービス API を使用できます。ただし、大容量のプロセス・リストおよびタスク・リストの照会には、パフォーマンス上の理由から、Business Process Choreographer EJB 照会テーブル API および REST 照会テーブル API を使用してください。

プロセスおよびタスク・データを検索するためのプログラミング・インターフェースの比較

Business Process Choreographer には、プロセスおよびタスク・データを検索するための照会テーブル API および照会 API があります。これらのインターフェースは、それぞれ特性が異なります。

選択する照会インターフェースは、いくつかの要因によって左右されます。この要因には、クライアント・アプリケーションが提供する必要がある機能、既存のエンド・ユーザー・クライアント・インフラストラクチャーが存在するかどうか、パフォーマンス上の考慮事項などがあります。使用するインターフェースを決定する際の参考として、照会テーブルおよび照会プログラミング・インターフェースの特性を比較した表を以下に示します。

特性	照会テーブル API	照会 API
アベイラビリティー	照会テーブル API は、Business Flow Manager EJB インターフェースおよび REST プログラミング・インターフェースで使用可能です。	照会 API は、EJB、Web サービス、JMS、および REST プログラミング・インターフェースで使用可能です。
内容検索のためのメソッド	この API には、以下のメソッドがあります。 • queryEntities • queryEntityCount • queryRows • queryRowCount	この API には、以下のメソッドがあります。 • query • queryAll • queryProcessTemplates • queryTaskTemplates
メタデータ検索のためのメソッド	この API には、以下のメソッドがあります。 • getQueryTableMetaData • findQueryTableMetaData	この API には、以下のメソッドがあります。 • QueryResultSet.getColumnType

特性	照会テーブル API	照会 API
照会テーブル名	照会テーブル API を実行する照会テーブルを指定します。一度に照会できる照会テーブルは 1 つだけです。 例えば `queryEntities("CUST.TASKS", ...)` と記述します。	SELECT 文節は、照会を実行する列および定義済みデータベース・ビューを指定します。この仕様は SQL の select 文節と同様です。 例えば `query("TASK.TKIID, TASK.STATE, WORK_ITEM.REASON", ...)` と記述します。
SELECT 文節および選択された属性	照会テーブル API のフィルター・オプションを使用して、照会が返す属性を指定します。照会は 1 つの照会テーブルに対して実行されるため、属性は名前によって一意的に識別できます。	SELECT 文節を使用して、属性を指定します。属性名の構文は、*view_name.attribute_name* です。例えば、タスク状態を検索するには、照会で TASK.STATE と指定します。
WHERE 文節およびフィルター	照会の結果をさらにフィルターに掛けるには、照会テーブル API で queryCondition プロパティーを使用します。1 次照会テーブル・フィルター、許可フィルター、または照会テーブル・フィルターが照会テーブル定義で指定されている場合、照会テーブルは、事前にフィルターに掛けた内容を提供します。	WHERE 文節を使用して、照会の結果をフィルター操作します。
WHERE 文節および選択基準	この形式の照会テーブル API では、照会 API の WHERE 文節は不要です。さらにフィルターに掛けるには、照会テーブル API で queryCondition プロパティーを使用します。 照会テーブル定義の選択基準は、接続される照会テーブルの特定のプロパティーを選択します。これは、照会 API の WHERE 文節によるフィルター処理に加えて実行されます。	選択基準は、照会 API では使用できません。しかし、選択基準は、QUERY_PROPERTY、TASK_CPROP、または TASK_DESC の名前またはロケールなどを定義する WHERE 文節の特定の部分と似ています。 例えば、`QUERY_PROPERTY.NAME='xyz'` の WHERE 文節は、QUERY_PROPERTY 接続される照会テーブルの照会テーブル定義で選択基準として `NAME='xyz'` を指定することと同じです。
作業項目および許可	WORK_ITEM 照会テーブルを使用して、作業項目にアクセスします。作業項目の使用方法は、照会テーブルの作成時に照会テーブル定義でカスタマイズしたり、AuthorizationOptions オブジェクトまたは AdminAuthorizationOptions オブジェクトを使用して照会テーブル API でカスタマイズすることができます。 例えば、TASK 照会テーブルを照会するときに全員作業項目を除外するには、queryCondition プロパティー `WI.EVERYBODY=0` を指定するか、AuthorizationOptions プロパティーで `setUseEverbody(Boolean.FALSE)` を指定します。	WORK_ITEM ビューを使用して、作業項目にアクセスします。照会結果では、作業項目の 4 つのタイプ (全員、個人、グループ、および継承) のすべてが考慮されます。作業項目を特定のタイプの作業項目にフィルタリングするには、WHERE 文節をカスタマイズします。 例えば、「全員」作業項目を除外するには、WHERE 文節に `WORK_ITEM.EVERYBODY=0` を指定します。
パラメーター	複合照会テーブルのフィルターおよび選択基準にパラメーターを使用できます。	保管照会文を使用していない場合は、照会 API のパラメーターを使用できません。
保管照会文と照会テーブル	保管照会文と照会テーブルの違いは、保管照会文は 1 つの特定の照会用に定義されるのに対し、照会テーブルは特定の照会セット用に定義されるという点にあります。例えば、照会テーブル定義では、order-by 文節を指定することはできません。通常、この情報は、照会の実行時にのみ利用できるからです。	保管照会文を使用して、定義済みのオプション・セットが含まれている照会を実行できます。
実体化ビュー	実体化ビューは、照会テーブル API では使用できません。	実体化ビューでは、照会のパフォーマンス向上のために、データベース技術が使用されます。
カスタム・テーブル	補足照会テーブルは、カスタム・テーブルと同じ機能を提供します。	カスタム・テーブルは、Business Process Choreographer データベース・スキーマの外部にあるデータを照会に組み込むために使用されます。

特性	照会テーブル API	照会 API
queryAll および許可オプション	queryAll 機能は、AdminAuthorizationOptions オブジェクトで提供されます。これは、AuthorizationOptions オブジェクトではなく照会テーブル API に渡すことができます。呼び出し元は、BPESystemAdministrator、TaskSystemAdministrator、 BPESystemMonitor、または TaskSystemMonitor にある必要があります。	queryAll メソッドは、BPESystemAdministrator Java EE ロールを持つユーザーが、特定のユーザーまたはグループの作業項目によって制限されることなく、照会結果内のすべてのオブジェクトを返すために使用できます。
国際化対応	照会テーブルの使用時に、照会テーブルの属性および照会テーブルで、ローカライズされた表示名および説明を使用できるようになっています。	選択したビューの列の名前が、データベースに表示されているとおりに、または選択した文節で指定されているとおりに返されます。

Business Process Choreographer での照会テーブル

照会テーブルは、Business Process Choreographer データベース・スキーマに含まれているデータに対するタスクおよびプロセス・リストの照会をサポートします。これには、ヒューマン・タスク・データ、Business Process Choreographer によって管理されるビジネス・プロセス・データ、および外部ビジネス・データが含まれます。照会テーブルは、クライアント・アプリケーションが使用できる Business Process Choreographer のデータの抽象化を提供します。このため、クライアント・アプリケーションに、照会テーブルを実際に実装する必要はありません。照会テーブル定義は Business Process Choreographer コンテナーにデプロイされており、照会テーブル API を使用してアクセス可能です。

照会テーブルには、以下の 3 つのタイプがあります。

- 定義済み照会テーブル
- 補足照会テーブル
- 複合照会テーブル

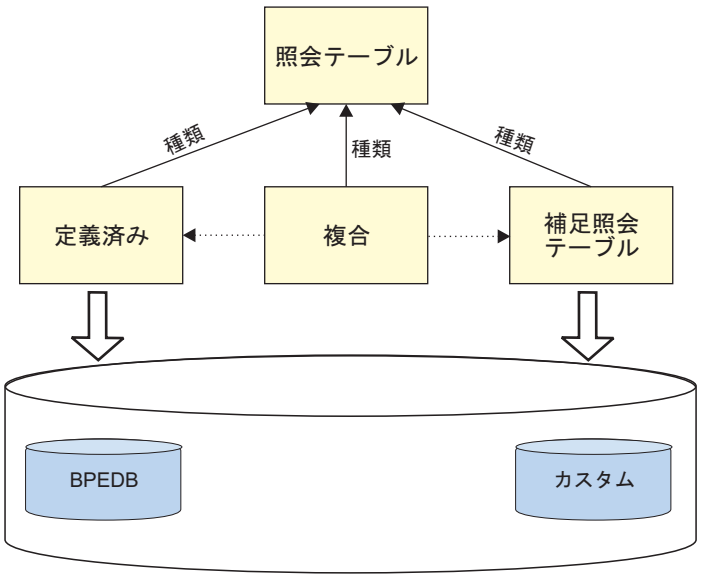


図 4. Business Process Choreographer での照会テーブル

照会テーブルは、照会テーブル・ランタイムの類似したモデルを使用して表現され、照会テーブル API を使用して照会することができます。定義済み照会テーブルおよび補足照会テーブルはデータベース内のテーブルまたはビューを直接指しますが、複合照会テーブルはこのデータの各部分をまとめて、単一の照会テーブルで表します。

照会テーブルは、Business Process Choreographer の事前定義データベース・ビューと既存の QUERY インターフェースを拡張するもので、以下のような特徴があります。

- パフォーマンスの観点から最適化されたアクセス・パターンを使用して、プロセスおよびタスク・リストの照会を実行するために最適化されています。
- 必要な情報へのアクセスが簡素化され、統合されます。
- 許可およびフィルターのオプションをきめ細かく構成できます。

照会テーブルはカスタマイズできます。例えば、特定のシナリオに関連するタスクまたはプロセス・インスタンスだけが含まれるように照会テーブルを構成できます。パフォーマンスが重要な場合 (大容量のプロセス・リストやタスク・リストを照会する場合など) も、照会テーブルを使用することをお勧めします。

Query Table Builder が Eclipse プラグインとして提供されていて、以下を行うことができます。

- 複合照会テーブルおよび補足照会テーブルの作成
- XML 形式での照会テーブル定義のインポートおよびエクスポート

Query Table Builder は、WebSphere Business Process Management SupportPacs のサイトでダウンロードできます。このサイトで、PA71 WebSphere Process Server - Query Table Builder を探します。リンクにアクセスするには、このトピックの関連参照のセクションを参照してください。

定義済み照会テーブル

定義済み照会テーブルは、Business Process Choreographer データベース内のデータへのアクセスを提供します。定義済み照会テーブルは、対応する定義済み Business Process Choreographer データベース・ビュー (TASK ビューまたは PROCESS_INSTANCE ビューなど) を照会テーブルの形式で表現したものです。これらの定義済み照会テーブルは、プロセスおよびタスク・リストの照会を実行するために最適化されているため、定義済みデータベース・ビューよりも機能とパフォーマンスが強化されています。

定義済み照会テーブルは、照会テーブル API を使用して直接照会できます。照会テーブル API を使用してテーブルにアクセスする場合は、照会 API を使用する場合と比較して、より多くの構成用オプションを使用できます。

プロパティー

定義済み照会テーブルには、以下のプロパティーがあります。

表 30. 定義済み照会テーブルのプロパティー

プロパティー	説明
名前	照会テーブル名は、いずれかの事前定義データベース・ビューの名前を大文字で表記したもの (TASK など) です。
属性	定義済み照会テーブルの属性は、照会に使用できる情報を定義します。これらは、定義済みデータベース・ビューによって指定される大文字の列の名前です。 属性は、名前とタイプで定義します。属性のタイプは、以下のいずれかです。 • **ブール**: ブール値 • **10 進数**: 浮動小数点数 • **ID**: オブジェクト ID (TASK 照会テーブルの TKIID など) • **番号**: 整数、短整数、または長整数 • **ストリング**: ストリング • **タイム・スタンプ**: タイム・スタンプ
許可	定義済み照会テーブルは、インスタンス・ベースまたはロール・ベースの許可を使用します。 • インスタンス・データを含む定義済み照会テーブルは、インスタンス・ベースの許可を必要とします。そのため、照会を実行するユーザーの作業項目を含むオブジェクトのみが返されます。ただし、AdminAuthorizationOptions オブジェクトを使用すると、いずれかのユーザーの作業項目が存在するかどうかの検査に単純化することができます。これらの照会では、ユーザーは、BPESystemAdministrator Java EE ロール (Business Flow Manager EJB を使用している場合) または TaskSystemAdministrator Java EE ロール (Human Task Manager EJB を使用している場合) を持っている必要があります。 • テンプレート・データを含む定義済み照会テーブルでは、ロール・ベースの許可が必要です。すなわち、BPESystemAdministrator Java EE ロール (Business Flow Manager EJB を使用している場合) または TaskSystemAdministrator Java EE ロール (Human Task Manager EJB を使用している場合) のユーザーのみが、それらの照会テーブルの内容にアクセスできます。

インスタンス・データを含む定義済み照会テーブル

以下の表は、インスタンス・データが含まれている定義済み照会テーブルを示しています。このような照会テーブルについては、以下の点が当てはまります。

- 複合照会テーブルの 1 次照会テーブルとして使用できます。
- 直接照会が実行される場合に、インスタンス・ベースの許可を使用します。これは、許可情報を格納するビュー (定義済み WORK_ITEM ビューまたは照会テーブル) との結合 (SQL-) によって実現されます。

- インスタンス・データ (タスク・インスタンスまたはプロセス・インスタンスのデータなど) が入ります。

表 31. インスタンス・データが含まれている定義済み照会テーブル

インスタンス・データ	照会テーブル名
プロセス・インスタンスのアクティビティーについての情報。	ACTIVITY
	ACTIVITY_ATTRIBUTE
	ACTIVITY_SERVICE
ヒューマン・タスクに属するエスカレーションについての情報。	ESCALATION
	ESCALATION_CPROP
	ESCALATION_DESC
プロセス・インスタンスについての情報。	PROCESS_ATTRIBUTE
	PROCESS_INSTANCE
	QUERY_PROPERTY
ヒューマン・タスクについての情報。	TASK
	TASK_CPROP
	TASK_DESC

WORK_ITEM 照会テーブルにもインスタンス・データは含まれていますが、このデータを 1 次照会テーブルまたは接続される照会テーブルとして使用することはできません。作業項目の情報は、インスタンス・ベースの許可を使用する照会テーブルの照会時に暗黙的に使用可能です。つまり、WORK_ITEM 照会テーブルの属性は、その属性が照会テーブルによって明示的に指定されていないとしても、インスタンス・ベースの許可を使用する照会テーブルの照会時に使用できます。

テンプレート・データを含む定義済み照会テーブル

テンプレート・データを含む定義済み照会テーブルは、ロール・ベースの許可を必要とします。これらの定義済み照会テーブルは、管理者が AdminAuthorizationOptions オブジェクトを使用することによってのみ照会できます。

以下の表は、テンプレート・データが含まれている定義済み照会テーブルを示しています。このような照会テーブルについては、以下の点が当てはまります。

- 複合照会テーブルの 1 次照会テーブルとして使用できます。
- 直接照会が実行される場合に、ロール・ベースの許可を使用します。そのため、API 照会メソッドを使用する呼び出し元は、BPESystemAdministrator Java EE ロール (Business Flow Manager EJB を使用している場合) または TaskSystemAdministrator Java EE ロール (Human Task Manager EJB を使用している場合) でなければならず、AdminAuthorizationOptions を使用する必要があります。
- タスク・テンプレートまたはプロセス・テンプレートのテンプレート・データなどのテンプレート・データが含まれます。

表 32. テンプレート・データが含まれている定義済み照会テーブル

テンプレート・データ	照会テーブル名
アプリケーション・コンポーネントについての情報。	APPLICATION_COMP
エスカレーション・テンプレートについての情報。	ESC_TEMPL
	ESC_TEMPL_CPROP
	ESC_TEMPL_DESC
プロセス・テンプレートについての情報。	PROCESS_TEMPLATE
	PROCESS_TEMPL_ATTR
タスク・テンプレートについての情報。	TASK_TEMPL
	TASK_TEMPL_CPROP
	TASK_TEMPL_DESC

関連概念

『補足照会テーブル』
Business Process Choreographer の補足照会テーブルは、Business Process Choreographer の管理対象ではないビジネス・データを照会テーブル API に対して公開します。補足照会テーブルを使用すると、ビジネス・プロセス・インスタンス情報またはヒューマン・タスク情報の取得時に、この外部データを定義済み照会テーブルのデータと一緒に使用することができます。

477 ページの『複合照会テーブル』
Business Process Choreographer の複合照会テーブルでは、データベース内のデータが固有の方法で表現されることはありません。このテーブルは、関連した定義済み照会テーブルおよび補足照会テーブルのデータの組み合わせからなります。複合照会テーブルを使用して、プロセス・インスタンス・リストまたはタスク・リスト (ユーザーの予定など) の情報を取得します。

485 ページの『照会テーブルの作成』
Business Process Choreographer の補足照会テーブルおよび複合照会テーブルは、Query Table Builder を使用してアプリケーションの開発中に作成します。定義済み照会テーブルは、作成することもデプロイすることもできません。定義済み照会テーブルは、 Business Process Choreographer のインストール時に利用できる照会テーブルで、Business Process Choreographer データベース・スキーマでの成果物を簡略表示することができます。

505 ページの『照会テーブルの照会』
照会は、照会テーブル API (Business Flow Manager EJB でのみ使用可能) を使用して、Business Process Choreographer 内の照会テーブルに対して実行されます。

494 ページの『照会テーブルの許可』
照会テーブルで照会を実行するときには、インスタンス・ベースの許可、ロール・ベースの許可、または許可なしを使用できます。

補足照会テーブル

Business Process Choreographer の補足照会テーブルは、Business Process Choreographer の管理対象ではないビジネス・データを照会テーブル API に対して公開します。補足照会テーブルを使用すると、ビジネス・プロセス・インスタンス

情報またはヒューマン・タスク情報の取得時に、この外部データを定義済み照会テーブルのデータと一緒に使用することができます。

補足照会テーブルは、Business Process Choreographer データベースのデータベース表またはデータベース・ビューと関連しています。これらは、カスタマー・アプリケーションによって保守されるビジネス・データが含まれている照会テーブルです。補足照会テーブルは、定義済み照会テーブルに含まれている情報に加えて、複合照会テーブルの情報も提供します。

補足照会テーブルには、以下のプロパティーがあります。

表 33. 補足照会テーブルのプロパティー

プロパティー	**説明**
名前	照会テーブル名は、Business Process Choreographer インストール済み環境内で固有でなければなりません。この名前は、照会の実行時に、照会される照会テーブルを識別するために使用されます。 照会テーブルは、名前 (*prefix.name* と定義される) によって一意的に識別されます。*prefix.name* の最大長は 28 文字です。接頭部は、予約済みの接頭部 `'IBM'` と異なっていなければなりません。例えば、`'COMPANY.BUS_DATA'` など。テーブル名の末尾に数字を使用しないでください。テーブルが照会内で複数回使用されている場合、テーブルの名前には 0 から 9 までの数字が付加されます。例えば、CUSTOM_VIEW0、CUSTOM_VIEW1 (以下同様) のようになります。テーブル名の末尾に既に数字が使用されている場合、Business Process Choreographer はこの数字を削除し、QueryUnknownTableException が発生します。
データベース名	データベースの関連テーブルまたはビューの名前。使用できるのは大文字のみです。
データベース・スキーマ	データベースの関連テーブルまたはビューのスキーマ。使用できるのは大文字のみです。データベース・スキーマは、Business Process Choreographer データベースのデータベース・スキーマとは異なっている必要があります。それでも、テーブルまたはビューには、Business Process Choreographer データベースにアクセスするために使用するのと同じ JDBC データ・ソースを使用してアクセス可能でなければなりません。
属性	補足照会テーブルの属性は、照会に使用できる情報を定義します。この属性は、関連データベース表またはビュー内の列の関連名と一致している必要があります。 属性は、名前とタイプで定義します。名前は大文字で定義します。属性のタイプは、以下のいずれかです。 • **ブール**: ブール値 • **10 進数**: 浮動小数点数 • **ID**: 長さが 16 バイトのオブジェクト ID (TASK 照会テーブルの TKIID など) • **番号**: 整数、短整数、または長整数 • **ストリング**: ストリング • **タイム・スタンプ**: タイム・スタンプ

表 33. 補足照会テーブルのプロパティー (続き)

プロパティー	説明
結合	結合は、複合照会テーブル内で接続する場合は、補足照会テーブルで定義する必要があります。結合は、補足照会テーブルの情報と 1 次照会テーブルの情報を相関させるために使用する属性を定義します。結合を定義する場合、ソース属性とターゲット属性のタイプは同じでなければなりません。
許可	補足照会テーブルには許可が指定されないため、すべての認証済みユーザーがコンテンツを参照できます。

関連概念

472 ページの『定義済み照会テーブル』
定義済み照会テーブルは、Business Process Choreographer データベース内のデータへのアクセスを提供します。定義済み照会テーブルは、対応する定義済み Business Process Choreographer データベース・ビュー (TASK ビューまたは PROCESS_INSTANCE ビューなど) を照会テーブルの形式で表現したものです。これらの定義済み照会テーブルは、プロセスおよびタスク・リストの照会を実行するために最適化されているため、定義済みデータベース・ビューよりも機能とパフォーマンスが強化されています。

『複合照会テーブル』
Business Process Choreographer の複合照会テーブルでは、データベース内のデータが固有の方法で表現されることはありません。このテーブルは、関連した定義済み照会テーブルおよび補足照会テーブルのデータの組み合わせからなります。複合照会テーブルを使用して、プロセス・インスタンス・リストまたはタスク・リスト (ユーザーの予定など) の情報を取得します。

485 ページの『照会テーブルの作成』
Business Process Choreographer の補足照会テーブルおよび複合照会テーブルは、Query Table Builder を使用してアプリケーションの開発中に作成します。定義済み照会テーブルは、作成することもデプロイすることもできません。定義済み照会テーブルは、 Business Process Choreographer のインストール時に利用できる照会テーブルで、Business Process Choreographer データベース・スキーマでの成果物を簡略表示することができます。

505 ページの『照会テーブルの照会』
照会は、照会テーブル API (Business Flow Manager EJB でのみ使用可能) を使用して、Business Process Choreographer 内の照会テーブルに対して実行されます。

494 ページの『照会テーブルの許可』
照会テーブルで照会を実行するときには、インスタンス・ベースの許可、ロール・ベースの許可、または許可なしを使用できます。

複合照会テーブル

Business Process Choreographer の複合照会テーブルでは、データベース内のデータが固有の方法で表現されることはありません。このテーブルは、関連した定義済み照会テーブルおよび補足照会テーブルのデータの組み合わせからなります。複合照会テーブルを使用して、プロセス・インスタンス・リストまたはタスク・リスト (ユーザーの予定など) の情報を取得します。

複合照会テーブルはクライアント開発者によって設計され、照会実行時のデータ・アクセスを最適化するために、フィルターおよび許可オプションを詳細に構成することができます。複合照会テーブルは、タスク・リストおよびプロセス・リストの照会に最適化された SQL で実現されています。

複合照会テーブルは、照会の実際の実装を抽象化することによって照会を最適化することができるため、Business Process Choreographer の標準照会 API の代わりに実動シナリオで使用することが推奨されています。

さらに、照会テーブルにアクセスするクライアントを再デプロイしなくても、実行時に複合照会テーブルを変更できます。

以下の図は、複合照会テーブルの内容の概要を示しています。

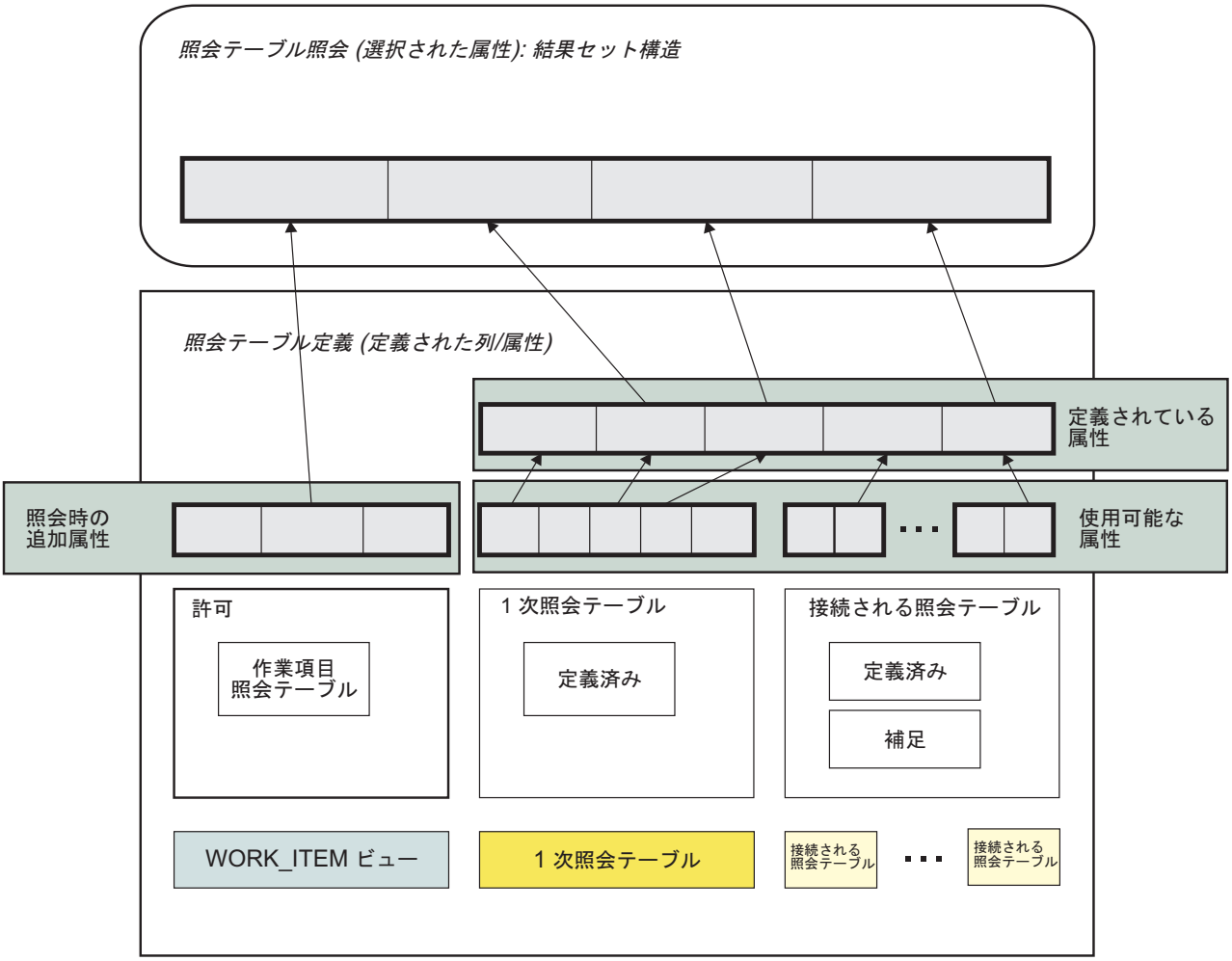


図 5. 複合照会テーブルの内容

すべての複合照会テーブルは、1 つの 1 次照会テーブルと 0 個以上の接続される照会テーブルで定義されます。

1 次照会テーブル:

- 複合照会テーブルに格納される主な情報を構成します。
- いずれかの定義済み照会テーブルでなければなりません。

- 複合照会テーブル内の各オブジェクトを 1 次キーによって一意的に識別します。例えば、TASK 定義済み照会テーブルの場合、1 次キーはタスク ID の TKIID です。
- インスタンス・ベースの許可を使用する場合は、WORK_ITEM 照会テーブルに含まれている作業項目を使用して、照会テーブルの内容を許可します。
- 複合照会テーブルの照会時にテーブルの行として返されるオブジェクトのリストを判別します。

接続される照会テーブル:

- 既にシステムにデプロイされている定義済み照会テーブルおよび補足照会テーブルにすることができます。
- 1 次照会テーブルで提供される情報に加えて、その他の情報を提供できます。例えば、TASK が 1 次照会テーブルの場合、TASK_DESC 照会テーブルで提供されるタスクの説明を、複合照会テーブルの内容に追加することができます。

通常、1 次照会テーブルは、複合照会テーブルの目的に基づいて選択されます。

- 複合照会テーブルがタスク・リストを表す場合、TASK 照会テーブルが 1 次照会テーブルになります。
- 複合照会テーブルがプロセス・リストを表す場合、PROCESS_INSTANCE 照会テーブルが 1 次照会テーブルになります。
- アクティビティーのリストは、ACTIVITY 1 次照会テーブルを使用して取得されます。
- ヒューマン・タスク・エスカレーションのリストは、ESCALATION 1 次照会テーブルを使用して取得されます。

1 次照会テーブルと接続される照会テーブルの関係

接続される照会テーブルと 1 次照会テーブルは、1 対 1 または 1 対 0 の関係でなければなりません。1 対 1 または 1 対 0 の関係に違反すると、照会の実行時にランタイム例外が発生します。

1 次照会テーブルと接続される照会テーブルは、接続される照会テーブルで定義されている結合属性を使用して相関させます。この結合属性は、Business Process Choreographer のさまざまな照会テーブル内のデータの関係を記述しているため、定義済み照会テーブルでは変更できません。通常、1 対 1 または 1 対 0 の関係を維持する場合は、結合属性で十分です。例えば、CONTAINMENT_CTX_ID 属性は、PROCESS_INSTANCE 照会テーブルの PIID 属性で識別される関連プロセス・インスタンス情報を接続するために、TASK 照会テーブルで使用されます。

1 対多の関係が存在する場合は、照会テーブルを定義するときに、Query Table Builder で追加の基準 (選択基準と呼ばれる) を指定する必要があります。例えば、`"LOCALE='en_US'"` のように指定します。タスクには、1 つのタスクに対してさまざまなロケールで識別される、複数の説明を用意することができます。

例 1:

以下の図は、接続される照会テーブルで指定される選択基準を可視化したサンプルを示しています。

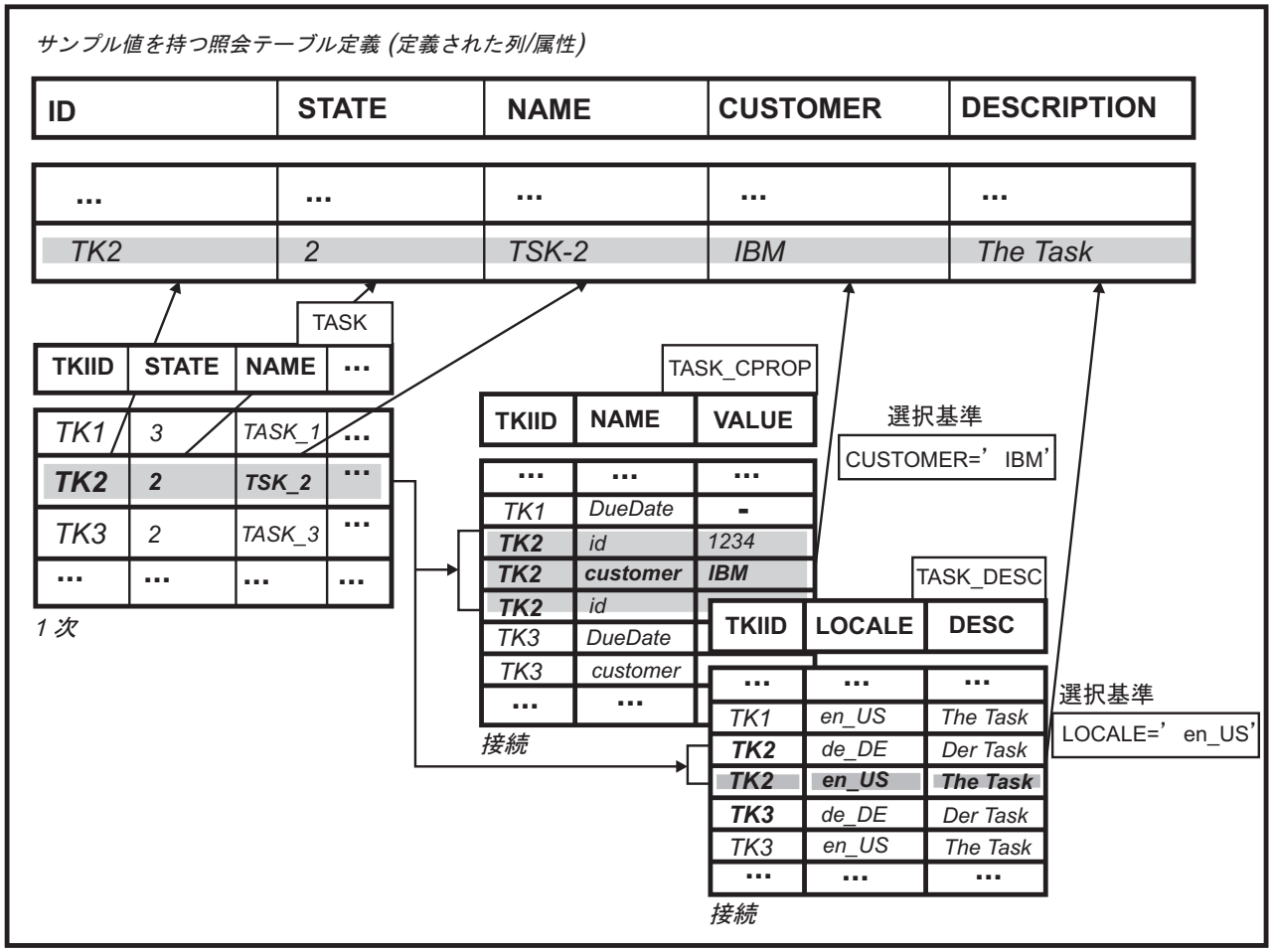


図 6. 選択基準が指定された複合照会テーブル

複合照会テーブルには、ID、STATE、NAME、CUSTOMER、および DESCRIPTION 属性が含まれています。

- ID、STATE、および NAME は、TASK 1 次照会テーブルによって提供されます。
- CUSTOMER は、TASK のカスタム・プロパティーです。カスタム・プロパティーは、TASK_CPROP 照会テーブルに格納されます。特定のタスクでは、カスタム・プロパティーは名前によって一意的に識別されます。例えば、選択基準 `"CUSTOMER='IBM'"` などがあります。
- DESCRIPTION は、TASK_DESC 照会テーブルに格納されるタスクの説明です。タスク・インスタンスごとに、そのタスクの説明がロケールによって一意的に識別されます。例えば、選択基準 `"LOCALE='en_US'"` などがあります。

例 2:

この例では、1 次照会テーブルと接続される照会テーブルの関係に焦点が当てられています。1 次照会テーブルとして TASK、接続される照会テーブルとして TASK_DESC が使用されています。複合照会テーブルを定義するときに、TASK_DESC 照会テーブルの LOCALE 属性を指定して、1 次照会テーブルと接続される照会テーブルの間に 1 対 1 または 1 対 0 の関係を持たせるようにする必

要があります。この表は、接続される照会テーブル TASK_DESC の有効な選択基準が指定された複合照会テーブルの内容のサンプルを示しています。

表 34. 複合照会テーブルの有効な内容

1 次照会テーブル TASK の情報	**接続される照会テーブル TASK_DESC の情報**	
NAME	LOCALE	DESCRIPTION
task_one	en_US	これは説明です。
task_two	en_US	これは説明です。
...	...	...

以下の表は、選択基準の設定が間違っている (1 対 1 または 1 対 0 の関係に違反している) 場合の仮想の無効な内容 (**太字**) を示しています。

表 35. 複合照会テーブルの無効な内容

TASK (1 次照会テーブル) から取得される情報	**TASK_DESC (接続される照会テーブル) から取得される情報**	
NAME	LOCALE	DESCRIPTION
task_one	en_US	これは説明です。
task_one	**de_DE**	**Das ist eine Beschreibung.**
...	...	...

プロパティー

複合照会テーブルには、以下のプロパティーがあります。

表 36. 複合照会テーブルのプロパティー

プロパティー	**説明**
名前	照会テーブル名は、Business Process Choreographer インストール済み環境内で固有でなければなりません。この照会テーブル名は、照会の実行時に、照会される照会テーブルを識別するために使用されます。 照会テーブルは、名前によって一意的に識別されます (複合照会テーブルの場合は *prefix.name* と定義されます)。*prefix.name* の最大長は 28 文字です。接頭部は、予約済みの接頭部'IBM' と異なっていなければなりません。例えば、'COMPANY.TODO_TASK_LIST' など。テーブル名の末尾に数字を使用しないでください。テーブルが照会内で複数回使用されている場合、テーブルの名前には 0 から 9 までの数字が付加されます。例えば、CUSTOM_VIEW0、CUSTOM_VIEW1 (以下同様) のようになります。テーブル名の末尾に既に数字が使用されている場合、Business Process Choreographer はこの数字を削除し、QueryUnknownTableException が発生します。

表 36. 複合照会テーブルのプロパティー (続き)

プロパティー	説明
属性	複合照会テーブルの属性は、照会に使用できる情報を定義します。 属性は、大文字の名前で定義します。タイプは参照先の属性から継承され、以下のいずれかになります。 • **ブール**: ブール値 • **10 進数**: 浮動小数点数 • **ID**: オブジェクト ID (TASK 照会テーブルの TKIID など) • **番号**: 整数、短整数、または長整数 • **ストリング**: ストリング • **タイム・スタンプ**: タイム・スタンプ 複合照会テーブルの属性は、1 次照会テーブルまたは接続される照会テーブルの属性への参照を使用して定義されます。複合照会テーブルの属性は、参照先の属性のタイプおよび定数を継承します。 照会テーブル定義の一部である属性に加えて、作業項目情報を実行時に照会することができます。これは、1 次照会テーブルがインスタンス・データを含む場合 (TASK や PROCESS_INSTANCE など)、および複合照会テーブルでインスタンス・ベースの許可を使用する場合に可能です。例えば、ユーザーが潜在的所有者であるヒューマン・タスクのみを返すように照会を定義することができます。

表 36. 複合照会テーブルのプロパティー (続き)

プロパティー	説明
許可	各複合照会テーブルでは、照会を実行するときにインスタンス・ベースの許可、ロール・ベースの許可、許可なしのいずれが使用されるかが定義されます。 インスタンス・ベースの許可が定義されている場合は、照会を実行するユーザーの作業項目が含まれているオブジェクトのみが返されます。ただし、AdminAuthorizationOptions を使用すると、いずれかのユーザーの作業項目が存在するかどうかの検査に単純化することができます。これらの照会では、ユーザーは BPESystemAdministrator Java EE ロール (Business Flow Manager EJB を使用している場合) または TaskSystemAdministrator Java EE ロール (Human Task Manager EJB を使用している場合) でなければならず、AdminAuthorizationOptions が照会テーブル API に渡される必要があります。 ロール・ベースの許可が定義されている場合、これらの照会では、ユーザーは BPESystemAdministrator Java EE ロール (Business Flow Manager EJB を使用している場合) または TaskSystemAdministrator Java EE ロール (Human Task Manager EJB を使用している場合) でなければならず、AdminAuthorizationOptions が照会テーブル API に渡される必要があります。 許可なしが定義されている場合は、関連オブジェクトの作業項目が照会テーブルに存在するかどうかが検査されずに、照会が実行されます。すべての認証済みユーザーが照会テーブルの内容を参照できます。 インスタンス・ベースの許可は、1 次照会テーブルがインスタンス・データを含む場合に定義できます。ロール・ベースの許可は、1 次照会テーブルがテンプレート・データを含む場合に定義できます。許可なしは、いずれの 1 次照会テーブルを使用するかにかかわらず、複合照会テーブルで定義できます。

フィルター

フィルターを使用して、複合照会テーブルに含めるオブジェクトまたは行の数を制限します。

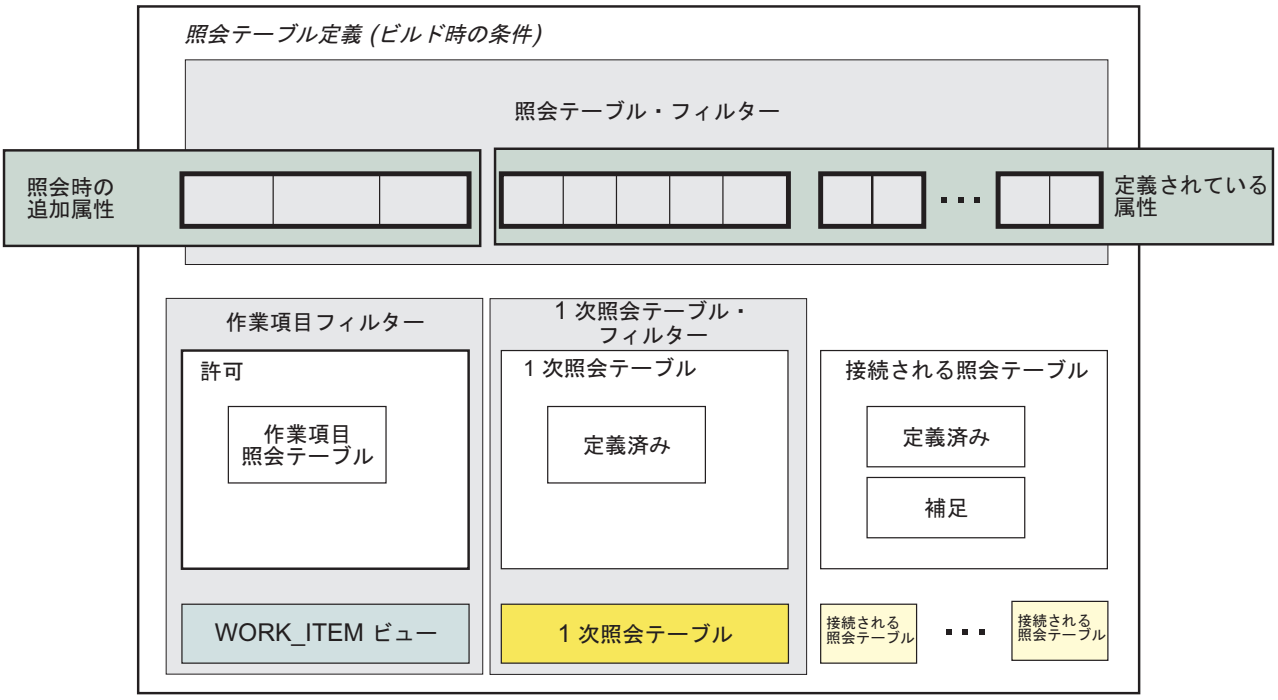


図 7. 複合照会テーブルのフィルター

複合照会テーブルのフィルターは、以下の開発時に定義できます。

- 1 次照会テーブル (1 次照会テーブル・フィルターとして)。
- 暗黙的に使用可能な WORK_ITEM 照会テーブル。これは、1 次照会テーブルにインスタンス・データが含まれている場合に、許可を実行します。このフィルターは、許可フィルターと呼ばれ、インスタンス・ベースの許可を使用するように複合照会テーブルが構成されている場合にのみ使用可能です。
- 複合照会テーブル (照会テーブル・フィルターとして)。

フィルターは、照会テーブルの作成時に定義されます。例えば、1 次照会テーブル TASK が含まれている複合照会テーブルは、作動可能状態 (1 次照会テーブル・フィルターとして "STATE=STATE_READY" が指定されている) のタスクに対してフィルター処理を行うことができます。

許可

1 次照会テーブルが含まれている複合照会テーブルの内容にアクセスするための許可は、1 次照会テーブルにアクセスするために使用される許可と似ています。複合照会テーブルは、より多くの制限を設けて構成できる点が異なります。

- インスタンス・ベースの許可を使用するように構成する場合は、複合照会テーブルに含まれているデータを対象として、WORK_ITEM 照会テーブルに既存の作業項目が存在するかどうかが検査されます。この検査は、1 次照会テーブルに対して行われます。複合照会テーブルの構成に応じて、全員、個人、グループ、および継承された作業項目が検査で使用されます。継承された作業項目を指定する場合は、関連する全員、個人、またはグループの作業項目が構成されている、プロセス・インスタンスを親として持つオブジェクト (参加ヒューマン・タスクなど) が複合照会テーブルに入れられます。通常、継承された作業項目は管理者の場合にのみ有用です。

- テンプレート・データを含む 1 次照会テーブルが含まれている複合照会テーブルは、インスタンス・ベースの許可を使用するように設定してはなりません。ロール・ベースの許可が使用されている場合、照会を実行できるのは BPESystemAdministrator Java EE ロール (Business Flow Manager EJB を使用している場合) または TaskSystemAdministrator Java EE ロール (Human Task Manager EJB を使用している場合) のユーザーのみで、AdminAuthorizationOptions オブジェクトを使用する必要があります。

関連概念

472 ページの『定義済み照会テーブル』
定義済み照会テーブルは、Business Process Choreographer データベース内のデータへのアクセスを提供します。定義済み照会テーブルは、対応する定義済み Business Process Choreographer データベース・ビュー (TASK ビューまたは PROCESS_INSTANCE ビューなど) を照会テーブルの形式で表現したものです。これらの定義済み照会テーブルは、プロセスおよびタスク・リストの照会を実行するために最適化されているため、定義済みデータベース・ビューよりも機能とパフォーマンスが強化されています。

475 ページの『補足照会テーブル』
Business Process Choreographer の補足照会テーブルは、Business Process Choreographer の管理対象ではないビジネス・データを照会テーブル API に対して公開します。補足照会テーブルを使用すると、ビジネス・プロセス・インスタンス情報またはヒューマン・タスク情報の取得時に、この外部データを定義済み照会テーブルのデータと一緒に使用することができます。

照会テーブルの作成

Business Process Choreographer の補足照会テーブルおよび複合照会テーブルは、Query Table Builder を使用してアプリケーションの開発中に作成します。定義済み照会テーブルは、作成することもデプロイすることもできません。定義済み照会テーブルは、 Business Process Choreographer のインストール時に利用できる照会テーブルで、Business Process Choreographer データベース・スキーマでの成果物を簡略表示することができます。

Query Table Builder は Eclipse プラグインとして使用可能であり、WebSphere Business Process Management SupportPacs のサイトからダウンロードできます。このサイトで、PA71 WebSphere Process Server - Query Table Builder を探します。リンクにアクセスするには、このトピックの関連参照のセクションを参照してください。

照会テーブルは、アプリケーションを開発およびデプロイする方法に影響を与えます。以下のステップでは、照会テーブルを使用する Business Process Choreographer アプリケーションを設計および開発する場合に関係するロールについて説明します。

表 37. 照会テーブルの作成ステップ

ステップ	担当者	説明
1. 分析	ビジネス・アナリスト、クライアント開発者	クライアント・アプリケーションで必要な照会テーブルを分析します。考慮すべき質問は、以下のとおりです。 • どれほどの数のタスク・リストまたはプロセス・リストをユーザーに提供しますか? 同じ照会テーブルを共用できるタスク・リストまたはプロセス・リストはありますか? • 使用する許可の種類は何ですか? インスタンス・ベースの許可、ロール・ベースの許可、または許可を使用しませんか? • 再利用可能なその他の照会テーブルが既にシステムで定義されていますか? • 照会テーブルで複数の言語のコンテンツを提供する必要がありますか? その場合、接続される照会テーブルの選択基準は LOCALE=$LOCALE でなければなりません。
2. 照会テーブルの作成	クライアント開発者、ビジネス・アナリスト	クライアント・アプリケーションで使用する照会テーブルを作成します。照会テーブルの照会で最善のパフォーマンスを引き出せるように、照会テーブルの定義を指定します。
3. 照会テーブルのデプロイメント	管理者	照会テーブルを使用するには、事前に照会テーブルをランタイムにデプロイしておく必要があります。このステップは、manageQueryTable.py wsadmin コマンドを使用して実行します。
4. 照会テーブルの照会	クライアント開発者	照会テーブル作成の最後のステップとして、照会テーブルに対して照会を実行します。クライアント開発者は、照会テーブルとその属性の名前を知っている必要があります。

照会テーブル API を使用して照会テーブルを照会するコードの例を以下に示します。簡単にするため、例 1 および 2 では定義済み照会テーブル TASK を照会します。例 3 および 4 では、複合照会テーブルを照会します。ここで、この複合照会テーブルはシステムにデプロイされているとします。アプリケーションの開発時には、定義済み照会テーブルを直接照会するのではなく、複合照会テーブルを使用してください。

例 1

```
// get the naming context and lookup the Business
// Flow Manager Enterprise JavaBeans home; note that the Business Flow
// Manager Enterprise JavaBeans home should be cached for performance
// reasons; also, it is assumed that there's an Enterprise JavaBeans
// reference to the local business flow manager Enterprise JavaBeans
Context ctx = new InitialContext();
LocalBusinessFlowManagerHome home =
 (LocalBusinessFlowManagerHome)
 ctx.lookup("java:comp/env/ejb/BFM");
```

```
// if the human task manager Enterprise JavaBeans is used, do:
// LocalHumanTaskManagerHome home =
//   (LocalHumanTaskManagerHome) ctx.lookup("java:comp/env/ejb/HTM");
// assuming that a EJB reference to the human task manager EJB
// has been defined

// create the business flow manager client-side stub
LocalBusinessFlowManager bfm = home.create();
// if the human task manager EJB is used, do:
// LocalHumanTaskManager htm = home.create();
// note that the human task manager Enterprise JavaBeans provides the
// same methods as the business flow manager Enterprise JavaBeans
// ************************************************
// ****************** example 1 ******************
// ************************************************

// execute a query against the TASK predefined query
// table; this relates to a simple My ToDo's task list
EntityResultSet ers = null;
ers = bfm.queryEntities("TASK", null, null, null);

// print the result to STDOUT
EntityInfo entityInfo = ers.getEntityInfo();
List attList = entityInfo.getAttributeInfo();
int attSize = attList.size();

Iterator iter = ers.getEntities().iterator();
while (iter.hasNext()) {
 System.out.print("Entity: ");
 Entity entity = (Entity) iter.next();
 for (int i = attSize - 1; i >= 0; i--) {
  AttributeInfo ai = (AttributeInfo) attList.get(i);
  System.out.print(
    entity.getAttributeValue(ai.getName()));
 }
 System.out.println();
}
```

例 2

```
// ************************************************
// ****************** example 2 ******************
// ************************************************

// same example as example 1, but using the row-based
// query approach
RowResultSet rrs = null;
rrs = bfm.queryRows("TASK", null, null, null);

attList = rrs.getAttributeInfo();
attSize = attList.size();

// print the result to STDOUT
while (rrs.next()) {
 System.out.print("Row: ");
 for (int i = attSize - 1; i >= 0; i--) {
  AttributeInfo ai = (AttributeInfo) attList.get(i);
  System.out.print(
    rrs.getAttributeValue(ai.getName()));
 }
 System.out.println();
}
```

例 3

```
// ************************************************
// ****************** example 3 *******************
// ************************************************

// execute a query against a composite query table
// that has been deployed on the system before;
// the name is assumed to be COMPANY.TASK_LIST
ers = bfm.queryEntities(
  "COMPANY.TASK_LIST", null, null, null);
^
//  print the result to STDOUT ...
```

例 4

```
// ************************************************
// ****************** example 4 *******************
// ************************************************

// query against the same query table as in example 3,
// but with customized options
FilterOptions fo = new FilterOptions();

// return only objects which are in state ready
fo.setQueryCondition("STATE=STATE_READY");

// sort by the id of the object
fo.setSortAttributes("ID");

// limit the number of entities to 50
fo.setThreshold(50);

// only get a sub-set of the defined attributes
// on the query table
fo.setSelectedAttributes("ID, STATE, DESCRIPTION");

AuthorizationOptions ao = new AuthorizationOptions();

// do not return objects that everybody is allowed
// to see
ao.setEverybodyUsed(Boolean.FALSE);

ers = bfm.queryEntities(
  "COMPANY.TASK_LIST", fo, ao, null);

//  print the result to STDOUT ...
```

関連概念

505 ページの『照会テーブルの照会』
照会は、照会テーブル API (Business Flow Manager EJB でのみ使用可能) を使用して、Business Process Choreographer 内の照会テーブルに対して実行されます。

『照会テーブルのフィルターおよび選択基準』
フィルターおよび選択基準は、Query Table Builder を使用した照会テーブルの作成時に定義します (SQL WHERE 文節に似た構文を使用します)。これらの明確に定義されたフィルターおよび選択基準を使用して、照会テーブルの属性に基づく条件を指定します。

472 ページの『定義済み照会テーブル』
定義済み照会テーブルは、Business Process Choreographer データベース内のデータへのアクセスを提供します。定義済み照会テーブルは、対応する定義済み Business Process Choreographer データベース・ビュー (TASK ビューまたは PROCESS_INSTANCE ビューなど) を照会テーブルの形式で表現したものです。これらの定義済み照会テーブルは、プロセスおよびタスク・リストの照会を実行するために最適化されているため、定義済みデータベース・ビューよりも機能とパフォーマンスが強化されています。

475 ページの『補足照会テーブル』
Business Process Choreographer の補足照会テーブルは、Business Process Choreographer の管理対象ではないビジネス・データを照会テーブル API に対して公開します。補足照会テーブルを使用すると、ビジネス・プロセス・インスタンス情報またはヒューマン・タスク情報の取得時に、この外部データを定義済み照会テーブルのデータと一緒に使用することができます。

477 ページの『複合照会テーブル』
Business Process Choreographer の複合照会テーブルでは、データベース内のデータが固有の方法で表現されることはありません。このテーブルは、関連した定義済み照会テーブルおよび補足照会テーブルのデータの組み合わせからなります。複合照会テーブルを使用して、プロセス・インスタンス・リストまたはタスク・リスト (ユーザーの予定など) の情報を取得します。

照会テーブルのフィルターおよび選択基準

フィルターおよび選択基準は、Query Table Builder を使用した照会テーブルの作成時に定義します (SQL WHERE 文節に似た構文を使用します)。これらの明確に定義されたフィルターおよび選択基準を使用して、照会テーブルの属性に基づく条件を指定します。

Query Table Builder のインストール方法について詳しくは、WebSphere Business Process Management SupportPacs のサイトを参照してください。このサイトで、PA71 WebSphere Process Server - Query Table Builder を探します。リンクにアクセスするには、このトピックの関連参照のセクションを参照してください。

属性

式で使用される属性は、照会テーブルの属性を参照します。使用可能な属性は、式を使用する場所によって異なります。クライアント開発者は、照会テーブル API に

渡される照会フィルターでのみ式を使用できます。複合照会テーブルの作成者は、他のさまざまな場所で式を使用できます。以下の表は、さまざまな場所で使用可能な属性を示しています。

表 38. 照会テーブルの式で使用できる属性

式を使用する場所	式	使用可能な属性
照会テーブル API	照会フィルター	• 照会テーブルに定義されているすべての属性。 • インスタンス・ベースの許可を使用する場合、WORK_ITEM 照会テーブルに定義されているすべての属性 (接頭部は 'WI.')。 例: • STATE=STATE_READY (照会テーブルに STATE 属性が含まれていて、この属性に STATE_READY 定数が定義されている場合)。 • STATE=STATE_READY AND WI.REASON=REASON_POTENTIAL_OWNER (照会テーブルに STATE 属性が含まれていて、照会テーブルでインスタンス・ベースの許可を使用する場合)
複合照会テーブル	照会テーブル・フィルター	
	1 次照会テーブル・フィルター	• 1 次照会テーブルに定義されているすべての属性。 例: • STATE=STATE_READY (照会テーブルに STATE 属性が含まれていて、この属性に STATE_READY 定数が定義されている場合)。
	許可フィルター	• WORK_ITEM 定義済み照会テーブルに定義されているすべての属性 (接頭部は 'WI.')。 例: • WI.REASON=REASON_POTENTIAL_OWNER
	選択基準	• 関連する接続される照会テーブルに定義されているすべての属性。 例: • LOCALE='en_US' (TASK_DESC 照会テーブルなどの接続される照会テーブルに LOCALE 属性が含まれている場合)。

以下の図は、式で使用するフィルターのさまざまな場所および選択基準と、例を示しています。

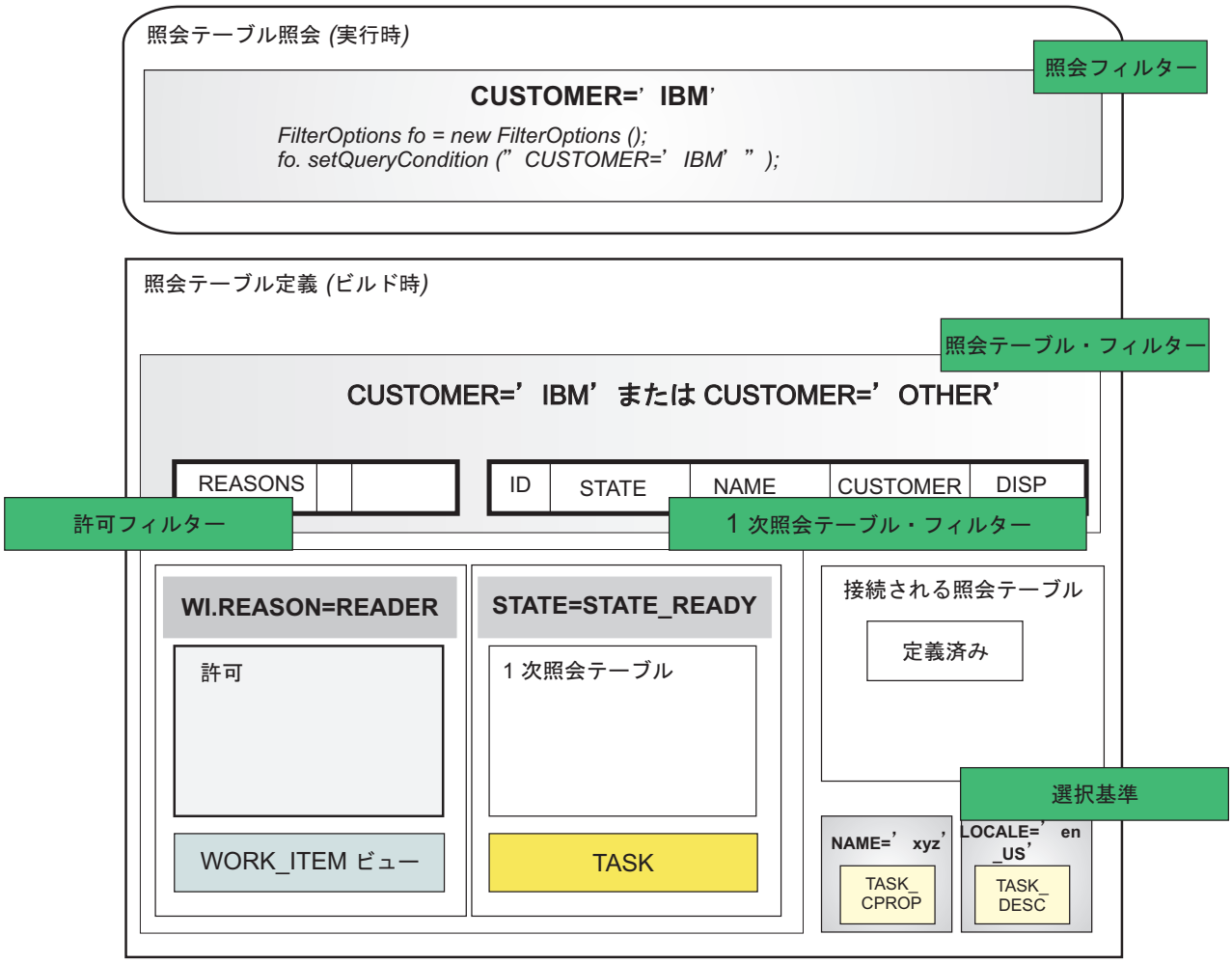


図 8. 式におけるフィルターおよび選択基準

式

式には以下の構文があります。

```
expression  ::=  attribute binary_op value |
                 attribute unary_op |
                 attribute list_op list |
                 (expression) |
                 expression AND expression |
                 expression> OR expression
```

以下のルールが適用されます。

- AND は、OR に優先します。副次式は、AND および OR を使用して結合されます。
- 括弧は式をグループ化するために使用でき、対で使用する必要があります。

例:

- STATE = STATE_READY
- NAME IS NOT NULL
- STATE IN (2, 5, STATE_FINISHED)
- ((PRIORITY=1) OR (WI.REASON=2)) AND (STATE=2)

式は特定のスコープ内で実行されます。このスコープによって、式で使用できる有効な属性が決まります。選択基準 (照会フィルター) は、照会を実行する照会テーブルのスコープ内で実行されます。

以下は、定義済み TASK 照会テーブルに対して実行される照会の例です。

```
'(STATE=STATE_READY AND WI.REASON=REASON_POTENTIAL_OWNER)
OR (WI.REASON=REASON_OWNER)'
```

2 項演算子

以下の 2 項演算子が使用可能です。

binary_op ::= = | < | > | <> | <= | >= | LIKE | NOT LIKE

以下のルールが適用されます。

- 2 項演算子の左側のオペランドは、照会テーブルの属性を参照している必要があります。
- 2 項演算子の右側のオペランドは、リテラル値、定数、またはパラメーターでなければなりません。
- LIKE および NOT LIKE 演算子は、属性タイプが STRING の属性でのみ有効です。
- 左側のオペランドと右側のオペランドは、互換性のある属性タイプでなければなりません。
- ユーザー・パラメーターは、左側の属性の属性タイプとの互換性がなければなりません。

例:

- STATE > 2
- NAME LIKE 'start%'
- STATE <> PARAM(theState)

単項演算子

以下の単項演算子が使用可能です。

unary_op ::= IS NULL | IS NOT NULL

以下のルールが適用されます。

- 単項演算子の左側のオペランドは、照会テーブルの属性を参照している必要があります。有効な属性は、フィルターの場所や選択基準によって異なります。
- ヌル値がないかどうかをすべての属性で確認できます。例えば、CUSTOMER IS NOT NULL と指定します。

例:

```
DESCRIPTION IS NOT NULL
```

リスト演算子

以下のリスト演算子が使用可能です。

list_op ::= IN | NOT IN

以下のルールが適用されます。

- リスト演算子の右側は、ユーザー・パラメーターで置き換えることはできません。
- ユーザー・パラメーターは、右側のオペランドのリスト内で使用できます。

例:

```
STATE IN (STATE_READY, STATE_RUNNING, PARAM(st), 1)
```

リストは、以下のように表現します。

list ::= *value* [, *list*]

以下のルールが適用されます。

- リスト演算子の右側は、ユーザー・パラメーターで置き換えることはできません。
- ユーザー・パラメーターは、右側のオペランドのリスト内で使用できます。

例:

- (2, 5, 8)
- (STATE_READY, STATE_CLAIMED)

値

式で使用する値は、以下のいずれかです。

- **定数**: 定数値。定義済み照会テーブルの属性用に定義されます。例えば、STATE_READY は、TASK 照会テーブルの STATE 属性用に定義されています。
- **リテラル**: ハードコーディングされる任意の値。
- **パラメーター**: パラメーターは、特定の値を使用した照会の実行時に置換されます。

定数 は、定義済み照会テーブルの一部の属性で使用できます。定義済み照会テーブルの属性で使用可能な定数については、定義済みビューの情報を参照してください。照会テーブルでは、整数値を定義する定数のみが公開されます。定数の代わりに、関連リテラル値またはパラメーターを使用することもできます。

例:

- TASK 照会テーブルの STATE 属性の STATE_READY をフィルターで使用して、タスクが作動可能状態になっているかどうかを確認できます。
- WORK_ITEM 照会テーブルの REASON 属性の REASON_POTENTIAL_OWNER をフィルターで使用して、照会テーブルに対して照会を実行するユーザーが潜在的所有者であるかどうかを確認できます。
- TASK 照会テーブルに対して照会を実行する場合、照会フィルター STATE=STATE_READY は STATE=2 と同じです。

式では**リテラル**を使用することもできます。タイム・スタンプおよび ID には、特殊な構文を使用する必要があります。

例:

- STATE=1
- NAME='theName'

- `CREATED > TS ('2008-11-26 T12:00:00')`
- `TKTID=ID('_TKT:801a011e.9d57c52.ab886df6.1fcc0000')`

式で使用する**パラメーター**は、複合照会テーブルの動的性に対応しています。ユーザー・パラメーターとシステム・パラメーターがあります。

- ユーザー・パラメーターは、PARAM (*name*) を使用して指定します。このパラメーターは、照会の実行時に指定する必要があります。com.ibm.bpe.api.Parameter クラスのインスタンスとして、照会テーブル API に渡されます。
- システム・パラメーターは、照会の実行時に指定しなくても、照会テーブル・ランタイムによって指定されるパラメーターです。システム・パラメーター `$USER` および `$LOCALE` が使用可能です。
  - `$USER` (ストリング) には、照会を実行するユーザーの値が入ります。
  - `$LOCALE` (ストリング) には、照会の実行時に使用されるロケールの値が入ります。`$LOCALE` の値は、例えば `'en_US'` などです。

特定のロケールを選択する接続される照会テーブルの選択基準にパラメーターを指定できます。例えば、複合照会テーブルの 1 次照会テーブルが TASK で、接続される照会テーブルが TASK_DESC の場合などです。パラメーターの例を以下に示します。

- `STATE=PARAM(theState)`
- `LOCALE=$LOCALE`
- `OWNER=$USER`

関連概念

485 ページの『照会テーブルの作成』
Business Process Choreographer の補足照会テーブルおよび複合照会テーブルは、Query Table Builder を使用してアプリケーションの開発中に作成します。定義済み照会テーブルは、作成することもデプロイすることもできません。定義済み照会テーブルは、 Business Process Choreographer のインストール時に利用できる照会テーブルで、Business Process Choreographer データベース・スキーマでの成果物を簡略表示することができます。

505 ページの『照会テーブルの照会』
照会は、照会テーブル API (Business Flow Manager EJB でのみ使用可能) を使用して、Business Process Choreographer 内の照会テーブルに対して実行されます。

関連タスク

526 ページの『Business Process Choreographer Explorer の照会テーブルの作成』
EJB 照会 API の代わりに照会テーブルを使用すると、Business Process Choreographer Explorer のパフォーマンスを向上させることができます。照会テーブルを作成するには、Query Table Builder を使用します。

照会テーブルの許可

照会テーブルで照会を実行するときには、インスタンス・ベースの許可、ロール・ベースの許可、または許可なしを使用できます。

許可タイプは、照会テーブルで定義されています。

- インスタンス・ベースの許可は、作業項目を使用して照会テーブルのオブジェクトを許可することを示します。これを行うには、適切な作業項目が存在するかどうかを確認してください。
- ロール・ベースの許可は、Java EE ロールに基づいています。これは、照会テーブルの内容を参照するために、API 照会メソッドを使用する呼び出し元が、BPESystemAdministrator Java EE ロール (Business Flow Manager EJB を使用している場合) または TaskSystemAdministrator Java EE ロール (Human Task Manager EJB を使用している場合) でなければならないことを示します。これは、テンプレート・データを含む定義済み照会テーブルおよびテンプレート・データを含む 1 次照会テーブルが含まれている複合照会テーブルの場合に使用可能です。これらの照会テーブルのオブジェクトには、関連作業項目がありません。
- 許可を指定していない場合は、フィルターを適用した後、すべての認証済みユーザーが照会テーブルのすべての内容を参照できます。

定義済み照会テーブルでの許可のタイプおよび複合照会テーブルと補足照会テーブルで構成可能な許可のタイプの概要を以下の表に示します。

表 39. 照会テーブルに応じた許可のタイプ

照会テーブル	インスタンス・ベースの許可	ロール・ベースの許可	許可なし
定義済み	インスタンス・データを含む定義済み照会テーブルの場合に必要です。	テンプレート・データを含む定義済み照会テーブルの場合に必要です。	N/A
複合	オフにすることができます。オフにすると、許可は使用されず、セキュリティー制約がオーバーライドされます。つまり、すべての認証済みユーザーは、それぞれのオブジェクトに対する権限を持っているかどうかに関係なく、照会テーブルを使用してデータを検索することができます。 テンプレート・データを含む 1 次照会テーブルが含まれている複合照会テーブルは、インスタンス・ベースの許可を使用するように設定してはなりません。	テンプレート・データを含む 1 次照会テーブルが含まれている複合照会テーブルの場合などは、オフにすることができます。その場合、許可は使用されず、セキュリティー制約がオーバーライドされます。つまり、すべての認証済みユーザーは、それぞれのオブジェクトに対する権限を持っているかどうかに関係なく、照会テーブルを使用してデータを検索することができます。 インスタンス・データを含む 1 次照会テーブルが含まれている複合照会テーブルは、ロール・ベースの許可を使用するように設定してはなりません。	フィルターを適用した後、すべての認証済みユーザーが照会テーブルのすべての内容を参照できます。
補足照会テーブル	補足照会テーブルは、Business Process Choreographer の管理対象ではなく、したがってこれらのテーブルの内容に対する許可情報がないため、インスタンス・ベースの許可を使用するように設定するべきではありません。	補足照会テーブルは、ロール・ベースの許可を使用するように設定してはなりません。	フィルターを適用した後、すべての認証済みユーザーが照会テーブルのすべての内容を参照できます。

照会テーブルのタイプに応じて使用可能な許可タイプのオプションの概要を以下の図に示します。また、さまざまな動作、照会テーブル API、およびその API の許可オプションも示しています。

図 9. 照会テーブルのインスタンス・ベースの許可

*) onBehalfUser を設定する場合、(A) が適用されます

作業項目を使用する照会結果内のオブジェクトに対するインスタンス・ベースの許可は、照会テーブル API に渡される許可パラメーターや、照会テーブルのインスタンス・ベースの許可フラグの設定によって決まります。

- (A) AuthorizationOptions オブジェクトを使用する定義済み照会テーブルまたは複合照会テーブルに対する照会では、この特定ユーザーの関連作業項目と相関する

エンティティーが返されます。AdminAuthorizationOptions オブジェクトを使用し、onBehalfUser を設定する場合も同様です。通常、タスク・リストまたはプロセス・リストがユーザーに提供される標準クライアントでは、照会テーブルと照会テーブル API パラメーターのこの組み合わせが使用されます。

- (B) 照会テーブルの内容全体は、照会テーブルのインスタンス・ベースの許可を使用して構成されるように、関連作業項目を持つエンティティーで構成されます。インスタンス・ベースの許可では、4 つの作業項目 (全員、個人、グループ、および継承) が考慮されます。 API 照会メソッドを使用する呼び出し元は、BPESystemAdministrator Java EE ロール (Business Flow Manager EJB を使用している場合) または TaskSystemAdministrator Java EE ロール (Human Task Manager EJB を使用している場合) でなければなりません。照会テーブルと照会テーブル API パラメーターのこの組み合わせは、使用可能なタスクまたはプロセスの完全なリストが表示または検索される管理シナリオ向けです。
- (C) AdminAuthorizationOptions または AuthorizationOptions が照会テーブル API に渡される場合、インスタンス・ベースの許可もロール・ベースの許可も使用しない照会テーブルに対する照会は、すべて同じ結果を返します。これは、補足照会テーブルおよび複合照会テーブルに当てはまります。作業項目または Java EE ロールに対する検査は行われないため、すべての認証済みユーザーに対して内容全体が表示されます。Business Process Choreographer で提供されるインスタンス・ベースまたはロール・ベースの許可の制約を適用することによってオブジェクトの可視性を制限したくないクライアントは、照会テーブル定義の作成時に許可検査をオフにすることができます。ただし、要求と実行を使用する場合は、ユーザーは関連作業項目を持っている必要があります。
- (D) ロール・ベースの許可が構成されている定義済み照会テーブルまたは複合照会テーブルのテンプレート・データには、ロール・ベースの許可を使用しなければアクセスできません。そのため、API 照会メソッドを使用する呼び出し元は、BPESystemAdministrator Java EE ロール (Business Flow Manager EJB を使用している場合) または TaskSystemAdministrator Java EE ロール (Human Task Manager EJB を使用している場合) でなければなりません。照会 API ではなく、照会テーブル API を使用して、テンプレート情報にアクセスできます。

作業項目およびインスタンス・ベースの許可

Business Process Choreographer で提供されるインスタンス・ベースの許可は、作業項目に基づいています。各作業項目では、誰が、どのオブジェクトに対して、どの権限を持っているかが記述されます。インスタンス・ベースの許可を使用する場合、WORK_ITEM 照会テーブルを使用して、この情報にアクセスします。

以下の表は、照会テーブルに対する照会の実行時にインスタンス・ベースの許可を使用する場合に考慮されるさまざまなタイプの作業項目について説明しています。

表 40. 作業項目タイプ

作業項目タイプ	説明
everybody	すべてのユーザーが特定のオブジェクト (タスクまたはプロセス・インスタンスなど) にアクセスできます。この場合、関連作業項目の EVERYBODY 属性は TRUE に設定されます。

表 40. 作業項目タイプ (続き)

作業項目タイプ	説明
individual	特定のユーザーのために作成される作業項目。関連作業項目の OWNER_ID 属性は、特定のユーザーに設定されます。1 つのオブジェクト (タスクなど) で、OWNER_ID 属性が異なる複数の作業項目が存在する場合があります。
group	特定のグループのユーザーのために作成される作業項目。関連作業項目の GROUP_NAME 属性は、特定のグループに設定されます。
inherited	プロセス・インスタンスの読者および管理者も、これらのプロセス・インスタンスに属するヒューマン・タスクへのアクセスをエスカレーションも含めて継承することができます。タスク照会内に継承された作業項目があるかどうかの検査は、実行時に複雑な SQL 結合と共に実行されるため、パフォーマンスに影響が及びます。

作業項目は、さまざまな状況で Business Process Choreographer によって作成されます。例えば、関連した担当者割り当て基準が指定されている場合は、タスク作成時に、さまざまなロール (リーダーや潜在的所有者など) の作業項目が作成されます。

以下の表は、照会テーブルに対する照会の実行時にインスタンス・ベースの許可を使用する場合に、定義済みの担当者割り当て基準に従って作成される作業項目のタイプを示しています。継承された作業項目は、プロセス・アプリケーションの開発時に明示的にモデル化されていない関係を反映するため、表に表示されていません。

表 41. 作業項目および担当者割り当て基準

作業項目タイプ	関連した担当者割り当て基準
everybody	全員
individual	*Nobody*、*Everybody*、および *Group* verb 以外のすべての担当者割り当て基準
group	Group

複合照会テーブルの許可フィルター

インスタンス・ベースの許可を使用する場合、複合照会テーブルに対して許可フィルターを指定することができます。このフィルターは、許可に使用される作業項目を作業項目の特定の属性に基づいて制限します。例えば、TASK 1 次照会テーブルを含む複合照会テーブルに対する許可フィルター`"WI.REASON=REASON_POTENTIAL_OWNER"` は、ユーザーが照会を実行するときに返されるタスクを制限します。結果には、そのユーザーの予定を表すタスクのみが含まれます。つまり、結果は、そのユーザーが要求する権限を持つタスクに制限されるということです。このフィルターは、照会テーブル・フィルターまたは照会フィルターとして指定することもできますが、照会のパフォーマンス上の理由で、それらのフィルターを許可フィルターとして指定することをお勧めします。

関連概念

472 ページの『定義済み照会テーブル』
定義済み照会テーブルは、Business Process Choreographer データベース内のデータへのアクセスを提供します。定義済み照会テーブルは、対応する定義済み Business Process Choreographer データベース・ビュー (TASK ビューまたは PROCESS_INSTANCE ビューなど) を照会テーブルの形式で表現したものです。これらの定義済み照会テーブルは、プロセスおよびタスク・リストの照会を実行するために最適化されているため、定義済みデータベース・ビューよりも機能とパフォーマンスが強化されています。

475 ページの『補足照会テーブル』
Business Process Choreographer の補足照会テーブルは、Business Process Choreographer の管理対象ではないビジネス・データを照会テーブル API に対して公開します。補足照会テーブルを使用すると、ビジネス・プロセス・インスタンス情報またはヒューマン・タスク情報の取得時に、この外部データを定義済み照会テーブルのデータと一緒に使用することができます。

477 ページの『複合照会テーブル』
Business Process Choreographer の複合照会テーブルでは、データベース内のデータが固有の方法で表現されることはありません。このテーブルは、関連した定義済み照会テーブルおよび補足照会テーブルのデータの組み合わせからなります。複合照会テーブルを使用して、プロセス・インスタンス・リストまたはタスク・リスト (ユーザーの予定など) の情報を取得します。

511 ページの『照会テーブル API の許可オプション』
Business Process Choreographer の照会テーブルに対して照会を実行する場合、許可オプションは入力パラメーターとして照会テーブル API のメソッドに渡すことができます。

関連タスク

526 ページの『Business Process Choreographer Explorer の照会テーブルの作成』
EJB 照会 API の代わりに照会テーブルを使用すると、Business Process Choreographer Explorer のパフォーマンスを向上させることができます。照会テーブルを作成するには、Query Table Builder を使用します。

照会テーブルの属性タイプ

属性タイプは、Business Process Choreographer で照会テーブルを定義する場合、照会でリテラル値を使用する場合、および照会結果の値にアクセスする場合に必要です。属性タイプごとに、ルールおよびマッピングを使用できます。

Java プログラミング言語およびデータベースで使用可能なタイプのサブセットを使用して、照会テーブルの属性のタイプを定義します。属性タイプは、具象 Java タイプまたはデータベース・タイプを抽象化したものです。補足照会テーブルでは、データベース・タイプから属性タイプへの有効なマッピングを使用する必要があります。

以下の表は、属性タイプを示しています。

表 42. 属性タイプ

属性タイプ	説明
ID	ヒューマン・タスク (TKIID)、プロセス・インスタンス (PIID)、または他のオブジェクトを識別するために使用する ID。例えば、ID は、指定された TKIID で識別される特定のヒューマン・タスクを要求または完了するために使用されます。
STRING	タスクの説明または照会プロパティーをストリングとして表現できます。
NUMBER	数値は、タスクの優先順位などの属性に使用されます。
TIMESTAMP	タイム・スタンプは、ヒューマン・タスクが作成された時刻や、プロセス・インスタンスが終了した時刻などの特定の時刻を表します。
DECIMAL	10 進数は、XSD の倍精度浮動小数点タイプの変数を使用して照会プロパティーを定義する場合などに、照会プロパティーのタイプとして使用できます。
BOOLEAN	ブール値は、2 つの値 true か false のいずれかです。例えば、ヒューマン・タスクには属性 autoClaim があります。この属性は、このタスクの潜在的所有者としてのユーザーが 1 人のみの場合に、タスクを自動的に要求するかどうかを識別します。

データベース・タイプから属性タイプへのマッピング

属性タイプは、Business Process Choreographer で照会テーブルを定義する場合、照会テーブルに対して照会を実行する場合、および照会結果の値にアクセスする場合に使用します。

以下の表は、データベース・タイプおよび属性タイプへのマッピングを示しています。

表 43. データベース・タイプから属性タイプへのマッピング

データベース・タイプ	属性タイプ
16 バイトのバイナリー・タイプ。これは、Business Process Choreographer テーブルの TASK の TKIID などの ID に使用されるタイプです。	ID
文字ベースのタイプ。長さは、照会テーブルの属性によって参照されるデータベース表の列によって異なります。	STRING
整数データベース・タイプ (整数、短整数、長整数など)。	NUMBER
タイム・スタンプ・データベース・タイプ。	TIMESTAMP
10 進数タイプ (浮動小数点、倍精度浮動小数点など)。	DECIMAL
ブール値に変換可能なタイプ (数値など)。1 は *true*、他のすべての数値は *false* と解釈されます。	BOOLEAN

通常、補足照会テーブルは、テーブルまたはビューの作成が不要な既存のデータベース表およびビューを参照します。

例

Business Process Choreographer で補足照会テーブルとして表わされるテーブル CUSTOM.ADDITIONAL_INFO が DB2 環境にあるとします。以下の SQL ステートメントは、このデータベース表を作成します。

```
CREATE TABLE CUSTOM.ADDITIONAL_INFO
(
  PIID       CHAR(16) FOR BIT DATA,
  INFO       VARCHAR(220),
  COUNT      INTEGER
);
```

データベース列タイプから照会テーブル属性タイプへの以下のマッピングは、CUSTOM.ADDITIONAL_INFO テーブルの補足照会テーブルで使用されます。

表 44. データベース・タイプから属性タイプへのマッピングの例

データベース列およびタイプ	照会テーブルの属性およびタイプ
PIID CHAR(16) FOR BIT DATA	PIID (ID)
INFO VARCHAR(220)	INFO (STRING)
COUNT INTEGER	COUNT (NUMBER)

属性タイプからリテラル表記へのマッピング

属性タイプは、Business Process Choreographer で照会テーブルを定義する場合、照会テーブルに対して照会を実行する場合、および照会結果の値にアクセスする場合に使用します。このトピックでは、属性タイプからリテラル表記へのマッピングについて説明します。

リテラル値は、式で使用し、複合照会テーブルのフィルターや照会テーブル API に渡されるフィルターのフィルター基準および選択基準を定義することができます。

以下の表は、属性タイプおよびリテラル値へのマッピングを示しています。プレースホルダーは、*イタリック* で示されています。属性タイプ ID および TIMESTAMP (照会テーブル API に渡せます) では特別な構文を使用することに注意してください。この構文は、照会 API でも使用します。

表 45. 属性タイプからリテラル値へのマッピング

属性タイプ	式におけるリテラル値としての構文および使用法
ID	`ID ('ID のストリング表記' )`
	クライアント・アプリケーションの開発時に、ID はストリングまたは com.ibm.bpe.api.OID インターフェースのインスタンスとして表記されます。ストリング表記は、toString メソッドを使用して、com.ibm.bpe.api.OID インターフェースのインスタンスから取得できます。ストリングは、単一引用符で囲む必要があります。
STRING	`'the string'`
	ストリングは引用符で囲む必要があります。

表 45. 属性タイプからリテラル値へのマッピング (続き)

属性タイプ	式におけるリテラル値としての構文および使用法
NUMBER	number
	引用符なしのテキストとしての数値。定義済み照会テーブルの一部の数値属性には定数が定義されており、その定数を使用できます。
TIMESTAMP	TS ('YYYY-MM-DDThh:mm:ss')
	タイム・スタンプは、以下の形式で指定する必要があります。 • YYYY 4 桁の年 • MM 2 桁の月 • DD 2 桁の日 • hh 2 桁の時刻 (24 時間表記) • mm 2 桁の分 • ss 2 桁の秒。タイム・スタンプは、ユーザーの時間帯の定義に従って解釈されます。
DECIMAL	number.fraction
	引用符なしのテキストとしての 10 進数。.fraction 部分はオプションです。
BOOLEAN	true、false
	テキストとしてのブール値。

例

- filterOptions.setQueryCondition("STATE=2");
- filterOptions.setQueryCondition("STATE=STATE_READY");
- 接続される照会テーブル TASK_DESC の選択基準: "LOCALE='en_US'"
- filterOptions.setQueryCondition("PTID=ID ('_PT:8001011e.1dee8e51.247d6df6.29a60000')");

属性タイプからパラメーターへのマッピング

属性タイプは、Business Process Choreographer で照会テーブルを定義する場合、照会テーブルに対して照会を実行する場合、および照会結果の値にアクセスする場合に使用します。

以下の表は、属性タイプとパラメーター値へのマッピングを示しています。これらのパラメーター値は、式で使用し、複合照会テーブルのフィルターや照会テーブル API に渡されるフィルターのフィルター基準および選択基準を定義することができます。

表 46. 属性タイプからユーザー・パラメーター値へのマッピング

属性タイプ	式におけるパラメーター値としての使用法
ID	PARAM*(name)* クライアント・アプリケーションの開発時に、ID はストリングまたは com.ibm.bpe.api.OID インターフェースのインスタンスとして表記されます。 パラメーターとして、これらの表記は両方とも有効です。有効な OID を反映するバイトの配列を使用することもできます (バイト)。
STRING	PARAM*(name)* 実行時に toString メソッドによって照会テーブル API に渡されるオブジェクトのストリング表記。
NUMBER	PARAM*(name)* 数値の java.lang.Long、java.lang.Integer、java.lang.Short、または java.lang.String 表記が、照会テーブル API に渡されます。定数の名前 (定義済み照会テーブルの一部の属性で定義されています) を渡すこともできます。
TIMESTAMP	PARAM*(name)* 有効な表記は、以下のとおりです。 • タイム・スタンプの java.lang.String 表記 • com.ibm.bpe.api.UTCDate のインスタンス • java.util.Calendar のインスタンス
DECIMAL	PARAM*(name)* 10 進数の java.lang.Long、java.lang.Integer、java.lang.Short、java.lang.Double、java.lang.Float、または java.lang.String 表記が、照会テーブル API に渡されます。
BOOLEAN	PARAM*(name)* 有効な値は以下のとおりです。 • ブール値の java.lang.String 表記 • 適切な値 (false の場合は 0、true の場合は 1) を持つ java.lang.Short、java.lang.Integer、java.lang.Long。 • java.lang.Boolean オブジェクト

例

```
...
// this example shows a query against a composite query table
// COMP.TASKS with a parameter "customer"
java.util.List params = new java.util.ArrayList();

list.add(new com.ibm.bpe.api.Parameter("customer", "IBM");
// the business flow manager Enterprise JavaBeans or the
// human task manager Enterprise JavaBeans can be used to access query tables
service.bfm.queryEntities("COMP.TASKS", null, null, params);
...
```

属性タイプから Java オブジェクト・タイプへのマッピング

属性タイプは、Business Process Choreographer で照会テーブルを定義する場合、照会テーブルに対して照会を実行する場合、および照会結果の値にアクセスする場合に使用します。このトピックでは、属性タイプから Java オブジェクト・タイプへのマッピングについて説明します。

以下の表は、属性タイプおよび照会結果セット内の Java オブジェクト・タイプへのマッピングを示しています。

表 47. 属性タイプから Java オブジェクト・タイプへのマッピング

属性タイプ	関連する Java オブジェクト・タイプ
ID	com.ibm.bpe.api.OID
STRING	java.lang.String
NUMBER	java.lang.Long
TIMESTAMP	java.util.Calendar
DECIMAL	java.lang.Double
BOOLEAN	java.lang.Boolean

例

```
...
// the following example shows a query against a composite query table
// COMP.TA; attribute "STATE" is of attribute type NUMBER
...
// run the query
// the business flow manager Enterprise JavaBeans or the
// human task manager Enterprise JavaBeans can be used to access query tables
EntityResultSet rs = bfm.queryEntities("COMP.TA",null,null,params);

// get the entities and iterate over it
List entities = rs.getEntities();
for (int i = 0 ; i < entities.size(); i++) {

      // work on a particular entity
      Entity en = (Entity) entities.get(i);

      // note that the following code could be written
      // more generalized using the attribute info objects
      // contained in ei.getAttributeInfo()

      // get attribute STATE
      Long state = (Long) en.getAttributeValue("STATE");
      ...
}
...
```

属性タイプの互換性

属性タイプは、Business Process Choreographer で照会テーブルを定義する場合、照会テーブルに対して照会を実行する場合、および照会結果の値にアクセスする場合に使用します。

以下の表は、属性タイプおよび互換性のある属性タイプを示しています。これらの属性タイプは、照会テーブル内でフィルターおよび選択基準を定義するために使用できます。互換性のある属性タイプには、**X** のマークが付いています。

表 48. 属性タイプの互換性

属性タイプ	ID	STRING	NUMBER	TIMESTAMP	DECIMAL	BOOLEAN
ID	X					
STRING		X				
NUMBER			X		X	
TIMESTAMP				X		
DECIMAL			X		X	
BOOLEAN						X

フィルターおよび条件を指定する照会テーブル式では、比較する属性または値のタイプに互換性がなければなりません。例えば、`WI.OWNER_ID=1` は無効なフィルターです。理由は、左オペランドが STRING 型であるのに対し、右オペランドが NUMBER 型であるためです。

照会テーブルの照会

照会は、照会テーブル API (Business Flow Manager EJB でのみ使用可能) を使用して、Business Process Choreographer 内の照会テーブルに対して実行されます。

照会は、1 つの照会テーブルでのみ実行されます。エンティティー・ベースの API メソッドおよび行ベースの API メソッドを使用して、照会テーブルから内容を取得します。入力パラメーターは、照会テーブル API のメソッドに渡されます。

関連概念

485 ページの『照会テーブルの作成』
Business Process Choreographer の補足照会テーブルおよび複合照会テーブルは、Query Table Builder を使用してアプリケーションの開発中に作成します。定義済み照会テーブルは、作成することもデプロイすることもできません。定義済み照会テーブルは、 Business Process Choreographer のインストール時に利用できる照会テーブルで、Business Process Choreographer データベース・スキーマでの成果物を簡略表示することができます。

472 ページの『定義済み照会テーブル』
定義済み照会テーブルは、Business Process Choreographer データベース内のデータへのアクセスを提供します。定義済み照会テーブルは、対応する定義済み Business Process Choreographer データベース・ビュー (TASK ビューまたは PROCESS_INSTANCE ビューなど) を照会テーブルの形式で表現したものです。これらの定義済み照会テーブルは、プロセスおよびタスク・リストの照会を実行するために最適化されているため、定義済みデータベース・ビューよりも機能とパフォーマンスが強化されています。

475 ページの『補足照会テーブル』
Business Process Choreographer の補足照会テーブルは、Business Process Choreographer の管理対象ではないビジネス・データを照会テーブル API に対して公開します。補足照会テーブルを使用すると、ビジネス・プロセス・インスタンス情報またはヒューマン・タスク情報の取得時に、この外部データを定義済み照会テーブルのデータと一緒に使用することができます。

477 ページの『複合照会テーブル』
Business Process Choreographer の複合照会テーブルでは、データベース内のデータが固有の方法で表現されることはありません。このテーブルは、関連した定義済み照会テーブルおよび補足照会テーブルのデータの組み合わせからなります。複合照会テーブルを使用して、プロセス・インスタンス・リストまたはタスク・リスト (ユーザーの予定など) の情報を取得します。

489 ページの『照会テーブルのフィルターおよび選択基準』
フィルターおよび選択基準は、Query Table Builder を使用した照会テーブルの作成時に定義します (SQL WHERE 文節に似た構文を使用します)。これらの明確に定義されたフィルターおよび選択基準を使用して、照会テーブルの属性に基づく条件を指定します。

照会テーブル API メソッド

照会は、照会テーブル API を使用して、Business Process Choreographer 内の照会テーブルに対して実行されます。エンティティー・ベースの API メソッドおよび行ベースの API メソッドを使用して、照会テーブルから内容を取得することができます。

照会テーブル API を使用して Business Process Choreographer 内の照会テーブルに対して照会を実行するために、以下のエンティティー・ベースのメソッドおよび行ベースのメソッドが提供されています。

表 49. 照会テーブルに対して実行される照会のメソッド

目的	メソッド
照会の内容	• queryEntities • queryRows これらのメソッドは両方とも、照会テーブルの内容を返します。queryEntities メソッドはエンティティーに基づいて内容を返し、queryRows は行に基づいて内容を返します。
オブジェクト数の照会	• queryEntityCount • queryRowCount これらのメソッドは両方とも、照会テーブル内のオブジェクト数を返しますが、実際の数はエンティティー・ベースのアプローチと行ベースのアプローチのいずれを取るかによって異なります。

queryEntities メソッドと queryEntityCount メソッドを使用するエンティティー・ベースの照会では、1 次照会テーブルの 1 次キーで定義された一意的に識別可能なエンティティーが照会テーブルに含まれていることが前提となっています。

queryRows メソッドと queryRowCount メソッドを使用する行ベースの照会は、JDBC のような結果セットを返します。これは行ベースであり、ナビゲーション用に first および next メソッドが用意されています。照会テーブル API を使用して照会テーブルで照会を実行したときに返された結果セットは、照会 API によって返された QueryResultSet と比較できます。一般に、行の数は、照会テーブルに含まれるエンティティーの数より大きくなります。行の結果セットには、同じエンティティー (例えば、TKIID などのタスク ID で識別されるヒューマン・タスク) が複数回出現する場合があります。

定義済み照会テーブルに含まれている特定のインスタンスは、Business Process Choreographer 環境内に 1 回のみ存在できます。インスタンスの例として、ヒューマン・タスクやビジネス・プロセスがあります。これらのインスタンスは、ID または ID のセットを使用して一意的に識別されます。これは、ヒューマン・タスクのインスタンスの場合は TKIID で、プロセス・インスタンスの場合は PIID です。

複合照会テーブルは、1 つの 1 次照会テーブルと 0 個以上の接続される照会テーブルで構成されます。複合照会テーブルに含まれているオブジェクトは、1 次照会テーブルに含まれているオブジェクトの固有の ID によって一意的に識別されます。複合照会テーブルのエンティティー・タイプは、そこに含まれる 1 次照会テーブルによって決まります。例えば、1 次照会テーブル TASK が含まれる複合照会テーブルには、TASK タイプのエンティティーが含まれます。1 次照会テーブルと接続される照会テーブルの間の 1 対 1 または 1 対 0 の関係により、接続される照会テーブルのエンティティーは重複することがありません。

エンティティー・ベースの照会では、1 次照会テーブルの 1 次キーで定義された、照会テーブルの一意的に識別可能なエンティティーを利用します。通常、ユーザー・インターフェースのクライアント・アプリケーション・プログラマーは、インスタンスが重複しておらず固有になっているかどうかに関心を払います。例えば、

ヒューマン・タスクは、ユーザー・インターフェース上で 1 回のみ表示されているかなどです。エンティティー・ベースの照会テーブル API を使用する場合、固有のインスタンスが返されます。

インスタンス・ベースの許可を使用する場合、行ベースの照会では、返される 1 次照会テーブルの行が重複する場合があります。

- WORK_ITEM 照会テーブルの情報は、照会を使用して取得されます。例えば、照会テーブルで定義されている属性に加えて WI.REASON 属性も取得する場合、複数の行が結果の対象となります。ユーザーがエンティティー (タスクやプロセス・インスタンスなど) にアクセスできる理由が複数存在する場合があるからです。
- インスタンス・ベースの許可を使用して、重複行の削除を指定しません。インスタンス・ベースの許可を使用する場合、作業項目の情報は取得されなくても、複数の行が返される可能性があります。

エンティティー・ベースの照会テーブル API を使用する場合:

- エンティティー・ベースの照会は、常に SQL distinct 演算子を使用して実行されます。
- エンティティー・ベースの照会では、作業項目に関連した情報の配列値を許容する結果が返されます。

照会テーブル API のパラメーター

Business Process Choreographer の照会テーブルに対する照会の実行時に、照会テーブル API メソッドを使用して内容を取得します。

以下の入力パラメーターが、照会テーブル API のメソッドに渡されます。

表 50. 照会テーブル API のパラメーター

パラメーター	オプション	タイプおよび説明
照会テーブル名	いいえ	java.lang.String
		照会テーブルの固有の名前。
フィルター・オプション	はい	com.ibm.bpe.api.FilterOptions (Business Flow Manager Enterprise JavaBeans が使用されている場合) または com.ibm.task.api.FilterOptions (Human Task Manager Enterprise JavaBeans が使用されている場合)。
		照会を定義するために使用できるオプション。例えば、照会しきい値は、返される結果数を制限するために、このパラメーターに設定します。
許可オプション	はい	com.ibm.bpe.api.AuthorizationOptions または com.ibm.bpe.api.AdminAuthorizationOptions (Business Flow Manager Enterprise JavaBeans が使用されている場合)。 com.ibm.task.api.AuthorizationOptions または com.ibm.task.api.AdminAuthorizationOptions (Human Task Manager Enterprise JavaBeans が使用されている場合)。
		インスタンス・ベースの許可を使用する場合は、許可をさらに制約することができます。ロール・ベースの許可が必要な照会テーブルの場合は、AdminAuthorizationOptions のインスタンスを渡す必要があります。

表 50. 照会テーブル API のパラメーター (続き)

パラメーター	オプション	タイプおよび説明
パラメーター	はい	com.ibm.bpe.api.Parameter の java.util.List (Business Flow Manager Enterprise JavaBeans が使用されている場合) または com.ibm.task.api.Parameter (Human Task Manager Enterprise JavaBeans が使用されている場合)。
		このパラメーターは、複合照会テーブルのフィルターまたは選択基準で指定されるユーザー・パラメーターを渡すために使用されます。

照会は、特定の 1 つの照会テーブルでのみ実行されます。複数の照会テーブル間の関係は、複合照会テーブルを使用して定義します。照会 API (照会テーブル API は除外) では、これはデータベース・ビューに対応します。

フィルターおよび選択基準は、Query Table Builder を使用した照会テーブルの作成時に式で指定します。詳しくはインフォメーション・センターで、複合照会テーブルに関するトピックと、照会テーブルのフィルターおよび検索条件に関するトピックを参照してください。Query Table Builder について詳しくは、WebSphere Business Process Management SupportPacs のサイトを参照してください。このサイトで、PA71 WebSphere Process Server - Query Table Builder を探します。リンクにアクセスするには、このトピックの関連参照のセクションを参照してください。

照会テーブル名:

Business Process Choreographer の照会テーブルに対して照会を実行する場合、照会テーブル名は入力パラメーターとして照会テーブル API のメソッドに渡されます。

照会テーブル名は、照会が実行された照会テーブルの名前です。

- 定義済み照会テーブルの場合は、定義済み照会テーブルの名前です。
- 複合照会テーブルおよび補足照会テーブルの場合、これは、照会テーブルのモデル化中に指定されるそれぞれの照会テーブルの名前です。複合照会テーブルまたは補足照会テーブルの名前は、*prefix.name* という命名規則に従いますが、*prefix* に'IBM' を使用することはできません。

照会テーブル名と接頭部の両方を大文字にする必要があります。照会テーブル名の最大長は 28 文字です。

照会テーブルのフィルター・オプション:

Business Process Choreographer の照会テーブルに対して照会を実行する場合、フィルター・オプションは入力パラメーターとして照会テーブル API のメソッドに渡すことができます。

com.ibm.bpe.api.FilterOptions クラスのインスタンス (Business Flow Manager Enterprise JavaBeans が使用されている場合) または com.ibm.task.api.FilterOptions のインスタンス (Human Task Manager Enterprise JavaBeans が使用されている場合) を、照会テーブル API に渡すことができます。フィルター・オプションでは、以下を使用して照会を構成できます。

- しきい値およびオフセット (skipCount)

- ソート属性 (SQL 照会の ORDER BY 文節に類似)
- ユーザー提供の照会フィルター
- 返される属性のセット (作業項目の情報も含まれる)
- その他

照会テーブルから取得できる結果セットは、照会テーブルの定義で指定します。ただし、照会の実行時に、追加のオプションを指定することもできます。以下の表は、FilterOptions オブジェクトを使用して、フィルター・オプションとして指定できるオプションを示しています。

表 51. 照会テーブル API のパラメーター: フィルター・オプション

オプション	タイプ	説明
選択した属性	java.lang.String	• 結果セットに返される照会テーブルの属性のコンマ区切りリスト。 • インスタンス・ベースの許可を使用する場合は、WORK_ITEM 照会テーブルの属性に接頭部 `WI.` を付けたもの (例えば `WI.REASON`) を指定することによって、作業項目の情報を取得できます。 • ヌルを指定する場合、照会テーブルのすべての属性が返されます (作業項目の情報はなし)。
ソート属性	java.lang.String	照会テーブル属性のコンマ区切りリスト。オプションで、ASC または DESC (それぞれ昇順、降順を表す) が後ろに続きます。 このリストは、SQL ORDER BY 文節に類似しています: *sortAttributes* ::= *attribute* [ASC|DESC] [, *sortAttributes*]。ASC または DESC を指定しない場合は、ASC が想定されます。ソートは、ソート属性の順序で実行されます。例えば、`"STATE DESC, NAME ASC"` という例では、TASK 照会テーブルのタスクが状態別に降順で、また同じ STATE のグループ内で NAME 別に昇順でソートされます。
しきい値	java.lang.Integer	以下の最大値を定義します。 • queryRows を使用する場合に返される行の最大数。 • queryEntities を使用する場合に返されるエンティティーの最大数。それぞれの照会テーブル内の使用可能なエンティティーの実際の数は、エンティティー結果セット内のエンティティー数が照会のエンティティー数のしきい値に達していなくても、そのしきい値を超える場合があります。これは、作業項目の情報を選択する場合の技術上の理由によります。 • queryRowCount または queryEntityCount を使用する場合に返される最大カウント。 デフォルトはヌルです。これは、しきい値が設定されていないことを意味します。
スキップ・カウント	java.lang.Integer	スキップされる行の数 (行ベースの照会) またはエンティティーの数 (エンティティー・ベースの照会) を定義します。しきい値パラメーターの場合と同様に、skipCount は、エンティティー・ベースの照会では正確でない場合があります。 スキップ・カウントは、大容量の結果セットのページ送りを行うために使用されます。デフォルトはヌルです。これは、skipCount が設定されていないことを意味します。

表 51. 照会テーブル API のパラメーター: フィルター・オプション (続き)

オプション	タイプ	説明
時間帯	java.util.TimeZone	タイム・スタンプの変換時に使用される時間帯。例えば、定義済み照会テーブル TASK の CREATED などです。指定しない場合 (ヌル)、サーバーの時間帯が使用されます。
ロケール	java.util.Locale	$LOCALE システム・パラメーターの値を計算するために使用するロケール。例えば、選択基準で $LOCALE を使用するには、'LOCALE=$LOCALE' のようにします。
個別行	java.lang.Boolean	行ベースの照会でのみ使用されます。true に設定すると、行ベースの照会で個別行が返されます。作業項目の情報が重複している可能性があるために固有の行が返されるという意味ではありません。
照会条件	java.lang.String	このオプションは、結果セットに対して追加のフィルタリングを行います。照会テーブルに定義されているすべての属性を参照できます。照会テーブルに許可が必要な場合は、WI 接頭部を使用して、WORK_ITEM 照会テーブルに定義された列を参照することもできます (例えば、WI.REASON=REASON_POTENTIAL_OWNER のように参照できます)。

照会テーブル API の許可オプション:

Business Process Choreographer の照会テーブルに対して照会を実行する場合、許可オプションは入力パラメーターとして照会テーブル API のメソッドに渡すことができます。

com.ibm.bpe.api.AuthorizationOptions クラスまたは com.ibm.bpe.api.AdminAuthorizationOptions のインスタンス (Business Flow Manager EJB が使用されている場合) を使用するか、com.ibm.task.api.AuthorizationOptions クラスまたは com.ibm.task.api.AdminAuthorizationOptions クラスのインスタンス (Human Task Manager EJB が使用されている場合) を使用して、照会実行時に追加の許可オプションを指定します。

インスタンス・ベースの許可を使用する場合、AuthorizationOptions クラスのインスタンスでは、照会で返される適格なインスタンスを識別するための作業項目のタイプを指定できます。

インスタンス・データが含まれている定義済み照会テーブルに対して照会が実行される場合、AuthorizationOptions クラスのインスタンスは、照会テーブル API に渡すことができます。このインスタンスは、インスタンス・データが含まれている 1 次照会テーブルを含む複合照会テーブルに対して照会が実行される場合で、インスタンス・ベースの許可を使用するように構成されている場合にも渡すことができます。テンプレート・データを含む定義済み照会テーブルまたはロール・ベースの許可を構成した複合照会テーブルに対して照会を実行すると、com.ibm.bpe.api.EngineNotAuthorizedException 例外 (Business Flow Manager EJB を使用している場合) または com.ibm.task.api.NotAuthorizedException (Human Task Manager EJB を使用している場合) がスローされます。他のすべての場合、照会テーブル API に渡される許可オプションは無視されます。

複合照会テーブルでは、包含されるオブジェクト (またはエンティティー) を識別するときに、考慮対象となる作業項目のタイプを制限することができます。 例えば、

このことは、照会テーブル API に渡される許可オプションが、全員作業項目を使用するように構成されている場合、全員作業項目が複合照会テーブルの定義で使用するように定義されている場合にのみ考慮されます。簡単なルールとして、照会テーブル定義で考慮対象として指定されていない作業項目タイプは、照会テーブル API によって考慮対象として上書きすることはできません。しかし、照会テーブル定義で考慮対象として指定されている作業項目タイプは、使用しないとして上書きすることができます。また、複合照会テーブルまたは定義済み照会テーブルの許可タイプは、照会テーブル API によって上書きすることはできません。

許可オブジェクトが指定されていない場合、または関連属性 (全員、個人、グループ、または継承) がヌルに設定されている場合 (デフォルト) は、照会する照会テーブルのタイプによって、さまざまな許可オプションのデフォルトが適用されます。

以下の表は、使用される照会テーブル・タイプおよび作業項目タイプのインスタンス・ベースの許可の許可オプションのデフォルトを示しています。

表 52. 照会テーブル API パラメーター: インスタンス・ベース許可の許可オプションのデフォルト

照会テーブルのタイプ	全員作業項目	個人作業項目	グループ作業項目	継承された作業項目
インスタンス・データを含む定義済み照会テーブル	TRUE	TRUE	TRUE	FALSE
テンプレート・データを含む定義済み照会テーブル	N/A	N/A	N/A	N/A
インスタンス・データが含まれている 1 次照会テーブルを含む複合照会テーブル	TRUE	TRUE	TRUE	TRUE
テンプレート・データが含まれている 1 次照会テーブルを含む複合照会テーブル	N/A	N/A	N/A	N/A
補足照会テーブル	N/A	N/A	N/A	N/A

N/A は、インスタンス・ベースの許可は使用されないこと、および作業項目に関連した許可オブジェクトの設定はすべて無視されることを意味します。

TRUE を指定すると、この作業項目のタイプを使用するように照会テーブルが定義されている場合、結果照会では特定の作業項目タイプのみが考慮されます。これは、インスタンス・データを含むすべての定義済み照会テーブルの場合に当てはま

りますが、複合照会テーブルの場合には当てはまりません。グループ作業項目の場合、ヒューマン・タスク・コンテナーでは後者も有効にする必要があります。継承された作業項目を TRUE に設定すると、例えば、プロセス・インスタンスの管理者は、プロセス・インスタンス用に作成される参加ヒューマン・タスク・インスタンスを確認できるようになります。

以下の場合は、AuthorizationOptions クラスのインスタンスではなく、AdminAuthorizationOptions クラスのインスタンスを指定します。

- 照会は、ロール・ベースの許可を持つ照会テーブルに対して実行します。テンプレート・データを含む定義済み照会テーブルはロール・ベースの許可を必要とします。テンプレート・データを含む 1 次照会テーブルを持つ複合照会テーブルは、ロール・ベースの許可を要求するように構成できます。
- インスタンス・データが含まれている照会テーブル、またはインスタンス・データが含まれている 1 次照会テーブルを含む複合照会テーブルに対して照会を実行する場合。この照会では、特定のユーザーの許可による制限に関係なく、照会テーブルの内容が返されます。この動作は、照会 API (照会テーブル API は除外) の queryAll メソッドを使用する場合と同じです。
- 別のユーザーの代わりに照会を実行する必要がある場合。

以下の表は、上記のさまざまな動作が行われる方法を示しています。

表 53. 照会テーブル API パラメーター: AdminAuthorizationOptions

状態	説明
onBehalfUser がヌルに設定されている	• ロール・ベースの許可を持つ照会テーブルに対して照会を実行すると、その照会テーブルのすべての内容が返されます。 • インスタンス・ベースの許可を使用する照会テーブルに対して照会が実行される場合は、照会テーブルに含まれている特定のオブジェクトに特定のユーザーの作業項目が存在するかどうかは検査されません。照会テーブルに含まれているすべてのオブジェクトが返されます。
onBehalfUser が特定のユーザーに設定されている	インスタンス・ベースの許可を使用すると、指定のユーザーの権限で照会が実行され、照会テーブルのオブジェクトはそのユーザーの作業項目に照らして検査されます。

AdminAuthorizationOptions を指定した場合、呼び出し元は、BPESystemAdministrator または BPESystemMonitor Java EE ロール (Business Flow Manager EJB を使用している場合) あるいは TaskSystemAdministrator または TaskSystemMonitor Java EE ロール (Human Task Manager EJB を使用している場合) でなければなりません。

関連概念

494 ページの『照会テーブルの許可』
照会テーブルで照会を実行するときには、インスタンス・ベースの許可、ロール・ベースの許可、または許可なしを使用できます。

パラメーター:

Business Process Choreographer の照会テーブルに対して照会を実行する場合、ユーザー・パラメーターを入力パラメーターとして照会テーブル API のメソッドに渡す

ことができます。照会テーブル定義で、1 次照会テーブル、許可、および照会テーブルのフィルターにパラメーターを指定できます。また、接続される照会テーブルの選択基準にパラメーターを指定することもできます。

システム・パラメーター $USER および $LOCALE は、フィルターおよび選択基準で実行時に置き換えられます。照会テーブル API に渡す必要はありません。$LOCALE システム・パラメーターを計算するための入力値は、フィルター・オプションでロケールを設定することによって指定します。

ユーザー・パラメーターは、照会の実行時に照会テーブル API に渡す必要があります。これは、com.ibm.bpe.api.Parameter クラスのインスタンスのリスト (Business Flow Manager EJB を使用している場合) または com.ibm.task.api.Parameter クラスのインスタンス (Human Task Manager EJB を使用している場合) を渡すことにより実行されます。

パラメーター・オブジェクトで、以下のプロパティーを指定する必要があります。

表 54. 照会テーブル API のユーザー・パラメーター

プロパティー	説明
名前	照会テーブル定義で使用されているパラメーターの名前。この名前では、大/小文字が区別されます。
値	パラメーターの値。パラメーターのタイプは、そのパラメーターを使用するすべてのフィルターおよび選択基準の左オペランドのタイプとの互換性がなければなりません。定義済み照会テーブルの一部の属性で定義されている定数 (STATE_READY など) はストリングとして渡すことができます。

例

```
// execute a query against a composite query
// table CUST.CPM with the primary query table filter
// set to 'STATE=PARAM(theState)'
EntityResultSet ers = null;
List parameterList = new ArrayList();
parameterList.add(new Parameter
("theState", new Integer(2)));

// run the query;
// the business flow manager EJB or the
// human task manager EJB can be used to access query tables
ers = bfm.queryEntities
("CUST.CPM", null, null, parameterList);

// work on the result set
// ...
```

照会テーブルの照会の結果

Business Process Choreographer の照会テーブルに対する照会の実行時に、照会テーブル API メソッドを使用します。 queryEntityCount メソッドまたは queryRowCount メソッドの照会結果は数値です。queryEntities および queryRows メソッドでは、結果セットが返されます。

EntityResultSet

Business Flow Manager Enterprise JavaBeans が使用されている場合は、com.ibm.bpe.api.EntityResultSet クラスのインスタンスが、メソッド queryEntities によって返されます。Human Task Manager Enterprise JavaBeans が使用されている場合は、com.ibm.task.api.EntityResultSet クラスのインスタンスが、メソッド queryEntities によって返されます。エンティティー結果セットには、以下のプロパティーがあります。

表 55. 照会テーブル API エンティティーのエンティティー結果セットのプロパティー

プロパティー	説明
queryTableName	照会が実行された照会テーブルの名前。
entityTypeName	• 複合照会テーブルに対して照会が実行された場合、これは 1 次照会テーブルの名前です。 • 定義済み照会テーブルまたは補足照会テーブルに対して照会が実行された場合、これは照会テーブルの名前です。つまり、*queryTableName* プロパティーと同じ値です。
EntityInfo	このプロパティーには、エンティティー結果セットに含まれるエンティティーのメタ情報が含まれています。com.ibm.bpe.api.AttributeInfo オブジェクトの java.util.List リスト (Business Flow Manager EJB が使用されている場合) または com.ibm.task.api.AttributeInfo のリスト (Human Task Manager EJB が使用されている場合) は、このオブジェクトに対して取得できます。このリストには、この結果セットのエンティティーに含まれる情報の属性名および属性タイプが含まれています。これらのエンティティーのキーを構成する属性のメタ情報も含まれています。
entities	com.ibm.bpe.api.Entity オブジェクトの java.util.List リスト (Business Flow Manager EJB が使用されている場合) または com.ibm.task.api.Entity オブジェクトのリスト (Human Task Manager EJB が使用されている場合)。
locale	$LOCALE システム・パラメーターに対して計算されたロケール。

Entity クラスのインスタンスには、照会テーブルの照会で取得された情報が含まれています。エンティティーは、一意的に識別可能なオブジェクト (タスク、プロセス・インスタンス、アクティビティー、エスカレーションなど) を表します。エンティティーでは、以下のプロパティーを使用できます。

表 56. 照会テーブル API エンティティーのエンティティー・プロパティー

プロパティー	説明
EntityInfo	エンティティー結果セットにも含まれている EntityInfo オブジェクト。com.ibm.bpe.api.AttributeInfo オブジェクトの java.util.List リスト (Business Flow Manager EJB が使用されている場合) または com.ibm.task.api.AttributeInfo のリスト (Human Task Manager EJB が使用されている場合) は、このオブジェクトに対して取得できます。このリストには、この結果セットのエンティティーに含まれる情報の属性名および属性タイプが含まれています。これらのエンティティーのキーを構成する属性のメタ情報も含まれています。
attributeValue (*attributeName*)	このエンティティーで取得された指定された属性の値。タイプは、この属性の関連した AttributeInfo オブジェクトに入れられます。
attributeValuesOfArray (*attributeName*)	値の配列。このプロパティーは、属性情報プロパティー *array* が true に設定されている場合 (この属性が現在作業項目の情報を参照している場合のみ) に使用します。

エンティティー結果セットに含まれるエンティティーの数は、エンティティーのリストに対して size メソッドを使用して取得します。

例: エンティティー・ベースの照会テーブル API:

```
...
// the following example shows a query against
// predefined query table TASK, using the entity-based API

...
// run the query
// service is a (Local)BusinessFlowManager object or a
// (Local)HumanTaskManager object
EntityResultSet rs = service.queryEntities("TASK", null, null, null);

// get the entities meta information
EntityInfo ei = rs.getEntityInfo();
List atts = ei.getAttributeInfo();

// get the entities and iterate over it
Iterator entitiesIter = rs.getEntities().iterator();
while (entitiesIter.hasNext()) {

      // work on a particular entity
      Entity en = (Entity) entitiesIter.next();

      for (int i = 0; i < atts.size(); i++) {
            AttributeInfo ai = (AttributeInfo) atts.get(i);
            Serializable value = en.getAttributeValue(ai.getName()) ;

            // process...
      }
}
...
```

RowResultSet

Business Flow Manager Enterprise JavaBeans が使用されている場合は、com.ibm.bpe.api.RowResultSet クラスのインスタンスが、メソッド queryRows によって返されます。Human Task Manager Enterprise JavaBeans が使用されている場合

は、com.ibm.task.api.RowResultSet クラスのインスタンスが、メソッド queryRows によって返されます。このタイプの結果セットは、JDBC 結果セットに似ています。行結果セットには、以下のプロパティーがあります。

表 57. 照会テーブル API 行の行結果セットのプロパティー

プロパティー	説明
primaryQueryTableName	• 複合照会テーブルに対して照会が実行された場合、これは 1 次照会テーブルの名前です。 • 定義済み照会テーブルまたは補足照会テーブルに対して照会が実行された場合、これは照会テーブルの名前です。つまり、*queryTableName* プロパティーと同じ値です。
attributeInfo	このプロパティーには、com.ibm.bpe.api.AttributeInfo オブジェクトのリスト (Business Flow Manager Enterprise JavaBeans が使用されている場合) または com.ibm.task.api.AttributeInfo オブジェクトのリスト (Human Task Manager Enterprise JavaBeans が使用されている場合) が含まれています。この結果セットのメタ情報が記述されています。AttributeInfo オブジェクトには、情報の属性名および属性タイプが含まれています。行結果セットはキーを持たないため、キーに関するメタデータは含まれません。
attributeValue	この行で取得された指定された属性の値。タイプは、この属性の関連した AttributeInfo オブジェクトに入れられます。
next、first、last、previous	行結果セットは、これらのメソッドを使用してナビゲートします。イテレーター、列挙型、または JDBC 結果セットと使用法を比較してください。

行結果セットに含まれる行の数は、行のリストに対して size メソッドを使用して取得します。

例: 行ベースの照会テーブル API

```
...
// the following example shows a query against
// predefined query table TASK, using the entity-based API
...
// run the query
// service is a (Local)BusinessFlowManager object or a
// (Local)HumanTaskManager object
RowResultSet rs = service.queryRows("TASK", null, null, null);

// get the entities meta information
List atts = rs.getAttributeInfo();

// get the entities and iterate over it
while (rs.next()) {

      // work on a particular row
      for (int i = 0; i < atts.size(); i++) {
            AttributeInfo ai = (AttributeInfo) atts.get(i);
            Serializable value = rs.getAttributeValue(ai.getName()) ;

            // process...
      }
}
...
```

メタデータ取得のための照会テーブルの照会

照会は、照会テーブル API を使用して、Business Process Choreographer 内の照会テーブルに対して実行されます。メソッドを使用して照会テーブルからメタデータを取得できます。

照会テーブル API を使用して Business Process Choreographer 内の照会テーブルに対して照会を実行するときにメタデータを取得するために、以下のメソッドが用意されています。

表 58. 照会テーブルでのメタデータ取得のためのメソッド

目的	メソッド
特定の照会テーブルのメタデータを返す	getQueryTableMetaData
特定のプロパティーを持つ照会テーブル・メタデータのリストを返す	findQueryTableMetaData
(エンティティーに基づいて) 照会テーブルの内容と、選択された属性のメタデータのサブセットを返す	queryEntities
(行に基づいて) 照会テーブルの内容と、選択された属性のメタデータのサブセットを返す	queryRows

照会テーブルのメタデータは、構造に関連するデータおよび国際化対応に関連するデータから構成されます。

照会テーブルの構造に関連したメタデータを以下の表に示します。

表 59. 照会テーブルの構造に関連したメタデータ

メタデータ	説明	getQuery-TableMetaData によって返される	findQuery-TableMetaData によって返される	queryEntities によって返される	queryRows によって返される
照会テーブル名	照会テーブルの名前	はい	はい	はい	はい
1 次照会テーブル名	補足照会テーブルおよび定義済み照会テーブルの場合は照会テーブルの名前。複合照会テーブルの場合は 1 次照会テーブルの名前	はい	はい	はい	はい
種類	照会テーブルのタイプ。predefined (定義済み)、composite (複合)、supplemental (補足) のいずれかです。	はい	はい	いいえ	いいえ
許可	照会テーブルで定義されている以下の許可。 • 作業項目の使用 • インスタンス・ベース、ロール・ベース、または許可なし	はい	はい	いいえ	いいえ

表 59. 照会テーブルの構造に関連したメタデータ (続き)

メタデータ	説明	getQuery-TableMetaData によって返される	findQuery-TableMetaData によって返される	queryEntities によって返される	queryRows によって返される
定義されている属性	照会テーブルで定義されている属性のメタデータ	はい	はい	いいえ。選択された属性のメタデータが返されます。	いいえ。選択された属性のメタデータが返されます。
キー属性	照会テーブルのキー属性	はい	はい	はい	いいえ。行ベースの照会には適用できません。

照会テーブルの国際化対応に関連したメタデータを以下の表に示します。

表 60. 照会テーブルの国際化対応に関連したメタデータ

メタデータ	説明	getQuery-TableMetaData によって返される	findQuery-TableMetaData によって返される	queryEntities によって返される	queryRows によって返される
locales[]	照会テーブルおよび属性の表示名および説明が定義されているロケール。	はい	はい	いいえ	いいえ
ロケール	API に渡されたロケールから得られた `$LOCALE` システム・パラメーターの値。	はい	はい	はい	はい
照会テーブルの表示名および説明	照会テーブルの表示名および説明 (定義されているすべてのロケールに対して提供されているもの)。	はい	はい	いいえ	いいえ
属性の表示名および説明	属性の表示名および説明 (定義されているすべてのロケールに対して提供されているもの)。	はい	はい	いいえ	いいえ

照会テーブル・メタデータを返す EJB 照会テーブル API のメソッドは、いずれもロケール・パラメーターを受け入れます (`FilterOptions.setLocale` や `MetaDataOptions.setLocale` など)。このパラメーターは、クライアントがユーザーに情報を提示するために使用する Java ロケールに設定する必要があります。このロケール・パラメーターは、`$LOCALE` システム・パラメーターの値を計算するために使用されます。このシステム・パラメーターは、フィルターおよび選択基準で使用できます。返されるロケールには、`$LOCALE` に使用される実際の Java ロケールが含まれます。

特定の照会テーブルの表示名および説明を取得する場合には、関連したメソッドに getLocale を渡すと、タスクの説明と同じロケールで表示名および説明が取得されます。例えば、これらの説明は `'LOCALE=$LOCALE'` の選択基準を使用して付加されます。

例

```
// the following example shows how meta data for a particular
// composite query table can be retrieved

...
// run the query
MetaDataOptions mdo = new MetaDataOptions("TASK", null, false, new Locale("en_US"));
List list = bfm.findQueryTableMetaData(mdo);

// to get the meta data of a specific query table
// use bfm.getQueryTableMetaData(...)

// iterate through the list of query tables that have TASK as primary query table
// => at least one query table is returned: the predefined query table TASK

Iterator iter = list.iterator();
while (iter.hasNext()) {
 QueryTableMetaData md = (QueryTableMetaData) iter.next();
 Locale effectiveLocale = md.getLocale();
 String queryTableDisplayName = md.getDisplayName(effectiveLocale);
 System.out.println("found query table: " + queryTableDisplayName);
 List attributesList = md.getAttributeMetaData();
 Iterator attrIter = attributesList.iterator();
 while (attrIter.hasNext()) {
  AttributeMetaData amd = (AttributeMetaData) attrIter.next();
  String attributeDisplayName = amd.getDisplayName(effectiveLocale);
  System.out.println("¥tattribute:" + attributeDisplayName);
 }
}
```

最良一致ロケール

接続される照会テーブルに条件を指定するときに、値 $LOCALE を使用すると、指定されたロケールがメタデータと正確に一致しない場合に、予期しない結果が返される可能性があります。例えば、メタデータで言語が en と指定されている照会テーブルに対して、照会でロケール en_US を渡すと、返される結果は null になります。

このような事態を避けるために、LOCALE=$LOCALE_BEST_MATCH を使用できます。これにより、照会で使用される実際のロケールを計算する最良一致アルゴリズムが適用されます。例えば、言語が en の照会テーブルに対して、ロケール en_US を使用して照会した場合、この照会は en を使用して実行されます。

条件 LOCALE=$LOCALE_BEST_MATCH に、他の論理演算子または比較演算子を指定することはできません。最良一致ロケール条件は、接続される照会テーブルでのみ使用できます。最良一致ロケール条件を他の照会で条件として指定すると、エラーが発生します。

照会テーブル・メタデータの国際化対応

照会テーブル・メタデータは国際化対応がサポートされています。

複合照会テーブルの表示名および説明は複数のロケールで指定できます。例えば、複合照会テーブルでは、en_US ロケール、de ロケール、およびデフォルト・ロケールで照会テーブルの表示名を定義することができます。これは、Query Table Builder を使用して照会テーブルを作成するときに行います。表示名および説明をローカラ

イズした照会テーブルをデプロイするには、照会テーブルを Business Process Choreographer コンテナーにデプロイするときに -deploy jarFile オプションを使用する必要があります。

ロケール処理の点では、照会テーブル API のメソッド queryEntities および queryRows の動作と、照会テーブル API のメタデータ・メソッド getQueryTableMetaData および findQueryTableMetaData は、Java リソース・バンドルによって提供されるものと同様です。

照会テーブル・メタデータの表示名および説明を照会テーブルの内容と整合させるために、$LOCALE システム・パラメーターの値は、照会テーブルで表示名および説明を指定しているロケールによって決定されます。

例

以下のシナリオを考えてみます。このシナリオでは、クライアントがタスク・リストまたはプロセス・リストを表示し、照会テーブルを照会するための要求を作成します。

- クライアントが、ユーザーに情報を提示するために使用するロケールを指定していない。アプリケーションが各種の言語に対応していない可能性があります。
  - 表示名および説明のデフォルト・ロケールが照会テーブルで指定されている。これには、現行バージョンの Query Table Builder で作成されたすべての複合照会テーブルおよび補足照会テーブルが該当します。そのため、$LOCALE の値は default に設定されています。
  - 照会テーブルが、デフォルト・ロケールの照会テーブルの表示名または説明を指定していない。これには、すべての定義済み照会テーブルと、-deploy qtdFile オプションを使用してデプロイされるすべての照会テーブルが該当します。$LOCALE の値は Java リソース・バンドル・メソッドに基づいて決定されます。
- クライアントが、ユーザーに情報を提示するために使用するロケールを指定した。例えば、照会テーブルに対する REST API を使用するときがこれに該当します。
  - 表示名および説明が照会テーブルで指定されている。クライアントによって渡されたロケールに基づき、Java リソース・バンドル・メソッドを使用して $LOCALE の値を計算します。
  - 表示名および説明が照会テーブルで指定されていない。$LOCALE の値が、クライアントによって渡された値に設定されます。

最良一致ロケール

接続される照会テーブルに条件を指定するときに、値 $LOCALE を使用すると、指定されたロケールがメタデータと正確に一致しない場合に、予期しない結果が返される可能性があります。例えば、メタデータで言語が en と指定されている照会テーブルに対して、照会でロケール en_US を渡すと、返される結果は null になります。

このような事態を避けるために、LOCALE=$LOCALE_BEST_MATCH を使用できます。これにより、照会で使用される実際のロケールを計算する最良一致アルゴリズムが適

用されます。例えば、言語が en の照会テーブルに対して、ロケール en_US を使用して照会した場合、この照会は en を使用して実行されます。

条件 LOCALE=$LOCALE_BEST_MATCH に、他の論理演算子または比較演算子を指定することはできません。最良一致ロケール条件は、接続される照会テーブルでのみ使用できます。最良一致ロケール条件を他の照会で条件として指定すると、エラーが発生します。

関連タスク

526 ページの『Business Process Choreographer Explorer の照会テーブルの作成』
EJB 照会 API の代わりに照会テーブルを使用すると、Business Process Choreographer Explorer のパフォーマンスを向上させることができます。照会テーブルを作成するには、Query Table Builder を使用します。

照会テーブルおよび照会パフォーマンス

照会テーブルでは、Business Process Choreographer のヒューマン・タスクおよびビジネス・プロセスのリストを取得するクライアント・アプリケーションを開発するためのクリーンなプログラミング・モデルが導入されています。照会テーブルを使用すると、パフォーマンスが向上します。照会テーブル API パラメーターと、パフォーマンスに影響するその他の要因に関する情報が提供されます。

照会テーブルに対する照会の応答時間は、主に使用した許可オプション、フィルター、および選択基準の影響を受けます。考慮すべき一般的なパフォーマンスに関するヒントは、以下のとおりです。

- 許可オプションはパフォーマンスに大きく影響します。許可は、可能な限り少ないオプション (個人およびグループの作業項目など) を使用して有効化してください。継承される作業項目の使用は避けてください。許可オプションは、照会の実行時にさらに制限できます。また、必要でない場合は、作業項目を使用する許可が不要であると指定してください。
- 作業項目を使用する許可が必要な場合は、許可フィルターを指定してください。例えば、潜在的所有者の作業項目を持つ照会テーブルのオブジェクトのみを許可するには、WI.REASON=REASON_POTENTIAL_OWNER を使用します。
- 1 次照会テーブルでのフィルター処理は効率的です (例えば、TASK が 1 次照会テーブルである照会テーブルで作動可能状態になっているタスクのみを許可するなど)。
- 照会テーブルでのフィルター処理および照会フィルターは、照会の実行時に渡されるフィルターであり、1 次フィルターよりもパフォーマンスが落ちます。
- 可能な場合は、フィルターおよび選択基準のパラメーターを使用するのを避けてください。
- フィルターおよび選択基準で LIKE 演算子を使用するのは避けてください。

複合照会テーブル定義

以下の表は、複合照会テーブルに定義されているオプションの照会パフォーマンスの影響について示しています。複合照会テーブル定義に関連した他のトピックの情報も含まれています。「パフォーマンスへの影響」列に示されている影響は、パフォーマンスへの影響の平均であり、実際に観察される影響は異なる可能性があります。

表 61. 複合照会テーブルのオプションが照会パフォーマンスに与える影響

オブジェクトまたはトピック	パフォーマンスへの影響	説明
照会テーブル・フィルター	マイナス	照会テーブルに対するフィルターは、照会パフォーマンスに対して最もマイナス影響を与えるフィルターです。一般に、これらのフィルターでは、データベースの定義済み索引を使用できません。
1 次照会テーブル・フィルター	プラス	1 次照会テーブルに対するフィルターでは、照会結果セットの計算の初期の段階で、ハイパフォーマンス・フィルター処理が提供されます。1 次照会テーブル・フィルターを使用して、照会テーブルの内容を制限することをお勧めします。
許可フィルター	プラス	許可に対するフィルターを使用すると、1 次照会テーブル・フィルターがパフォーマンスを改善するように、照会のパフォーマンスが改善されます。可能な場合は、許可フィルターを適用してください。例えば、読者作業項目を考慮してはならない場合は、`WI.REASON=REASON_READER` を指定します。
選択基準	なし	1 次照会テーブルと接続される照会テーブルとの一部の関係では、1 対 1 または 1 対 0 の関係を満たすために、選択基準の定義が必要です。一般に、選択基準は、少数の行に対してのみ評価されるため、パフォーマンスに与える影響はわずかです。
パラメーター	なし	現在、照会テーブルでパラメーターを使用しても、パフォーマンスにマイナス影響が及ぶことはありません。それでも、パラメーターは、必要な場合にのみ使用してください。
インスタンス・ベースの許可	マイナス	インスタンス・ベースの許可を使用する場合は、作業項目が存在するかどうかを照会テーブルの各オブジェクトで検査する必要があります。作業項目は、WORK_ITEM 照会テーブルの項目として表現されます。この検査はパフォーマンスに影響を及ぼします。
インスタンス・ベースの許可フラグ: • everybody • individuals • groups • inherited	マイナス	照会テーブルで使用するように指定された各タイプの作業項目はパフォーマンスに影響を与えます。大容量の照会を実行するアプリケーションでは、個人およびグループの作業項目、または、いずれか一方の作業項目のみを使用する必要があります。通常、継承された作業項目は不要です。予定を表すヒューマン・タスクを返すタスク・リストを定義する場合は特にそうです。継承された作業項目は、それが必要であることが明らかな場合にのみ使用します。例えば、ユーザーがエンクロージング・ビジネス・プロセスの許可に基づく読み取りアクセス権限を持っている可能性があるビジネス・プロセスに属するタスクのリストを返すような場合です。
ロール・ベースの許可または許可なし。	なし	ロール・ベースの許可または許可なしを使用する場合は、作業項目に対する検査が実行されません。

表 61. 複合照会テーブルのオプションが照会パフォーマンスに与える影響 (続き)

オブジェクトまたはトピック	パフォーマンスへの影響	説明
定義済み属性の数	現在なし	現在、照会テーブルに含まれている属性の数は、パフォーマンスに影響を与えることはありません。それでも、照会テーブルでは、必要な属性のみを使用してください。

照会テーブル API

以下の表は、照会テーブル API で指定されているオプションの照会パフォーマンスの影響について示しています。「パフォーマンスへの影響」列に示されている影響は、パフォーマンスへの影響の平均であり、実際に観察される影響は異なる可能性があります。

表 62. 照会テーブル API のオプションが照会パフォーマンスに与える影響

オプション	パフォーマンスへの影響	説明
選択した属性	マイナス (少ないほど良い)	照会テーブルに対する照会の実行時に選択されている属性の数は、データベースと Business Process Choreographer 照会テーブル・ランタイムの両方で処理する必要がある数に影響を与えます。また、複合照会テーブルでは、接続される照会テーブルの情報を取得する必要があるのは、その情報が選択された属性で指定されているか、照会テーブル・フィルターまたは照会フィルターによって参照されている場合のみです。
照会フィルター	マイナス	指定する場合、照会フィルターがパフォーマンスに与える影響は、現在、照会テーブル・フィルターと同じです。ただし、フィルターは、照会テーブル API に渡すのではなく、照会テーブルで指定することをお勧めします。
ソート属性	マイナス	照会結果セットのソートはコストがかかる操作であり、ソートを使用する場合、データベース最適化は制限されます。必要がない場合は、ソートは使用しないでください。ただし、ほとんどのアプリケーションでソートが必要です。
しきい値	プラス	しきい値を指定すると、照会パフォーマンスが大幅に改善される可能性があります。しきい値は、常に指定するようにしてください。
スキップ・カウント	マイナス	照会結果セット内の特定の数のオブジェクトのスキップは影響が大きいため、必要な場合 (照会結果に対してページ送りを実行する場合など) にのみ実行してください。
時間帯	なし	時間帯設定はパフォーマンスに影響を与えません。
ロケール	なし	ロケール設定はパフォーマンスに影響を与えません。
個別行	マイナス	照会で重複行の削除を使用すると、パフォーマンスにいくらかの影響が及びますが、重複していない行を取得するために必要です。このオプションは、行ベースの照会にのみ影響を与え、他の場合は無視されます。

表 62. 照会テーブル API のオプションが照会パフォーマンスに与える影響 (続き)

オプション	パフォーマンスへの影響	説明
照会数のカウント	プラス	エンティティーの総数または特定の照会の行数が必要な場合、つまり、照会テーブルのすべての項目の内容が必要であるわけではない場合にのみ、queryEntityCount または queryRowCount メソッドを使用してください。Business Process Choreographer ランタイムは、照会数のカウントでのみ有効な最適化を適用できます。

その他の考慮事項

そのほかに以下の点についてもパフォーマンスに関する考慮が必要です。

表 63. 照会テーブルのパフォーマンス: その他の考慮事項

項目	説明
システム上の照会テーブルの数	照会テーブルの照会のパフォーマンスは、Business Process Choreographer コンテナーにデプロイされている照会テーブルの数によって影響を受けることはありません。同様に、ビジネス・プロセス・インスタンスのナビゲーションや、ヒューマン・タスクの要求または実行操作も、現在時点では影響を受けることはありません。保守容易性上の理由から、照会テーブルは妥当な数にとどめてください。一般に、1 つの照会テーブルは、ユーザー・インターフェースに表示される 1 つのタスク・リストまたはプロセス・リストを表します。
データベースのチューニング	照会テーブルの内容にアクセスするために最適化された SQL を使用しますが、Business Process Choreographer データベースでは、以下のデータベース・チューニングを実装する必要があります。 • データベース・サーバー上で実行されている他のプロセスやハードウェア制約を考慮に入れ、データベース・メモリーを最大に設定する必要があります。 • データベース上の統計は、最新の状態でなければならず、定期的に更新する必要があります。通常、このような手順は、大規模なトポロジーでは既に実装されています。例えば、データベース内のデータの変更を反映するために、最適化プログラムのデータベース統計を週に一度収集します。 • データベース・システムには、データ・コンテナーを再編成 (デフラグ) するためのツールがあります。データベース内のデータの物理的なレイアウトが、照会パフォーマンスおよび照会のアクセス・パスに影響を与える可能性もあります。 • 良好な照会パフォーマンスには、最適な索引が重要です。Business Process Choreographer には、一般的なシナリオのプロセス・ナビゲーションと照会パフォーマンスの両方のために最適化されている定義済みの索引が用意されています。カスタマイズされた環境では、大容量のタスクまたはプロセス・リストの照会をサポートするために、追加の索引が必要となる場合があります。照会テーブルに対して実行される照会をサポートするには、データベースで提供されるツールを使用してください。

Business Space の照会テーブルの作成

Query Table Builder では、定義済みプロパティーを含む複合照会テーブル定義を使用して、Business Space の照会テーブルを作成できます。

始める前に

Query Table Builder は Eclipse プラグインとして使用可能であり、WebSphere Business Process Management SupportPacs のサイトからダウンロードできます。このサイトで、PA71 WebSphere Process Server - Query Table Builder を探します。リンクにアクセスするには、このトピックの関連参照のセクションを参照してください。

手順

1. Query Table Builder で、プロジェクトを右クリックし、「**新規**」 → 「**Business Space の複合照会定義 (Composite Query Definition for Business Space)**」を選択します。ウィザードの指示に従って、照会テーブル定義を作成します。新規照会テーブル定義は、定義済みプロパティーで構成されています。必要な場合は、照会テーブル定義にさたにプロパティーを追加し、その照会テーブル定義ファイルを WebSphere Process Server にデプロイします。

   注: Query Table Builder でプロパティーに付けた名前が、Business Space for Choreographer でタスク・プロパティーの名前として使用されます。
2. 照会テーブル定義ファイルが作成されてデプロイされると、それらを Business Space で構成できます。例えば、「タスク・リスト」ウィジェットの照会テーブル定義ファイルをデプロイした場合は、以下のようにします。
   a. ウィジェット・メニューを開き、「**構成**」を選択し、次に「**コンテンツ**」タブを選択します。
   b. 「**コンテンツ**」タブで「**表示するタスク・リストを選択します**」ドロップダウン・リストを開いて、ウィジェットのユーザーに対して使用可能にすることができるリストを表示します。「**タスク・リストの追加**」を選択します。デプロイした照会テーブル定義が、このリストで選択できるようになっているはずです。

   照会テーブル定義が選択できない場合は、Query Table Builder に戻り、定義ファイルが正しく定義され、デプロイされたかどうかを確認する必要があります。

Business Process Choreographer Explorer の照会テーブルの作成

EJB 照会 API の代わりに照会テーブルを使用すると、Business Process Choreographer Explorer のパフォーマンスを向上させることができます。照会テーブルを作成するには、Query Table Builder を使用します。

始める前に

Query Table Builder は Eclipse プラグインとして使用可能であり、WebSphere Business Process Management SupportPacs のサイトからダウンロードできます。このサイトで、PA71 WebSphere Process Server - Query Table Builder を探します。リ

ンクにアクセスするには、このトピックの関連参照のセクションを参照してください。

手順

1. Query Table Builder で、プロジェクトを右クリックし、「**新規**」 → 「**Business Space の複合照会定義 (Composite Query Definition for Business Space)**」を選択します。 このオプションを使用すると、Business Process Choreographer Explorer に必要なすべての列が確実に事前選択されます。
2. ウィザードの指示に従って、照会テーブル定義を作成します。 必要な場合は、照会テーブル定義にさらにプロパティーを追加します。照会テーブルを定義するときは、以下の側面を考慮してください。

   フィルター基準

   照会テーブルに基づいて Business Process Choreographer Explorer のビューを作成するときは、検索基準に追加のフィルターまたは変数を指定することはできません。照会テーブルの作成時に、これらのフィルター基準および変数のパラメーターを指定する必要があります。

   照会テーブル定義でパラメーターを使用することにより、照会テーブルを Business Process Choreographer Explorer の複数のビューに使用できます。柔軟性を増すために、カスタム・ビュー照会の実行時にパラメーターのデフォルト値を上書きできるかどうかを指定することもできます。

   許可 照会テーブルに基づいて Business Process Choreographer Explorer のビューを作成するときは、ユーザー・ロールに基づいて検索基準をフィルタリングすることはできません。照会テーブルを定義するときに、ユーザー・ロールのフィルター基準を設定する必要があります。テンプレート情報に基づく 1 次照会テーブルの場合は、ロール・ベースの許可ではなくインスタンス・ベースの許可を許可タイプとして使用してください。インスタンス情報に基づく 1 次照会テーブルの場合は、適切なインスタンス・ベースの許可フィルターを指定してください。

   国際化対応

   Query Table Builder でプロパティーを定義するときに、これらのプロパティーの名前と説明を複数の言語で指定することもできます。カスタマイズ・ビューの照会が実行されると、Business Process Choreographer Explorer は、ブラウザーの言語設定に該当する翻訳を使用します。

   照会テーブル定義の表示名および説明

   Query Table Builder で、ビューがサポートするすべての言語での表示名と説明を指定できます。

   列の表示名および説明

   Business Process Choreographer Explorer は、実行時に、結果リストに表示される列について、適切に国際化された列名を取得します。1 次照会テーブルに含まれている列 (PIID など) の場合、Business Process Choreographer Explorer は、サポートされるすべての言語について既に使用可能である翻訳を使用します。

   接続される照会テーブルに含まれている列 (QUERY_PROPERTY など) の場合は、ビジネスでサポートするすべての言語での表示名と説明を Query Table Builder で指定する必要があります。

タスク名および説明

WebSphere Integration Developer でタスク名および説明を国際化した場合、それらは、ブラウザーの言語および国の設定に従って Business Process Choreographer Explorer に表示されます。ブラウザーの設定がプロセス・モデルに定義された設定と一致しない場合は、デフォルト言語の翻訳が使用されます。

ソート基準

照会テーブル定義のソート基準を定義するときに、いくつかのプロパティー (例えば、プロセス状態) は整数値として保管されますが、Business Process Choreographer Explorer では、これらは翻訳されたストリングとして結果リストに表示されます。これにより、一部の言語で予期しないソート結果になる可能性があります。

新規照会テーブル定義は、定義済みプロパティーと、追加で定義されるプロパティーで構成されています。

次のタスク

Query Table Builder の照会定義をアプリケーション・サーバー上にデプロイし、テストします。これが、Business Process Choreographer Explorer が接続しているサーバーである場合は、Business Process Choreographer Explorer を自分用または別のユーザー・グループ用にカスタマイズするときに、この照会テーブルを使用できます。Business Process Choreographer Explorer が別のサーバーに接続している場合は、照会テーブルを使用してカスタマイズ・ビューを作成する前に、その照会テーブルを適切なサーバーにデプロイする必要があります。

関連概念

494 ページの『照会テーブルの許可』
照会テーブルで照会を実行するときには、インスタンス・ベースの許可、ロール・ベースの許可、または許可なしを使用できます。

489 ページの『照会テーブルのフィルターおよび選択基準』
フィルターおよび選択基準は、Query Table Builder を使用した照会テーブルの作成時に定義します (SQL WHERE 文節に似た構文を使用します)。これらの明確に定義されたフィルターおよび選択基準を使用して、照会テーブルの属性に基づく条件を指定します。

520 ページの『照会テーブル・メタデータの国際化対応』
照会テーブル・メタデータは国際化対応がサポートされています。

Business Process Choreographer EJB 照会 API

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。

すべてのユーザーが query メソッドを呼び出すことができます。このメソッドは、作業項目が存在しているオブジェクトのプロパティーを戻します。 queryAll メソッドは、Java EE ロール BPESystemAdministrator、 TaskSystemAdministrator、BPESystemMonitor、 TaskSystemMonitor のいずれかが割り当てられているユーザー

のみが呼び出すことができます。このメソッドは、データベースに格納されているすべてのオブジェクトのプロパティーを戻します。

すべての API 照会は SQL 照会にマップされます。生成される SQL 照会の形式は、次の要因によって異なります。

- 照会が、Java EE ロールの 1 つが割り当てられているユーザーによって呼び出されたかどうか。
- 照会対象オブジェクト。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。
- from 文節、結合条件、およびユーザー固有のアクセス制御条件の挿入。

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。

API query メソッドの構文

Business Process Choreographer API の照会の構文は、SQL 照会の構文に似ています。照会には、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、および時間帯パラメーターを組み込むことができます。

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 64. 各種オブジェクト・タイプの照会構文

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 64. 各種オブジェクト・タイプの照会構文 (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);`

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 文節の構文と似ており、コンマを使用して文節のパーツを区切ります。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 つの列を指定する必要があります。列は、ビュー名と列名を使用して完全に指定される必要があります。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列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 WHERE 文節の構文に似ています。文節から明示的に SQL を追加したり、API 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

  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。

- 設定操作: IN

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。

結果をソートするビューの列のリストを指定できます。これらの列は、ビューと列の名前で完全に修飾されている必要があります。

order-by 文節の構文は、SQL の order-by 文節の構文と似ており、文節の各パーツをコンマで区切ります。列を昇順にソートする場合は `ASC` を指定し、列を降順にソートする場合は `DESC` を指定します。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。

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。複数の列を指定すると、照会結果セットはまず最初の列の値で順序付けされ、次に 2 番目の列の値で順序付けされる、という具合に続きます。SQL 照会のように、order-by 文節の列を、位置によって指定することはできません。

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、無視して照会結果セットで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。この数は、照会結果セットの先頭から数えます。

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実装します。

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。

実動シナリオでの照会結果セットには数千から数百万の項目が含まれる可能性があるので、しきい値パラメーターの値を指定します。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。しきい値パラメーターは、例えば少数の項目のみが一度に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。

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')",
                 (String)null,
                 (Integer)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')",
                 (String)null, (Integer)null, (TimeZone)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。

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。

このタスクについて

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。

手順

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%'",
 (String)null, (Integer)null,
 (Integer)null, (TimeZone)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。

照会結果

照会結果セットには、Business Process Choreographer API 照会の結果が入ります。

結果セットのエレメントは、呼び出し元により指定された where 文節の条件を満たし、かつ呼び出し元に対し表示が許可されているオブジェクトのプロパティーです。エレメントは、API の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 getTimestampAsLong
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",
                                         (String)null, (String)null,
                                         (Integer)null, (TimeZone)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"
```

ユーザー固有のアクセス条件

SQL SELECT ステートメントが API 照会から生成されるときに、ユーザー固有のアクセス条件が追加されます。この条件により、呼び出し元により指定されている条件に一致し、呼び出し元に対し許可されているオブジェクトのみが呼び出し元に戻されます。

追加されるアクセス条件は、ユーザーがシステム管理者であるかどうかによって異なります。

システム管理者以外のユーザーが呼び出した照会

生成される SQL WHERE 文節では、API where 文節と、ユーザー固有のアクセス制御条件が結合されます。この照会は、ユーザーに対しアクセスが許可されているオブジェクト、つまりユーザーが作業項目を所有しているオブジェクトのみを取得します。作業項目とは、ビジネス・オブジェクト (タスクやプロセスなど) の許可のロールへのユーザーまたはユーザー・グループの割り当てを表します。例えば、ユーザー John Smith が特定のタスクの潜在的所有者ロールのメンバーである場合、この関係を表す作業項目オブジェクトがあります。

例えば、グループ作業項目が無効な場合に、システム管理者以外のユーザーがタスクを照会すると、以下のアクセス条件が WHERE 文節に追加されます。

```
FROM TASK TA, WORK_ITEM WI
WHERE WI.OBJECT_ID = TA.TKIID
AND ( WI.OWNER_ID = 'user'
      OR WI.OWNER_ID = null AND WI.EVERYBODY = true )
```

したがって John Smith が、自身が潜在的所有者であるタスクのリストを取得する場合、API where 文節は次のようになります。

```
"WORK_ITEM.REASON == WORK_ITEM.REASON.REASON_POTENTIAL_OWNER"
```

この API where 文節により、SQL ステートメントに次のアクセス条件が追加されます。

```
FROM TASK TA, WORK_ITEM WI
WHERE WI.OBJECT_ID = TA.TKIID
AND ( WI.OWNER_ID = 'JohnSmith'
      OR WI.OWNER_ID = null AND WI.EVERYBODY = true)
AND WI.REASON = 1
```

つまり、John Smith が、自身がプロセス・リーダーまたはプロセス管理者であるアクティビティーやタスクと、作業項目を所有していないアクティビティーやタスクを表示するには、PROCESS_INSTANCE ビューのプロパティー (PROCESS_INSTANCE.PIID など) を照会の select、where、または order-by 文節に追加する必要があります。

グループ作業項目が有効な場合は、ユーザーに対し、グループがアクセスできるオブジェクトへのアクセスを許可するアクセス条件が WHERE 文節に追加されます。

システム管理者が呼び出した照会

システム管理者は、query メソッドを呼び出し、作業項目が関連付けられているオブジェクトを取得できます。この場合、生成される SQL 照会には WORK_ITEM ビューとの結合が追加されますが、WORK_ITEM.OWNER_ID のアクセス制御条件は追加されません。

この場合、タスクの SQL 照会には以下が含まれます。

```
FROM TASK TA, WORK_ITEM WI
WHERE WI.OBJECT_ID = TA.TKIID
```

queryAll 照会

このタイプの照会を呼び出すことができるのは、システム管理者またはシステム・モニターのみです。アクセス制御条件も WORK_ITEM ビューとの結合も追加されません。このタイプの照会では、すべてのオブジェクトの全データが戻されます。

query メソッドと queryAll メソッドの例

以下の例は、標準的な各種 API 照会の構文と、照会の実行時に生成される関連 SQL ステートメントを示します。

例: 作動可能状態のタスクの照会

この例では、query メソッドを使用して、ログオン・ユーザーが作業可能なタスクを取得する方法を示します。

John Smith は、自分に割り当てられているタスクを一覧表示します。ユーザーがタスクの作業を行うには、そのタスクが作動可能状態になっている必要があります。ログオン・ユーザーには、そのタスクの潜在的所有者作業項目も必要です。この照会の query メソッド呼び出しを、次のコード・スニペットに示します。

```
query( "DISTINCT TASK.TKIID",
           "TASK.KIND IN ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING )
             AND " +
           "TASK.STATE = TASK.STATE.STATE_READY AND " +
           "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
            (String)null, (String)null, (Integer)null, (TimeZone)null )
```

SQL SELECT ステートメントの生成時には、次のアクションが実行されます。

- アクセス制御条件が where 文節に追加されます。この例では、グループ作業項目が有効でないことが想定されています。
- 定数 (TASK.STATE.STATE_READY など) が、数値に置き換えられます。
- FROM 文節と結合条件が追加されます。

この API 照会から生成される SQL ステートメントを、次のコード・スニペットに示します。

```
SELECT DISTINCT TASK.TKIID
   FROM   TASK TA, WORK_ITEM WI,
   WHERE  WI.OBJECT_ID = TA.TKIID
   AND    TA.KIND IN ( 101, 105 )
   AND    TA.STATE = 2
   AND    WI.REASON = 1
   AND  ( WI.OWNER_ID = 'JohnSmith' OR WI.OWNER_ID = null AND  WI.EVERYBODY = true )
```

API 照会を特定のプロセスのタスク (sampleProcess など) に限定する場合、この照会は次のようになります。

```
query( "DISTINCT TASK.TKIID",
           "PROCESS_TEMPLATE.NAME = 'sampleProcess' AND "+
           "TASK.KIND IN ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING )
             AND " +
           "TASK.STATE = TASK.STATE.STATE_READY AND " +
           "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
            (String)null, (String)null, (Integer)null, (TimeZone)null )
```

例: 要求済み状態のタスクの照会

この例では、query メソッドを使用して、ログオン・ユーザーが要求したタスクを取得する方法を示します。

ユーザー John Smith は、自身が要求したタスクのうち、まだ要求済み状態であるタスクを検索します。「John Smith により要求された」ことを指定する条件は TASK.OWNER = 'JohnSmith' です。この照会の query メソッド呼び出しを、次のコード・スニペットに示します。

```
query( "DISTINCT TASK.TKIID",
           "TASK.STATE = TASK.STATE.STATE_CLAIMED AND " +
           "TASK.OWNER = 'JohnSmith'",
           (String)null, (String)null, (Integer)null, (TimeZone)null )
```

この API 照会から生成される SQL ステートメントを、次のコード・スニペットに示します。

```
SELECT DISTINCT TASK.TKIID
   FROM   TASK TA, WORK_ITEM WI,
   WHERE  WI.OBJECT_ID = TA.TKIID
```

```
   AND    TA.STATE = 8
   TA.OWNER = 'JohnSmith'
   AND  ( WI.OWNER_ID = 'JohnSmith' OR WI.OWNER_ID = null AND WI.EVERYBODY = true )
```

タスクが要求されると、タスクの所有者に対して作業項目が作成されます。したがって、John Smith が要求したタスクを取得する照会のもう 1 つの作成方法として、TASK.OWNER = 'JohnSmith' を使用する代わりに、次の条件を照会に追加する方法があります。

```
WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER
```

照会は、次のコード・スニペットのようになります。

```
query( "DISTINCT TASK.TKIID",
           "TASK.STATE = TASK.STATE.STATE_CLAIMED AND " +
           "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER",
           (String)null, (String)null, (Integer)null, (TimeZone)null )
```

SQL SELECT ステートメントの生成時には、次のアクションが実行されます。

- アクセス制御条件が where 文節に追加されます。この例では、グループ作業項目が有効でないことが想定されています。
- 定数 (TASK.STATE.STATE_READY など) が、数値に置き換えられます。
- FROM 文節と結合条件が追加されます。

この API 照会から生成される SQL ステートメントを、次のコード・スニペットに示します。

```
SELECT DISTINCT TASK.TKIID
   FROM   TASK TA, WORK_ITEM WI,
   WHERE  WI.OBJECT_ID = TA.TKIID
   AND    TA.STATE = 8
   AND    WI.REASON = 4
   AND  ( WI.OWNER_ID = 'JohnSmith' OR WI.OWNER_ID = null AND WI.EVERYBODY = true )
```

John は休暇に入るため、所属するチームのリーダーである Anne Grant が、John の現在の作業割り当てを確認するとします。Anne にはシステム管理者権限が付与されています。呼び出す照会は、John が呼び出した照会と同じです。ただし Anne は管理者であるため、生成される SQL ステートメントが異なります。生成される SQL ステートメントを次のコード・スニペットに示します。

```
SELECT DISTINCT TASK.TKIID
   FROM   TASK TA, WORK_ITEM WI,
   WHERE  TA.TKIID = WI.OBJECT_ID =
   AND    TA.STATE = 8
   AND   TA.OWNER = 'JohnSmith')
```

Anne は管理者であるため、アクセス制御条件が WHERE 文節に追加されません。

例: エスカレーションの照会

この例では、query メソッドを使用して、ログオン・ユーザーのエスカレーションを取得する方法を示します。

タスクがエスカレートされると、エスカレーション受信者作業項目が作成されます。ユーザー Mary Jones が、Mary 自身にエスカレートされたタスクのリストを表示するとします。この照会の query メソッド呼び出しを、次のコード・スニペットに示します。

```
query( "DISTINCT ESCALATION.ESIID, ESCALATION.TKIID",
         "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_ESCALATION_RECEIVER",
           (String)null, (String)null, (Integer)null, (TimeZone)null )
```

SQL SELECT ステートメントの生成時には、次のアクションが実行されます。

- アクセス制御条件が where 文節に追加されます。この例では、グループ作業項目が有効でないことが想定されています。
- 定数 (TASK.STATE.STATE_READY など) が、数値に置き換えられます。
- FROM 文節と結合条件が追加されます。

この API 照会から生成される SQL ステートメントを、次のコード・スニペットに示します。

```
SELECT DISTINCT ESCALATION.ESIID, ESCALATION.TKIID
  FROM   ESCALATION ESC, WORK_ITEM WI
  WHERE  ESC.ESIID = WI.OBJECT_ID
  AND    WI.REASON  = 10
  AND
 ( WI.OWNER_ID = 'MaryJones' OR WI.OWNER_ID = null AND  WI.EVERYBODY = true )
```

例: queryAll メソッドの使用

この例では、queryAll メソッドを使用して、1 つのプロセス・テンプレートに属するアクティビティーをすべて取得する方法を示します。

queryAll メソッドは、システム管理者権限またはシステム・モニター権限が付与されているユーザーだけが使用できます。プロセス・テンプレート sampleProcess に属するすべてのアクティビティーを取得する照会の queryAll メソッド呼び出しを、次のコード・スニペットに示します。

```
queryAll( "DISTINCT ACTIVITY.AIID",
       "PROCESS_TEMPLATE.NAME = 'sampleProcess'",
       (String)null, (String)null, (Integer)null, (TimeZone)null )
```

この API 照会から生成される SQL 照会を、次のコード・スニペットに示します。

```
SELECT DISTINCT ACTIVITY.AIID
  FROM   ACTIVITY AI, PROCESS_TEMPLATE PT
  WHERE  AI.PTID = PT.PTID
  AND    PT.NAME = 'sampleProcess'
```

この呼び出しは管理者により実行されるため、生成される SQL ステートメントにアクセス制御条件は追加されません。WORK_ITEM ビューとの結合も追加されません。つまり、この照会では、プロセス・テンプレートのすべてのアクティビティー (作業項目のないアクティビティーを含む) が取得されます。

例: 照会への照会プロパティーの組み込み

この例では、query メソッドを使用して、ビジネス・プロセスに属するタスクを取得する方法を示します。このプロセスに対して定義されている照会プロパティーを、検索に組み込むとします。

例えば、1 つのビジネス・プロセスに属し、作動可能状態にあるヒューマン・タスクをすべて検索するとします。プロセスには照会プロパティー **customerID** とその値 CID_12345、および名前空間があります。この照会の query メソッド呼び出しを、次のコード・スニペットに示します。

```
query ( " DISTINCT TASK.TKIID, TASK_TEMPL.NAME, TASK.STATE,
                   PROCESS_INSTANCE.NAME",
          " QUERY_PROPERTY.NAME = 'customerID' AND " +
          " QUERY_PROPERTY.STRING_VALUE = 'CID_12345' AND " +
          " QUERY_PROPERTY.NAMESPACE =
             'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/' AND " +
          " TASK.KIND IN
             ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
          " TASK.STATE = TASK.STATE.STATE_READY ",
          (String)null, (String)null, (Integer)null, (TimeZone)null );
```

2 番目の照会プロパティー (**Priority** など) と特定の名前空間を照会に追加する場合、照会の query メソッド呼び出しは次のようになります。

```
query ( " DISTINCT TASK.TKIID, TASK_TEMPL.NAME, TASK.STATE,
                   PROCESS_INSTANCE.NAME",
          " QUERY_PROPERTY1.NAME = 'customerID' AND " +
          " QUERY_PROPERTY1.STRING_VALUE = 'CID_12345' AND " +
          " QUERY_PROPERTY1.NAMESPACE =
                 'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/' AND " +
          " QUERY_PROPERTY2.NAME = 'Priority' AND " +
          " QUERY_PROPERTY2.NAMESPACE =
                 'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/' AND " +
          " TASK.KIND IN
             ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
          " TASK.STATE = TASK.STATE.STATE_READY ",
          (String)null, (String)null, (Integer)null, (TimeZone)null );
```

複数の照会プロパティーを照会に追加する場合は、コード・スニペットに示されているように、追加する各プロパティーに番号を付ける必要があります。ただし、カスタム・プロパティーの照会を実行するとパフォーマンスに影響します。照会に含まれているカスタム・プロパティーの数に応じてパフォーマンスが低下します。

例: 照会へのカスタム・プロパティーの組み込み

この例では、query メソッドを使用して、カスタム・プロパティーが指定されたタスクを取得する方法を示します。

例えば、カスタム・プロパティー **customerID** とその値 CID_12345 が指定されており、作動可能状態にあるヒューマン・タスクをすべて検索するとします。この照会の query メソッド呼び出しを、次のコード・スニペットに示します。

```
query ( " DISTINCT TASK.TKIID ",
          " TASK_CPROP.NAME = 'customerID' AND " +
          " TASK_CPROP.STRING_VALUE = 'CID_12345' AND " +
          " TASK.KIND IN
             ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
          " TASK.STATE = TASK.STATE.STATE_READY ",
          (String)null, (String)null, (Integer)null, (TimeZone)null );
```

タスクとそのカスタム・プロパティーを取得する場合、照会の query メソッド呼び出しは次のようになります。

```
query ( " DISTINCT TASK.TKIID, TASK_CPROP.NAME, TASK_CPROP.STRING_VALUE",
          " TASK.KIND IN
             ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
          " TASK.STATE = TASK.STATE.STATE_READY ",
          (String)null, (String)null, (Integer)null, (TimeZone)null );
```

この API 照会から生成される SQL ステートメントを、次のコード・スニペットに示します。

```
SELECT DISTINCT TA.TKIID , TACP.NAME , TACP.STRING_VALUE
   FROM  TASK TA LEFT JOIN TASK_CPROP TACP ON (TA.TKIID = TACP.TKIID),
         WORK_ITEM WI
   WHERE WI.OBJECT_ID = TA.TKIID
   AND    TA.KIND IN ( 101, 105 )
   AND    TA.STATE = 2
   AND  (WI.OWNER_ID = 'JohnSmith' OR WI.OWNER_ID IS NULL AND WI.EVERYBODY = 1 )
```

この SQL ステートメントには、TASK ビューと TASK_CPROP ビューの外部結合が含まれています。つまり、WHERE 文節の条件を満たすタスクは、カスタム・プロパティーが含まれていない場合でも取得されます。

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。

このタスクについて

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。

関連概念

『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。

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。

関連タスク

542 ページの『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。

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。

このタスクについて

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。

手順

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",
                (Integer)null, (TimeZone)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), null);
```

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。Human Task Manager API を使用して、保管照会文に関する情報を取得する場合は、StoredQueryData ではなく、戻されるオブジェクトの StoredQuery を使用します。

```
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。

このタスクについて

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。

手順

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",
              (Integer)null, (TimeZone)null,
              (List)null, (String)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",
                          (Integer)null, (Integer)null, (List)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。

```
StoredQueryData 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();
```

Human Task Manager API を使用して、保管照会文に関する情報を取得する場合は、 StoredQueryData ではなく、戻されるオブジェクトの StoredQuery を使用します。

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。

手順

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",
              (Integer)null, (TimeZone)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。

```
StoredQueryData storedQuery = process.getStoredQuery
   ("CustomerOrdersStartingWithA");
String selectClause = storedQuery.getSelectClause();
String whereClause = storedQuery.getWhereClause();
String orderByClause = storedQuery.getOrderByClause();
Integer threshold = storedQuery.getThreshold();
String owner = storedQuery.getOwner();
```

Human Task Manager API を使用して、保管照会文に関する情報を取得する場合は、 StoredQueryData ではなく、戻されるオブジェクトの StoredQuery を使用します。

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");
```

第 11 章 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。

このタスクについて

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。

手順

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実装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許可の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。

関連概念

代替管理許可モード
管理タスクを組み込むようにプロセス・テンプレートをモデル化した場合は、この管理モードによって、それらの管理タスクおよび関連アクション (エスカレーションやモニターなど) が非アクティブ化されます。このモードで実行すると、インスタンス・ベースの管理が使用不可になると共に、プロセス管理およびモニターがシステム管理者またはシステム・モニターであるユーザーに制限されるので、パフォーマンスが向上します。

関連資料

579 ページの『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。

599 ページの『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。

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。

このタスクについて

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。

セッション Bean のリモート・インターフェースにアクセスする

ビジネス・プロセスまたはヒューマン・タスク用の EJB クライアント・アプリケーションでは、Bean のリモート・ホーム・インターフェースを介して、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。

このタスクについて

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。

手順

1. セッション Bean のリモート・インターフェース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 Platform, Enterprise Edition (Java EE) 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. ビジネス・オブジェクトの定義を提供する方法を決定します。

   リモート・クライアント・アプリケーションでビジネス・オブジェクトを操作するには、プロセスまたはタスクとの対話に使用されるビジネス・オブジェクトに対応するスキーマ (XSD または WSDL ファイル) にアクセスできる必要があります。これらのファイルには、以下のいずれかの方法でアクセスできます。
   - クライアント・アプリケーションが Java EE 管理対象環境で稼働しない場合は、ファイルをクライアント・アプリケーションの EAR ファイルにパッケージします。
   - クライアント・アプリケーションが Java EE 管理対象環境の Web アプリケーションまたは EJB クライアントの場合は、ファイルをクライアント・アプリケーションの EAR ファイルにパッケージするか、リモート成果物のロードを利用します。

a. Business Process Choreographer EJB API である createMessage および ClientObjectWrapper.getObject メソッドを使用して、サーバー上の対応するアプリケーションからリモート・ビジネス・オブジェクト定義を透過的にロードします。

b. サービス・データ・オブジェクト・プログラミング API を使用して、ビジネス・オブジェクトを、すでにインスタンス化されたビジネス・オブジェクトの一部として作成するか、または読み取ります。これを行うには、DataObject インターフェースで commonj.sdo.DataObject.createDataObject メソッドまたは getDataObject メソッドを使用します。

c. XML スキーマ any または anyType を使用して型が指定されるビジネス・オブジェクト・プロパティーの値としてビジネス・オブジェクトを作成したい場合は、ビジネス・オブジェクト・サービスを使用してビジネス・オブジェクトを作成するか、または読み取ります。これを行うには、スキーマのロード元となるアプリケーションを指すようにリモート成果物ローダーのコンテキストを設定する必要があります。これにより、適切なビジネス・オブジェクト・サービスを使用できます。

   ビジネス・オブジェクトの作成の例を以下に示します。ここで "ApplicationName" は、ビジネス・オブジェクト定義が含まれているアプリケーションの名前です。

```
BOFactory bofactory = (BOFactory) new
   ServiceManager().locateService("com/ibm/websphere/bo/BOFactory");

com.ibm.wsspi.al.ALContext.setContext
   ("RALTemplateName", "ApplicationName");
try {
     DataObject dataObject = bofactory.create("uriName", "typeName" );
   } finally {
     com.ibm.wsspi.al.ALContext.unset();
   }
```

   XML 入力の読み取りの例を以下に示します。ここで "ApplicationName" は、ビジネス・オブジェクト定義が含まれているアプリケーションの名前です。

```
BOXMLSerializer serializerService =
  (BOXMLSerializer) new ServiceManager().locateService
    ("com/ibm/websphere/bo/BOXMLSerializer");
ByteArrayInputStream input = new ByteArrayInputStream(<?xml?>..);

com.ibm.wsspi.al.ALContext.setContext
          ("RALTemplateName", "ApplicationName");
try {
     BOXMLDocument document = serializerService.readXMLDocument(input);
     DataObject dataObject = document.getDataObject();
   } finally {
     com.ibm.wsspi.al.ALContext.unset();
   }
```

3. Java Naming and Directory Interface (JNDI) からセッション Bean のリモート・ホーム・インターフェースを見つけます。

   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 のリモート・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 Java EE ロールのいずれかがあるかどうかを示します。このコンテキストを使用して、管理セキュリティーが設定されていない場合でも、呼び出しごとに呼び出し元の権限を確認します。管理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("java:comp/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・ロック競合を防ぐには、並列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("java:comp/UserTransaction");
     ```

```
transaction.begin();

//read the activity instance
process.getActivityInstance(aiid);
//claim the activity instance
process.claim(aiid);

transaction.commit();
```

getActivityInstance メソッドおよびその他の読み取り操作は、読み取りロックを設定します。この例では、アクティビティー・インスタンス上の読み取りロックは、アクティビティー・インスタンス上の更新ロックにアップグレードされます。これにより、トランザクションが並列で実行されるときに、データベース・デッドロックが発生することがあります。

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);
```

セッション Bean のローカル・インターフェースにアクセスする

ビジネス・プロセスまたはヒューマン・タスク用の EJB クライアント・アプリケーションでは、Bean のローカル・ホーム・インターフェースを介して、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。

このタスクについて

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、ヒューマン・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。

手順

1. セッション Bean のローカル・インターフェース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 Platform, Enterprise Edition (Java EE) 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) からセッション Bean のローカル・ホーム・インターフェースを見つけます。

   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 BusinessFlowManager bean

    LocalBusinessFlowManagerHome processHome =
         (LocalBusinessFlowManagerHome)initialContext.lookup
         ("java:comp/env/ejb/LocalBusinessFlowManagerHome");
   ```

   セッション Bean のローカル・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. 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 Java EE ロールのいずれかがあるかどうかを示します。このコンテキストを使用して、管理セキュリティーが設定されていない場合でも、呼び出しごとに呼

び出し元の権限を確認します。管理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("java:comp/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("java:comp/UserTransaction");

   transaction.begin();

   //read the activity instance
   process.getActivityInstance(aiid);
   //claim the activity instance
   process.claim(aiid);

   transaction.commit();
   ```

   getActivityInstance メソッドおよびその他の読み取り操作は、読み取りロックを設定します。この例では、アクティビティー・インスタンス上の読み取りロックは、アクティビティー・インスタンス上の更新ロックにアップグレードされます。これにより、トランザクションが並列で実行されるときに、データベース・デッドロックが発生することがあります。

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 HumanTaskManager bean
LocalHumanTaskManagerHome taskHome =
       (LocalHumanTaskManagerHome)initialContext.lookup
       ("java:comp/env/ejb/LocalHumanTaskManagerHome");
```

```
...
//Access the local interface of the 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。

このタスクについて

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。

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

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		
	読者	スターター	管理者
createMessage	x	x	x
createWorkItem			x
delete			x
deleteWorkItem			x
forceTerminate			x
getActiveEventHandlers	x		x

アクション	呼び出し元のプリンシパルのロール		
	読者	スターター	管理者
getActivityInstance	x		x
getAllActivities	x		x
getAllWorkItems	x		x
getClientUISettings	x	x	x
getCustomProperties	x	x	x
getCustomProperty	x	x	x
getCustomPropertyNames	x	x	x
getFaultMessage	x	x	x
getInputClientUISettings	x	x	x
getInputMessage	x	x	x
getOutputClientUISettings	x	x	x
getOutputMessage	x	x	x
getProcessInstance	x	x	x
getVariable	x	x	x
getWaitingActivities	x	x	x
getWorkItems	x		x
restart			x
resume			x
setCustomProperty		x	x
setVariable			x
suspend			x
transferWorkItem			x

注: プロセス管理がシステム管理者に制限される場合、インスタンス・ベースの管理は使用不可になります。つまり、プロセス、スコープ、およびアクティビティーに対する管理アクションは BPESystemAdministrator ロールのユーザーに制限されます。さらに、プロセス・インスタンスまたはその一部の読み取り、表示、およびモニターを実行できるのは、BPESystemAdministrator ロールまたは BPESystemMonitor ロールのユーザーのみになります。この管理モードについて詳しくは、 58 ページの『代替プロセス管理許可モード』を参照してください。

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

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を保証するものではありません。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				
	読者	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
getInputMessage	x	x	x	x	x
getOutputMessage	x	x	x	x	x
getVariable	x	x	x	x	x
getVariableNames	x	x	x	x	x
getInputVariableNames	x	x	x	x	x
getOutputVariableNames	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

注: プロセス管理がシステム管理者に制限される場合、インスタンス・ベースの管理は使用不可になります。つまり、プロセス、スコープ、およびアクティビティーに対する管理アクションは BPESystemAdministrator ロールのユーザーに制限されます。さらに、プロセス・インスタンスまたはその一部の読み取り、表示、およびモニターを実行できるのは、BPESystemAdministrator ロールまたは BPESystemMonitor ロールのユーザーのみになります。この管理モードについて詳しくは、58 ページの『代替プロセス管理許可モード』を参照してください。

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。

このタスクについて

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。

このタスクについて

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。

このタスクについて

microflow によって要求/応答操作が実装されている場合、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。

手順

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),
  (TimeZone)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。

このタスクについて

microflow によって要求/応答操作が実装されている場合、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。

手順

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),
  (TimeZone)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。

手順

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.EXECUTION_MODE_LONG_RUNNING",
      "PROCESS_TEMPLATE.NAME",
       new Integer(50),
       (TimeZone)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、読者、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 human task、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。

手順

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.EXECUTION_MODE_LONG_RUNNING",
      "PROCESS_TEMPLATE.NAME",
       new Integer(50),
       (TimeZone)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、読者、および編集者が決定され、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 human task、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。

始める前に

このタスクについて

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。

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。呼び出し元はプロセス管理者またはシステム管理者でなければなりません。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールの呼び出し元のみがこのアクションを実行できます。

手順

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。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アクティブ化されません。autonomy 属性が `child` に設定されたサブプロセスも、状態が実行中、失敗中、強制終了中、または補正中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・タスクおよびスタンドアロン・タスクは、中断されません。

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

終了、強制終了、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。

このタスクについて

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/応答操作を実装する場合、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。

呼び出し元はプロセス管理者またはシステム管理者でなければなりません。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールの呼び出し元のみがこのアクションを実行できます。

手順

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

場合によっては、リカバリー不能状態にあるトップレベルのプロセス・インスタンスを強制終了しなければならないことがあります。

このタスクについて

このアクションを実行するには、呼び出し元がプロセス管理者またはシステム管理者でなければなりません。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールの呼び出し元のみがこのアクションを実行できます。

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、このアクションは例外的な場合にのみ行ってください。

手順

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);
```

補正を使用してプロセス・インスタンスを終了する場合、プロセスの補正は、障害が最上位スコープで発生したかのように実行されます。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、予定タスク、またはインライン呼び出し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。

このタスクについて

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。

手順

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",
                    (String)null, (Integer)null, (TimeZone)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。

ヒューマン・タスク・アクティビティーの処理

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。

このタスクについて

ヒューマン・タスク・アクティビティーが活動状態にされると、アクティビティー・インスタンスと、関連した予定タスクの両方が作成されます。ヒューマン・タスク・アクティビティーおよび作業項目管理の処理は、Human Task Manager に委任されます。アクティビティー・インスタンスの状態変更はすべてタスク・インスタンスに反映され、その反対にタスク・インスタンスの状態変更はアクティビティー・インスタンスに反映されます。

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。

手順

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",
                    (String)null, (Integer)null, (TimeZone)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 human task 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, myFault,(String)faultNames.get(0) );
```

   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、強制完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。

関連概念

43 ページの『アクティビティーおよびビジネス・プロセスのエラー動作の継続』
ビジネス・プロセスを定義するときには、予期しない障害が発生し、その障害に対して障害ハンドラーが定義されていない場合の動作を指定できます。プロセスを定義するときに「**エラーの継続**」設定を使用すると、障害が発生した場合にプロセスを停止するように指定できます。

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。 initiateAndClaimFirst および 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。この例では、クライアント・サイド・ページ・フローを使用した単独ユーザー・ワークフローの実装を示します。

このタスクについて

単独ユーザー・ワークフローは、ページ・フロー または*画面フロー* とも呼ばれます。以下の 2 種類のページ・フローがあります。

- クライアント・サイド・ページ・フロー。このフローでは、複数ページの Lotus® Forms フォームなどのクライアント・サイド・テクノロジーを使用して、異なるページ間のナビゲーションを実現しています。
- サーバー・サイド・ページ・フロー。このフローは、ビジネス・プロセスと、後続のタスクが同じ担当者に割り当てられるようにモデル化された一連のヒューマン・タスクを使用して実現されます。

サーバー・サイド・ページ・フローはクライアント・サイド・ページ・フローよりも強力ですが、処理のために消費するサーバー・リソースは増えます。したがって、このタイプのワークフローは、主として次の状態の場合に使用することを検討してください。

- ユーザー・インターフェースで実行された手順間でサービスを呼び出す必要がある場合。例えば、データの検索や更新など。
- ユーザー・インターフェースでの対話が完了した後に CEI イベントの書き込みを要求する監査要件が存在する場合。

単独ユーザー・ワークフローの典型的な例は、オンライン・ブックストアでのオーダー・プロセスです。この場合、購入者は書籍を注文するための一連の操作を実行します。この一連の操作は、ヒューマン・タスク・アクティビティー (予定タスク) として実装できます。購入者が複数の書籍を注文することを決定した場合、それがオーダー・プロセスの開始と次のヒューマン・タスク・アクティビティーの要求に相当します。

initiateAndClaimFirst API はページ・フローを開始します。すなわち、指定されたプロセスを開始し、アクティビティーのシーケンス内の最初のヒューマン・タスク・アクティビティーを要求します。そして、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。

completeAndClaimSuccessor API はヒューマン・タスク・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・インスタンス内の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、プロセス・モデル内のすべてのヒューマン・タスク・アクティビティーのトランザクション動作が participates に設定される必要があります。

この例を、Business Flow Manager API と Human Task Manager API の両方を使用する例と比較してください。

手順

1. 書籍注文プロセスを開始し、アクティビティー・シーケンス内の最初のアクティビティーを要求します。 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。

   a. プロセス・テンプレートを取得して、該当するタイプの入力メッセージを作成します。

   ```
   ProcessTemplateData template = process.getProcessTemplate("CustomerOrder");
   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");
   }
   ```

   b. プロセスを開始し、最初のヒューマン・タスク・アクティビティーを要求します。

   ```
   InitiateAndClaimFirstResult result =
     process.initiateAndClaimFirst("CustomerOrder", "MyOrderProcess", input);
   AIID aiid = result.getAIID();
   ClientObjectWrapper input = result.getInputMessage();
   ```

   最初のアクティビティーが要求されると、要求されたアクティビティーの入力メッセージおよび ID が返されます。

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);
```

このアクションは、オーダー番号が含まれる出力メッセージを設定し、シーケンス内の次のアクティビティーを要求します。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。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ヒューマン・タスク・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
  ...
}

aiid = successor.getAIID();
```

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。

関連タスク

605 ページの『ヒューマン・タスクを含む単一の個人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。この例では、サーバー・サイド・ページ・フローを使用した単独ユーザー・ワークフローの実装方法を示します。ワークフローの処理には、Business Flow Manager と Human Task Manager API の両方が使用されます。

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。

このタスクについて

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。

手順

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。

このタスクについて

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アクティブ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。

手順

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。片方向インターフェースを使用する長期実行プロセスの結果は、そのプロセスに出力がないため、getOutputMessage メソッドを使用して取り出すことはできません。ただし、代わりに変数の内容を照会できます。

このタスクについて

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。

手順

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",
                         (String)null, (Integer)null, (TimeZone)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。

このタスクについて

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。

このタスクについて

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。

ヒューマン・タスク・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。

手順

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'",
                    (String)null, (Integer)null, (TimeZone)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、停止したヒューマン・タスク・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。

関連概念

43 ページの『アクティビティーおよびビジネス・プロセスのエラー動作の継続』
ビジネス・プロセスを定義するときには、予期しない障害が発生し、その障害に対して障害ハンドラーが定義されていない場合の動作を指定できます。プロセスを定義するときに「**エラーの継続**」設定を使用すると、障害が発生した場合にプロセスを停止するように指定できます。

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。

このタスクについて

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。

手順

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'",
                     (String)null, (Integer)null, (TimeZone)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、停止したヒューマン・タスク・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、プロセス状態が失敗状態で終了する前にプロセス・レベルの障害ハンドラーが実行されます。

関連概念

43 ページの『アクティビティーおよびビジネス・プロセスのエラー動作の継続』
ビジネス・プロセスを定義するときには、予期しない障害が発生し、その障害に対して障害ハンドラーが定義されていない場合の動作を指定できます。プロセスを定義するときに「**エラーの継続**」設定を使用すると、障害が発生した場合にプロセスを停止するように指定できます。

結合、ループ、またはカウンター評価の失敗により停止したアクティビティーの修復

結合またはループ条件、あるいは forEach カウンター値の評価時に例外が発生したためにアクティビティーが停止することがあります。管理者は、評価がまた失敗する可能性があるなどの理由から、アクティビティーの実行を再試行しないことに決めます。このような場合、Business Process Choreographer EJB API を使用して式に適切な値を指定することにより、プロセスのナビゲーションを続行できます。

このタスクについて

任意のタイプのアクティビティーに対する結合条件の値や、while または repeat-until アクティビティーのループ条件の値を設定できます。開始カウンターと最終カウンターの値、および forEach アクティビティーの完了分岐の最大数を設定することもできます。完了分岐数に設定する値は、プロセス・モデルの forEach アクティビティーの定義によって異なります。モデルに早期終了条件が指定されている場合は、完了分岐の最大数の値を設定します。モデルに早期終了条件が指定されていない場合は、完了分岐の最大数の値を `null` に設定します。

次のサンプルは、ループ条件の値を設定する方法を示します。

手順

1. ループ条件の評価が失敗したために停止した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
   ACTIVITY.STOP_REASON = ACTIVITY.STOP_REASON.STOP_REASON_IMPLEMENTATION_FAILED AND
   (ACTIVITY.KIND = ACTIVITY.KIND.KIND_WHILE OR
      ACTIVITY.KIND = ACTIVITY.KIND.KIND_REPEAT_UNTIL) AND
   PROCESS_INSTANCE.NAME='CustomerOrder'",
   (String)null, (Integer)null, (TimeZone)null );
```

   同様に、結合条件または forEach カウンターの評価が失敗したために停止したアクティビティーもリストできます。

   - 結合条件の失敗については、以下の式を使用します。

     ```
     ACTIVITY.STOP_REASON.STOP_REASON_ACTIVATION_FAILED
     ```

   - forEach カウンターの失敗については、以下の式を使用します。

```
ACTIVITY.STOP_REASON.STOP_REASON_IMPLEMENTATION_FAILED AND
(ACTIVITY.KIND = ACTIVITY.KIND.KIND_FOR_EACH_SERIAL OR
   ACTIVITY.KIND = ACTIVITY.KIND.KIND_FOR_EACH_PARALLEL)
```

このアクションにより、ループ条件の評価が失敗したために停止した CustomerOrder プロセス・インスタンスのアクティビティーが返されます。

2. ループ条件の値 (例: true) を指定します。

```
if (result.size() > 0)
{
  result.first();
  AIID aiid = (AIID) result.getOID(1);

  process.forceLoopCondition(aiid, true);
}
```

このアクションにより、アクティビティーのループ条件の値が true に設定され、プロセス・インスタンスのナビゲーションが続行します。

同様に、結合条件の値 (`process.forceJoinCondition(aiid, true);`) または forEach アクティビティー・カウンターの値 (`process.forceForEachCounterValues(aiid, 1, 5, new Integer(2));`) を設定できます。

停止したアクティビティーに関連付けられた相関セットの更新

相関セットは、Web サービス間のステートフル・コラボレーションをサポートするために使用します。このような場合、Business Process Choreographer EJB API を使用して式に適切な値を指定することにより、プロセスのナビゲーションを続行できます。

このタスクについて

アクティビティーが停止状態の場合は、次のいずれかの理由により、関連する相関セットの更新が必要になる可能性があります。

- 相関セットの評価時に例外が発生しました。相関セットは初期化することになっていますが、相関セットが既に初期化されています。
- 相関セットの評価時に例外が発生しました。相関セットは初期化しないことになっていますが、相関セットの値が設定されていません。この状態は、例えば、初期化アクティビティーがスキップされたために発生することがあります。
- アクティビティーを再試行する必要があります。相関セットがアクティビティーによって初期化された場合は、forceRetry メソッドを呼び出す前に相関セットを未初期化または変更できます。
- アクティビティーを完了する必要があります。相関セットがアクティビティーによって初期化された場合は、forceComplete メソッドを呼び出す前に相関セットを未初期化または変更できます。

プロセス・インスタンスまたはアクティビティー・インスタンスの相関セット・インスタンスを取得できます。以下の例では、相関セット・インスタンスを初期化または未初期化する方法を示します。

手順

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'",
                    (String)null, (Integer)null, (TimeZone)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. アクティビティーに定義されている相関セット・インスタンスを取得します。

```
AIID aiid = null;

List correlationSet = null;

if (result.size() > 0)
{
   result.first();
   AIID aiid = (AIID) result.getOID(1);

   ActivityInstanceData activity = process.getActivityInstance(aiid);

   correlationSet = process.getCorrelationSetInstances
                    (aiid, activity.getInputMessageTypeName());
}
```

3. 相関セット (例: MyCorrelationSet) を未初期化します。

```
for ( int i=0; i<correlationSet.size(); i++ )
{
  CorrelationSetInstanceData correlationSetInstance =
            (CorrelationSetInstanceData)correlationSet.get(i);

  if ( correlationSetInstance.isInitialized() &&
       correlationSetInstance.getCorrelationSetName().equals("MyCorrelationSet") )
  {
    process.uninitializeCorrelationSet
       ( activity.getProcessInstanceID(), correlSetInstance.getCorrelationSetName() );
  }
}
```

このアクションにより、相関セット MyCorrelationSet が未初期化されます。

4. 相関セット (例: MyCorrelationSet) を初期化します。 この例では、相関セットのストリング値プロパティーが設定されます。

```
for ( int i=0; i<correlationSet.size(); i++ )
{
  CorrelationSetInstanceData correlationSetInstance =
            (CorrelationSetInstanceData)correlationSet.get(i);

  if ( correlationSetInstance.getCorrelationSetName().equals("MyCorrelationSet") )
  {
    List correlationSetProperties =
                    correlationSetInstance.getCorrelationSetProperties();
    for ( int j=0; j<correlationSetProperties.size(); j++ )
    {
      CorrelationPropertyInstanceData property =
                (CorrelationPropertyInstanceData)correlationSetProperties.get(j);

      if ( property.getPropertyName().equals("MyProperty") )
      {
        property.setValue("NewValue");

        process.initializeCorrelationSet
           ( activity.getProcessInstanceID(), correlationSetInstance );
      }
    }
  }
}
```

このアクションにより、相関セット MyCorrelationSet のストリング値プロパティー MyProperty が初期化されます。

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 65. 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。
queryProcessTemplates	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・インスタンスの開始に関連しています。

表 66. プロセス・インスタンスの開始に関連する 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 66. プロセス・インスタンスの開始に関連する API メソッド (続き)

メソッド	説明
initiateAndSuspend	プロセス・インスタンスを作成しますが、そのプロセス・インスタンスのそれ以上の処理を直ちに中断します。
initiateAndClaimFirst	プロセス・インスタンスを作成し、最初のインライン・ヒューマン・タスクを要求します。
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 67. 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	終了、失敗、または強制終了状態にある、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。
queryEntities	照会テーブルを使用して、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。

表 67. 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 (続き)

メソッド	説明
getWaitingActivities	メッセージを待機しているアクティビティーに関する情報を返し、それらのアクティビティーの処理を続行できるようにします。
migrate	プロセス・インスタンスを、指定された新しいバージョンのプロセス・モデルにマイグレーションします。

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 68. 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	アクティビティー・インスタンスを完了し、ログオン担当者の同じプロセス・インスタンス内の次のアクティビティーを要求します。
forceComplete	以下のものを強制的に完了させます。 • 状態が「実行中」または「停止」のアクティビティー・インスタンス。 • 状態が「作動可能」または「要求済み」のヒューマン・タスク・アクティビティー。 • 状態が「待機中」の wait アクティビティー。
forceRetry	以下のものを強制的に繰り返します。 • 状態が「実行中」または「停止」のアクティビティー・インスタンス。 • 状態が「作動可能」または「要求済み」のヒューマン・タスク・アクティビティー。

表 68. 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 (続き)

メソッド	説明
forceNavigate、 forceForEach、 forceLoop、 forceJoin	これらのメソッドは、停止したアクティビティーのナビゲーションを強制します。
skip	アクティビティーの処理をスキップします。
jump	あるアクティビティーから別のアクティビティーにジャンプします。
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。
queryEntities	照会テーブルを使用して、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 69. 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。

このタスクについて

Human Task Manager 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。

同期インターフェースを起動する呼び出しタスクの開始

呼び出しタスクは、Service Component Architecture (SCA) コンポーネントに関連付けられます。タスクは、開始されると SCA コンポーネントを起動します。呼び出しタスクを同期的に開始するのは、関連した SCA コンポーネントを同期的に呼び出せる場合に限ってください。

このタスクについて

このような SCA コンポーネントは、Microflow や単純な Java クラスなどとして実装できます。

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。

手順

1. オプション: タスク・テンプレートをリストして、実行する呼び出し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),
     (TimeZone)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呼び出しタスクの開始

呼び出しタスクは、Service Component Architecture (SCA) コンポーネントに関連付けられます。タスクは、開始されると SCA コンポーネントを起動します。呼び出しタスクを非同期的に開始するのは、関連した SCA コンポーネントを非同期的に呼び出せる場合に限ってください。

このタスクについて

このような SCA コンポーネントは、長時間実行プロセスや片方向操作などとして実装できます。

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。

手順

1. オプション: タスク・テンプレートをリストして、実行する呼び出し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),
       (TimeZone)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,
                            (ReplyHandlerWrapper)null);
   ```

タスク・インスタンスの作成と開始

このシナリオでは、コラボレーション・タスク (API ではヒューマン・タスクとも呼ばれる) 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。

手順

1. オプション: タスク・テンプレートをリストして、実行するコラボレーション・タスクのタスク・テンプレート ID (TKTID) を探します。

   タスク・テンプレート ID 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。

   ```
   TaskTemplate[] taskTemplates = task.queryTaskTemplates
   ("TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_HUMAN",
    "TASK_TEMPL.NAME",
     new Integer(50),
     (TimeZone)null);
   ```

   結果は名前でソートされます。ソート済みのタスク・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. コラボレーション・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,
                                          (ReplyHandlerWrapper)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。

予定タスクまたはコラボレーション・タスクの処理

予定タスク (API では参加タスク とも呼ばれる) またはコラボレーション・タスク (API ではヒューマン・タスクとも呼ばれる) 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。プロセスがヒューマン・タスク・アクティビティーにナビゲートしたときなどに、予定タスクとそれに関連した作業項目が作成されます。

このタスクについて

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。

手順

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",
                 (String)null, (Integer)null, (TimeZone)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。

関連概念

102 ページの『コラボレーション・タスクの状態遷移図』
コラボレーション・タスクは、担当者が他の担当者の作業を実行するときに使用できるタスクです。コラボレーション・タスクのライフ・サイクル内では、特定の対話は特定のタスク状態でのみ可能であり、これらの対話が今度はタスクの状態に影響を与えます。

タスク・インスタンスの中断と再開

コラボレーション・タスク・インスタンス (API ではヒューマン・タスク とも呼ばれる) または予定タスク・インスタンス (API では*参加タスク* とも呼ばれる) を中断できます。

始める前に

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。

このタスクについて

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。

手順

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",
                                   (String)null,
                                   (Integer)null,
                                   (TimeZone)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

予定タスク (API では参加 タスクとも呼ばれる) またはコラボレーション・タスク (API ではヒューマン・タスク とも呼ばれる) は非同期に実行します。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。

このタスクについて

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。

手順

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",
                                    (String)null, (Integer)null, (TimeZone)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。

手順

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。

手順

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",
                    (String)null, (Integer)null, (TimeZone)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。

このタスクについて

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。

手順

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'",
                 (String)null, (Integer)null, (TimeZone)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。

このタスクについて

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ヒューマン・タスク・アクティビティー・インスタンス、プロセス・インスタンス、またはタスク・インスタンスのいずれかです。理由は、ユーザーがオブジェクトに対して持っているロールから派生します。ユーザーは、オブジェクト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 つのオブジェクトが複数の作業項目を持つことが可能です。例えば、予定タスク・インスタンスでは、管理者、読者、編集者、所有者の作業項目を同時に持つことができます。

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。

手順

- 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'",
                                    (String)null, (Integer)null,
                                    (TimeZone)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'",
                                    (String)null, (Integer)null,
                                    (TimeZone)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、読者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'",
                 (String)null, (Integer)null, (TimeZone)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。

このタスクについて

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。

臨時の予定タスクまたはコラボレーション・タスクをモデル化できます。それには、com.ibm.task.api.TaskModel クラスのインスタンスを作成し、それを使用して再使用可能なタスク・テンプレートを作成するか、または 1 回実行のタスク・インスタンスを直接作成します。TaskModel クラスのインスタンスを作成するために、一連のファクトリー・メソッドが com.ibm.task.api.ClientTaskFactory ファクトリー・クラスで使用可能です。実行時のヒューマン・タスクのモデル化は、Eclipse Modeling Framework (EMF) に基づいています。

手順

1. createResourceSet ファクトリー・メソッドを使用して org.eclipse.emf.ecore.resource.ResourceSet を作成します。
2. オプション: 複雑なメッセージ・タイプを使用する予定の場合、org.eclipse.xsd.XSDFactory (ファクトリー・メソッド getXSDFactory() を使用して取得できる) を使ってそのメッセージ・タイプを定義するか、または loadXSDSchema ファクトリー・メソッドを使用して既存の XML スキーマを直接インポートできます。

   複雑なタイプを WebSphere Process Server が使用できるようにするには、そのタイプをエンタープライズ・アプリケーションの一部としてデプロイします。
3. タイプ javax.wsdl.Definition の Web サービス記述言語 (WSDL) 定義を作成またはインポートします。

   createWSDLDefinition メソッドを使用して新規の WSDL 定義を作成できます。次に、それをポート・タイプおよび操作に追加できます。また、loadWSDLDefinition ファクトリー・メソッドを使用して、既存の WSDL 定義を直接インポートできます。
4. createTTask ファクトリー・メソッドを使用してタスク定義を作成します。

   より複雑なタスク・エレメントを追加または操作する場合は、getTaskFactory ファクトリー・メソッドを使って取得できる com.ibm.wbit.tel.TaskFactory クラスを使用できます。
5. createTaskModel ファクトリー・メソッドを使用してタスク・モデルを作成し、それをステップ 1 で作成されたリソース・バンドルに渡します。リソース・バンドルはその間に作成されたその他のすべての成果物を集約します。
6. オプション: TaskModel validate メソッドを使用してモデルを検証します。

タスクの結果

TaskModel パラメーターを持つ Human Task Manager EJB API create メソッドの 1 つを使用して、再使用可能なタスク・テンプレートを作成するか、または 1 回実行のタスク・インスタンスを作成します。

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。

このタスクについて

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。

手順

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" ),
             (Map)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。

     ```
     atask.createTask( taskModel, (String)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, (String)null );
     ```

タスクの結果

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。

このタスクについて

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。

手順

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" ),
             (Map)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" );
     ```

タスクの結果

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。

このタスクについて

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。

手順

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", (String)null, (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。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" );
     ```

タスクの結果

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。

このタスクについて

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。

手順

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", (String)null, (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。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" );
     ```

タスクの結果

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。

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 70. 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。
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。

表 70. タスク・テンプレート用の API メソッド (続き)

メソッド	説明
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。
createAndStartTaskAsSubtask	指定されたタスクのサブタスクとしてタスク・インスタンスを作成および開始します。
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 71. 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	呼び出し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。
startTaskAsSubtask	タスク・インスタンスのサブタスクとしてタスクを開始します。
suspend	コラボレーション・タスクまたは予定タスクを中断します。
resume	コラボレーション・タスクまたは予定タスクを再開します。
restart	タスク・インスタンスを再始動します。
terminate	指定されたタスク・インスタンスを終了します。呼び出し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。
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。
completeWithFollowOnTask	タスク・インスタンスを完了し、後続タスクを開始します。

表 71. タスク・インスタンス用の API メソッド (続き)

メソッド	説明
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 72. 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。
triggerEscalation	エスカレーションを手動でトリガーします。

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 73. 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。

ビジネス・プロセスおよびヒューマン・タスク用アプリケーションの開発

ほとんどのビジネス・プロセス・シナリオには担当者が関係します。例えば、プロセスが開始または管理されるとき、あるいはヒューマン・タスク・アクティビティーが実行されるときには、ビジネス・プロセスに担当者の対話が必要となります。これらのシナリオをサポートするために、Business Flow Manager API と Human Task Manager API の両方を使用する必要があります。

このタスクについて

ビジネス・プロセス・シナリオに担当者を組み込むために、ビジネス・プロセスに以下の種類のタスクを含めることができます。

- インライン呼び出しタスク (API では*親タスク* とも呼ばれる)。

  すべての receive アクティビティー、pick アクティビティーの各 onMessage エレメント、およびイベント・ハンドラーの各 onEvent エレメントに対して呼び出しタスクを提供できます。次いで、このタスクはプロセスの開始、または実行中のプロセス・インスタンスとの通信を許可されているユーザーを制御します。
- 管理タスク。

  プロセスの管理またはプロセスの失敗したアクティビティーに対する管理操作の実行を許可されたユーザーを指定するための管理タスクを提供できます。
- 予定タスク (API では*参加タスク* とも呼ばれる)。

  予定タスクはヒューマン・タスク・アクティビティーを実装します。このタイプのアクティビティーにより、プロセスに担当者を含めることができます。

ビジネス・プロセス内のヒューマン・タスク・アクティビティーは、担当者がビジネス・プロセス・シナリオで実行する予定タスクを表します。 Business Flow Manager API と Human Task Manager API の両方を使用して、以下のシナリオを実現できます。

- ビジネス・プロセスとは、プロセスに属するすべてのアクティビティー (予定タスクによって表されるヒューマン・タスク・アクティビティーを含む) のコンテナーです。プロセス・インスタンスが作成されると、固有オブジェクト ID (PIID) が割り当てられます。
- ヒューマン・タスク・アクティビティーがプロセス・インスタンスの実行中に活動状態にされると、アクティビティー・インスタンスが作成されます。これはその固有オブジェクト ID (AIID) によって識別されます。同時に、インライン予定タスク・インスタンスも作成されます。これはそのオブジェクト ID (TKIID) によって識別されます。ヒューマン・タスク・アクティビティーとタスク・インスタンスの関係は、次のようにオブジェクト ID を使用して実現されます。
  - アクティビティー・インスタンスの予定タスク ID が、関連した予定タスクの TKIID に設定されます。
  - タスク・インスタンスの包含コンテキスト ID が、関連したアクティビティー・インスタンスを含むプロセス・インスタンスの PIID に設定されます。
  - タスク・インスタンスの親コンテキスト ID が、関連したアクティビティー・インスタンスの AIID に設定されます。

- すべてのインライン予定タスク・インスタンスのライフ・サイクルは、プロセス・インスタンスによって管理されます。プロセス・インスタンスが削除されると、タスク・インスタンスも削除されます。例えば、包含コンテキスト ID がプロセス・インスタンスの PIID に設定されているすべてのタスクが自動的に削除されます。

開始できるプロセス・テンプレートまたはアクティビティーの判別

ビジネス・プロセスは、Business Flow Manager API の call、initiate、または sendMessage メソッドの呼び出しにより開始できます。プロセスに 1 つの開始アクティビティーしかない場合は、パラメーターとしてプロセス・テンプレート名を必要とするメソッド・シグニチャーを使用できます。プロセスに複数の開始アクティビティーがある場合は、開始アクティビティーを明示的に識別する必要があります。

このタスクについて

ビジネス・プロセスをモデル化する場合、モデラーは、ユーザーのサブセットだけしかプロセス・テンプレートからプロセス・インスタンスを作成できないように決定することができます。これは、インライン呼び出しタスクをプロセスの開始アクティビティーに関連付け、許可制限をそのタスクに指定することで実行できます。タスクの潜在的スターターまたは管理者だけが、タスクのインスタンスの (およびプロセス・テンプレートのインスタンスの) 作成を許可されます。

インライン呼び出しタスクが開始アクティビティーと関連付けられていない場合、または許可制限がタスクに指定されていない場合、誰でも開始アクティビティーを使用してプロセス・インスタンスを作成できます。

プロセスは、それぞれが異なる潜在的スターターまたは管理者に対する担当者照会を持つ、複数の開始アクティビティーを持つことができます。これはつまり、ユーザーがアクティビティー A を使用したプロセスの開始は許可されるが、アクティビティ B を使用しては許可されないということです。

手順

1. Business Flow Manager API を使用して、開始済み状態のプロセス・テンプレートの現行バージョンのリストを作成します。

   ヒント: queryProcessTemplates メソッドは、開始されていないアプリケーションの一部であるプロセス・テンプレートだけを除外します。そのため、結果をフィルター処理せずにこのメソッドを使用する場合、メソッドはどのような状態にあるかに関係なく、すべてのバージョンのプロセス・テンプレートを戻します。

   ```
   // current timestamp in UTC format, converted to yyyy-mm-ddThh:mm:ss
   String now = (new UTCDate()).toXsdString();
   String whereClause = "PROCESS_TEMPLATE.STATE =
                         PROCESS_TEMPLATE.STATE.STATE_STARTED AND
                         PROCESS_TEMPLATE.VALID_FROM =
                         (SELECT MAX(VALID_FROM) FROM PROCESS_TEMPLATE
                          WHERE NAME=PROCESS_TEMPLATE.NAME AND
                          VALID_FROM <= TS('" + now + "'))";

   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
   ```

```
                                        ( whereClause,
                                          "PROCESS_TEMPLATE.NAME",
                                          (Integer)null, (TimeZone)null);
```

結果はプロセス・テンプレート名でソートされます。

2. プロセス・テンプレートのリストと、ユーザーが許可されている開始アクティビティーのリストを作成します。

   プロセス・テンプレートのリストには、単一の開始アクティビティーがあるプロセス・テンプレートが含まれます。これらのアクティビティーは、保護されていないか、またはログオン・ユーザーによる開始が許可されていないかのいずれかです。あるいは、少なくとも 1 つの開始アクティビティーにより開始できるプロセス・テンプレートを収集することもできます。

   ヒント: プロセス管理者は、プロセス・インスタンスを開始することもできます。ただし、Business Flow Manager が代替プロセス管理許可モードを使用している場合、プロセス管理はシステム管理者に制限されるため、BPESystemAdministrator ロールのユーザーのみがこのアクションを実行できます。したがって、テンプレートの完全なリストを入手するには、ログオン・ユーザーが管理者であるかどうかを確認することも必要です。

   ```
   List authorizedProcessTemplates = new ArrayList();
   List authorizedActivityServiceTemplates = new ArrayList();
   ```

3. プロセス・テンプレートごとに、開始アクティビティーを判別します。

   ```
   for( int i=0; i<processTemplates.length; i++ )
   {
     ProcessTemplateData template = processTemplates[i];
     ActivityServiceTemplateData[] startActivities =
                   process.getStartActivities(template.getID());
   ```

4. 開始アクティビティーごとに、関連したインライン呼び出しタスク・テンプレートの ID を取得します。

   ```
   for( int j=0; j<startActivities.length; j++ )
   {
     ActivityServiceTemplateData activity = startActivities[j];
     TKTID tktid = activity.getTaskTemplateID();
   ```

   a. 呼び出しタスク・テンプレートが存在しない場合、プロセス・テンプレートはこの開始アクティビティーによっては保護されません。

      この場合、この開始アクティビティーを使用して、誰でもプロセス・インスタンスを作成できます。

      ```
      boolean isAuthorized = false;
             if ( tktid == null )
          {
            isAuthorized = true;
            authorizedActivityServiceTemplates.add(activity);
          }
      ```

   b. 呼び出しタスク・テンプレートが存在する場合は、Human Task Manager API を使用して、ログオン・ユーザーの許可を検査します。

      例ではログオン・ユーザーは Smith です。ログオン・ユーザーは、呼び出しタスクの潜在的なスターターまたは管理者である必要があります。

      ```
      if ( tktid != null )
          {
            isAuthorized =
      ```

```
        task.isUserInRole
            (tkid, "Smith", WorkItem.REASON_POTENTIAL_STARTER) ||
        task.isUserInRole(tktid, "Smith", WorkItem.REASON_ADMINISTRATOR);

      if ( isAuthorized )
      {
        authorizedActivityServiceTemplates.add(activity);
      }
    }
```

ユーザーに指定されたロールがある場合、またはそのロールの担当者割り当て基準が指定されていない場合、isUserInRole メソッドは値 true を戻します。

5. プロセス・テンプレート名だけを使用してプロセスが開始できるかを確認します。

```
if ( isAuthorized && startActivities.length == 1 )
    {
      authorizedProcessTemplates.add(template);
    }
```

6. ループを終了します。

```
  } // end of loop for each activity service template
}   // end of loop for each process template
```

ヒューマン・タスクを含む単一の個人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。この例では、サーバー・サイド・ページ・フローを使用した単独ユーザー・ワークフローの実装方法を示します。ワークフローの処理には、Business Flow Manager と Human Task Manager API の両方が使用されます。

このタスクについて

単独ユーザー・ワークフローは、*ページ・フロー* または*画面フロー* とも呼ばれます。以下の 2 種類のページ・フローがあります。

- クライアント・サイド・ページ・フロー。このフローでは、複数ページの Lotus Forms フォームなどのクライアント・サイド・テクノロジーを使用して、異なるページ間のナビゲーションを実現しています。
- サーバー・サイド・ページ・フロー。このフローは、ビジネス・プロセスと、後続のタスクが同じ担当者に割り当てられるようにモデル化された一連のヒューマン・タスクを使用して実現されます。

サーバー・サイド・ページ・フローはクライアント・サイド・ページ・フローよりも強力ですが、処理のために消費するサーバー・リソースは増えます。したがって、このタイプのワークフローは、主として次の状態の場合に使用することを検討してください。

- ユーザー・インターフェースで実行された手順間でサービスを呼び出す必要がある場合。例えば、データの検索や更新など。
- ユーザー・インターフェースでの対話が完了した後に CEI イベントの書き込みを要求する監査要件が存在する場合。

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、ヒューマン・タスク・アクティビティー (予定タスク) として実装できます。購入者が複数の書籍を注文する場合は、これが次のヒューマン・タスク・アクティビティーの要求に相当します。一連のタスクに関する情報は Business Flow Manager によって保守されますが、タスク自体は Human Task Manager によって保守されます。

この例を、Business Flow Manager API のみを使用する例と比較してください。

手順

1. Business Flow Manager API を使用して、作業対象となるプロセス・インスタンスを取得します。

   この例では、CustomerOrder プロセスのインスタンスです。

   ```
   ProcessInstanceData processInstance =
      process.getProcessInstance("CustomerOrder");
   String piid = processInstance.getID().toString();
   ```

2. Human Task Manager API を使用して、指定されたプロセス・インスタンスの一部である、準備のできた予定タスク (種類は参加) を照会します。

   タスクの包含コンテキスト ID を使用して、含まれているプロセス・インスタンスを指定します。単一の個人ワークフローの場合、照会によって、一連のヒューマン・タスク・アクティビティーの最初のヒューマン・タスク・アクティビティーに関連付けられている予定タスクが戻されます。

   ```
   //
    // Query the list of to-do tasks that can be claimed by the logged-on user
    // for the specified process instance
   //
   QueryResultSet result =
        task.query("DISTINCT TASK.TKIID",
                   "TASK.CONTAINMENT_CTX_ID = ID('" + piid + "') AND
                    TASK.STATE = TASK.STATE.STATE_READY AND
                    TASK.KIND = TASK.KIND.KIND_PARTICIPATING AND
                    WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                   (String)null, (Integer)null, (TimeZone)null);
   ```

3. 戻される予定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 activityInput = 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。

4. 予定タスクに関連付けられたヒューマン・タスク・アクティビティーを判別します。

以下のメソッドのいずれかを使用して、アクティビティーをそのタスクに相関させます。

- task.getActivityID メソッド:

```
AIID aiid = task.getActivityID(tkiid);
```

- タスク・オブジェクトの一部である親コンテキスト ID:

```
AIID aiid = null;
Task taskInstance = task.getTask(tkiid);

OID oid = taskInstance.getParentContextID();
if ( oid != null and oid instanceof AIID )
{
  aiid = (AIID)oid;
}
```

5. タスクでの作業が終了したら、Business Flow Manager API を使用してタスクとそれに関連するヒューマン・タスク・アクティビティーを完了し、プロセス・インスタンス内の次のヒューマン・タスク・アクティビティーを要求します。

   ヒューマン・タスク・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 human task activity and its associated to-do task,
// and claim the next human task activity
CompleteAndClaimSuccessorResult successor =
   process.completeAndClaimSuccessor(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、シーケンス内の次のヒューマン・タスク・アクティビティーを要求します。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。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ヒューマン・タスク・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。

6. 次のヒューマン・タスク・アクティビティーを処理します。

```
ClientObjectWrapper nextInput = successor.getInputMessage();
if ( nextInput.getObject()!=
               null && nextInput.getObject() instanceof DataObject )
{
  activityInput = (DataObject)input.getObject();
  // read the values
```

```
  ...
}

aiid = successor.getAIID();
```

7. ステップ 5 を続行して、ヒューマン・タスク・アクティビティーを完了し、次のヒューマン・タスク・アクティビティーを取得します。

関連タスク

568 ページの『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。 initiateAndClaimFirst および 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。この例では、クライアント・サイド・ページ・フローを使用した単独ユーザー・ワークフローの実装を示します。

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。

このタスクについて

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。

関連概念

40 ページの『ビジネス・プロセスでの障害処理』
プロセスで障害が発生すると、ナビゲーションが障害ハンドラーまたは障害リンクで続行されます。

Business Process Choreographer EJB 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。

このタスクについて

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 します。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。

ヒューマン・タスク・アクティビティーに設定された障害の検査

ヒューマン・タスク・アクティビティーが処理されると、そのヒューマン・タスク・アクティビティーは正常に完了できます。この場合は、出力メッセージを渡すことができます。ヒューマン・タスク・アクティビティーが正常に完了しない場合は、障害メッセージを渡すことができます。

このタスクについて

障害メッセージを読んでエラーの原因を調べることができます。

手順

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",
                     (String)null, (Integer)null, (TimeZone)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

適切に設計されたプロセスでは、例外および障害は、通常は障害ハンドラーによって処理されます。invoke アクティビティーで発生した例外または障害に関する情報は、アクティビティー・インスタンスから取得できます。

このタスクについて

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。

手順

1. 停止状態のヒューマン・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_STOPPED AND
                    ACTIVITY.KIND=ACTIVITY.KIND.KIND_INVOKE",
                    (String)null, (Integer)null, (TimeZone)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();
  }
}
```

障害が起こったプロセス・インスタンスについての処理されない例外または障害の確認

適切に設計されたプロセスでは、例外および障害はふつう障害ハンドラーによって処理されます。プロセスが両方向操作を実装する場合、障害または処理済みの例外に関する情報は、プロセス・インスタンス・オブジェクトの障害名プロパティーから取得できます。障害の場合は、getFaultMessage API を使用して、対応する障害メッセージを取得することもできます。

このタスクについて

例外がどの障害ハンドラーにも処理されないことが原因でプロセス・インスタンスが失敗した場合は、未処理の例外に関する情報をプロセス・インスタンス・オブジェクトから取得できます。これとは対照的に、障害が障害ハンドラーによってキャッチされた場合、その障害に関する情報は入手できません。ただし、FaultReplyException 例外を使用すれば、障害の名前およびメッセージを取り出して、呼び出し側に戻ることができます。

手順

1. 失敗状態のプロセス・インスタンスをリストします。

```
QueryResultSet result =
     process.query("PROCESS_INSTANCE.PIID",
                   "PROCESS_INSTANCE.STATE =
                       PROCESS_INSTANCE.STATE.STATE_FAILED",
                    (String)null, (Integer)null, (TimeZone)null);
```

   このアクションは、障害を起こしたプロセス・インスタンスが含まれる照会結果セットを戻します。

2. 処理されない例外の情報を読み取ります。

```
if (result.size() > 0)
{
  result.first();
  PIID piid = (PIID) result.getOID(1);
  ProcessInstanceData pInstance = process.getProcessInstance(piid);

  ProcessException excp = pInstance.getUnhandledException();
  if ( excp instanceof RuntimeFaultException )
  {
   RuntimeFaultException xcp = (RuntimeFaultException)excp;
   Throwable cause = xcp.getRootCause();
  }
  else if ( excp instanceof StandardFaultException )
  {
   StandardFaultException xcp = (StandardFaultException)excp;
   String faultName = xcp.getFaultName();
  }
  else if ( excp instanceof ApplicationFaultException )
  {
   ApplicationFaultException xcp = (ApplicationFaultException)excp;
   String faultName = xcp.getFaultName();
  }
}
```

タスクの結果

この情報を使用して、障害名または問題の根本原因を調べます。

第 12 章 ビジネス・プロセスおよびヒューマン・タスク用 Web サービス API クライアント・アプリケーションの開発

Business Process Choreographer 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・アプリケーションの開発プロセスは、Web サービス・プロキシーの生成や、クライアント・アプリケーションへのセキュリティー・ポリシーおよびトランザクション・ポリシーの追加など、いくつもの必須のステップとオプションのステップで構成されています。

このタスクについて

バージョン 7 からは、バージョン 6 (リリース 6.0.2 で初めて公開) での JAX-RPC ベースの Business Process Choreographer Web サービス API に代わって、JAX-WS ベースの Web サービス API が使用されます。JAX-RPC ベースの Business Process Choreographer Web サービス API は推奨されなくなり、JAX-WS ベースの API を使用して新規 Web サービス・クライアント・アプリケーションを実装する必要があります。

注: Business Process Choreographer の Java Message Service (JMS) API は、まだバージョン 6 の WSDL および XML スキーマ定義を使用しています。

任意の Web サービス・クライアント環境でクライアント・アプリケーションを開発できます。以下のステップは、このようなアプリケーションの開発に必要なアクションの概要を示します。

手順

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。
3. 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 *Web サービス・プロキシー* を生成します。
4. 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。
5. 必要なセキュリティー・ポリシーまたはトランザクション・ポリシーをクライアント・アプリケーションに追加します。

Web サービス・コンポーネントおよび一連の制御

Web サービス・アプリケーションでは、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. Web サービス・プロキシー (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、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. Web サービス・プロキシーはデータを 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。

例

Human Task Manager Web サービス API にアクセスして、予定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予定タスクのリストを要求します。
2. WebSphere Process Server は予定タスクのリストを返します。
3. クライアント・アプリケーションは次に、claim Web サービス呼び出しを発行して、いずれかの予定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 Web サービス API 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。

- 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 Web Services (JAX-WS 2.0) 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、WebSphere Integration Developer 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。
- 操作のパラメーター・エレメントで maxOccurs 属性を使用しないでください。またはこの属性の値がデフォルト値の `maxOccurs="1"` に設定されていることを確認してください。
- Web サービス操作のビジネス・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 Web Services (JAX-WS 2.0) ダウンロード・ページ

使用すべき WSDL のスタイル

JAX-WS ベースの Business Process Choreographer Web サービス API

バージョン 7 からは、バージョン 6 (リリース 6.0.2 で初めて公開) での JAX-RPC ベースの Business Process Choreographer Web サービス API に代わって、JAX-WS ベースの Web サービス API が使用されます。2 つの Business Process Choreographer Web サービス・インターフェース (ビジネス・プロセス用およびヒューマン・タスク用に 1 つずつ) が用意されており、それぞれに独自のファイル成果物および XML 定義の名前空間があります。

次の表に、JAX-WS ベースの Web サービスのファイル成果物および XML 定義の名前空間の概要を示します。

表 74. JAX-WS ベースの Web サービスのファイル成果物および XML 定義の名前空間

Business Process Choreographer Web サービス・インターフェース	JAX-WS Web サービス・ファイル成果物	JAX-WS Web サービス XML 名前空間
Business Flow Manager Web サービス	BFMJAXWSService.wsdl	http://www.ibm.com/xmlns/prod/websphere/business-process/services/7.0/Binding
Business Flow Manager Web サービス・インターフェース	BFMJAXWSInterface.wsdl	http://www.ibm.com/xmlns/prod/websphere/business-process/services/7.0
Business Flow Manager Web サービス・データ型	BFMDataTypes.xsd	http://www.ibm.com/xmlns/prod/websphere/business-process/types/7.0
Business Flow Manager コールバック Web サービス	BFMJAXWSCallbackService.wsdl	http://www.ibm.com/xmlns/prod/websphere/business-process/callback-services/7.0/Binding
Business Flow Manager コールバック Web サービス・インターフェース	BFMJAXWSCallbackInterface.wsdl	http://www.ibm.com/xmlns/prod/websphere/business-process/callback-services/7.0
Human Task Manager Web サービス	HTMJAXWSService.wsdl	http://www.ibm.com/xmlns/prod/websphere/human-task/services/7.0/Binding
Human Task Manager Web サービス・インターフェース	HTMJAXWSInterface.wsdl	http://www.ibm.com/xmlns/prod/websphere/human-task/services/7.0
Human Task Manager Web サービス・データ型	HTMDataTypes.xsd	http://www.ibm.com/xmlns/prod/websphere/human-task/types/7.0

表 74. JAX-WS ベースの Web サービスのファイル成果物および XML 定義の名前空間 (続き)

Business Process Choreographer Web サービス・インターフェース	JAX-WS Web サービス・ファイル成果物	JAX-WS Web サービス XML 名前空間
Human Task Manager コールバック Web サービス	HTMJAXWSCallbackService.wsdl	http://www.ibm.com/xmlns/prod/websphere/human-task/callback-services/7.0/Binding
Human Task Manager コールバック Web サービス・インターフェース	HTMJAXWSCallbackInterface.wsdl	http://www.ibm.com/xmlns/prod/websphere/human-task/callback-services/7.0
共通 Business Process Choreographer データ型	BPCDataTypes.xsd	http://www.ibm.com/xmlns/prod/websphere/bpc-common/types/7.0

Business Process Choreographer Web サービス API: 標準

以下のリンクを使用して、Web アプリケーションに適用される標準についての関連する補足情報を検索します。情報は、IBM 以外のインターネット・サイトにあり、情報の技術的な正確性はサイトの提供者が管理しています。

これらのリンクは便宜上提供しているものです。多くの場合、情報は、IBM WebSphere Process Server に固有のものではありませんが、一般的な Web サービスを理解するのに役立ちます。

- Java API for XML-based Web Services (JAX-WS 2.0) (JSR-224; Java Community Process)
- Java Architecture for XML Binding (JAXB) 2.0 (JSR-222; Java Community Process)
- Web Services Description Language (WSDL) 1.1 (W3C)
- XML Schema Part 0: Primer Second Edition (W3C)
- XML Schema Part 1: Structures Second Edition (W3C)
- XML Schema Part 2: Datatypes Second Edition (W3C)
- Simple Object Access Protocol (SOAP) 1.1 (W3C)
- Web Services Policy Framework (WS-Policy) 1.5 (W3C)
- WS-Security 1.1 (OASIS)
- WS-Security UserName Token Profile 1.1 (OASIS)
- WS-AtomicTransaction 1.2 (OASIS)
- WS-Interoperability Basic Profile 1.1 (WS-Interoperability Organization)

Web サービス・クライアント・アプリケーションの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 Business Process Choreographer 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。

このタスクについて

エクスポート対象の成果物は、以下のとおりです。

- Business Process Choreographer Web サービス API (常に Web サービス・プロキシー生成のために必要) を構成する Web サービス・エンドポイント、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。

- Business Process Choreographer WSDL ファイル (常に Web サービス・プロキシー生成のために必要) のサービス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)
- WebSphere サーバーで実行されているビジネス・プロセスまたはヒューマン・タスクのインターフェースおよびデータ型を記述する独自の WSDL ファイルおよび XSD ファイル。これらの追加ファイルは、クライアント・アプリケーションが Web サービス API を介してビジネス・プロセスまたはヒューマン・タスクと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、プロセス・インスタンスまたはタスク・インスタンスと直接対話することなく Business Process Choreographer により実行できる操作 (照会の発行など) を呼び出すだけの場合は、これらは不要です。
- Web サービス API のサービス品質属性を記述した Web サービス・ポリシー (WS-Policy) ファイル。これらは、クライアント・サイドの Web サービス・ポリシーを作成するための基礎とするためにエクスポートされる場合があります。

  WS-Security
  要求メッセージには、UserName トークンまたは LPTA トークンが含まれていなければなりません。

  WS-Transaction
  要求メッセージには、WS-AtomicTransaction コンテキストを含めることができます。このコンテキストが存在する場合、要求は呼び出し元のトランザクション・スコープで処理されます。

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、Web サービス・プロキシーおよびヘルパー・クラスを生成するために使用されます。

Business Process Choreographer 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。これらのファイルは、アプリケーションの Web サービス・プロキシーを生成するために使用されます。

始める前に

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。

このタスクについて

これらの WSDL ファイルと、その WSDL ファイルで参照される XSD ファイルを公開する必要があります。すると、それらのファイルを WebSphere 環境からご使用の開発環境にコピーして、Web サービス・プロキシーおよびヘルパー・クラスの生成に使用することができます。Business Process Choreographer WSDL ファイルの公開が必要なのは一度だけです。

Web サービス・アプリケーションのビジネス・プロセス WSDL ファイルの公開

管理コンソールを使用して、WSDL ファイルを公開します。

手順

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**」をクリックします。

   注: 「**アプリケーション**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere 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。

タスクの結果

エクスポートされた .zip ファイルの名前は `BPEContainer_nodename_servername_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 ファイルによって参照される XSD ファイルが含まれています。

注: エクスポートされた .zip ファイルには、バージョン 7 で導入された JAX-WS Web サービスとバージョン 6 で使用される JAX-RPC Web サービスの両方の WSDL および XSD 成果物が含まれています。wsimport ツールを使用して Web サービス・プロキシーを生成する場合、JAX-WS Web サービス成果物を選択します。このとき JAX-RPC 成果物は無視されます。

Web サービス・アプリケーションのヒューマン・タスク WSDL ファイルの公開

管理コンソールを使用して、WSDL ファイルを公開します。

手順

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**」をクリックします。

   注: 「**アプリケーション**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere 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。

タスクの結果

エクスポートされた .zip ファイルの名前は `TaskContainer_nodename_servername_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 ファイルによって参照される XSD ファイルが含まれています。

注: エクスポートされた .zip ファイルには、バージョン 7 で導入された JAX-WS Web サービスとバージョン 6 で使用される JAX-RPC Web サービスの両方の WSDL および XSD 成果物が含まれています。wsimport ツールを使用して Web サービス・プロキシーを生成する場合、JAX-WS Web サービス成果物を選択します。このとき JAX-RPC 成果物は無視されます。

ビジネス・プロセスおよびヒューマン・タスク Web サービス・アプリケーションの WSDL および XSD ファイルのエクスポート

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。WSDL インターフェース定義と XML スキーマ・データ型定義を、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。

このタスクについて

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・プロセスまたはヒューマン・タスクごとに、繰り返してください。

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・テンプレート名前空間
- 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ます。

手順

1. WebSphere Integration Developer 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、UserData 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。

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。これらのクラスは、エクスポートされた WSDL ファイルおよび XSD ファイルが含まれている Web サービス・プロキシー生成の一部として生成されます。

Web サービス・インターフェースの汎用操作は、ビジネス・プロセスまたはヒューマン・タスクによって実装される操作との間で文書ラッパー・エレメントを伝搬します。前述の例で示したサンプル操作の場合、Web サービス SOAP メッセージは、以下のようになります。

```
<soapenv:Envelope xmlns:soapenv="..." ...>
  <soapenv:Header>
    ...
  </soapenv:Header>
  <soapenv:Body>
    <bfm:sendMessage
         xmlns:bfm="http://www.ibm.com/xmlns/prod/websphere/business-process/services/7.0">
      <processTemplateName>customerProcessTemplate</processTemplateName>
      <portType xmlns:cns="http://example.com/customerProcess">cns:customerProcessPortType</portType>
      <operation>updateCustomer</operation>
      <input>
        <cns:updateCustomer xmlns:cns="http://example.com/customerProcess">
          <street>1600 Pennsylvania Avenue Northwest</street>
          <city>Washington, DC 20006</city>
        </cns:updateCustomer>
      </input>
    </bfm:sendMessage>
  </soapenv:Body>
</soapenv:Envelope>
```

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任意の Java ベースの開発環境を使用して、Business Process Choreographer 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。

Web サービス・プロキシーの生成 (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、*Web サービス・プロキシー* を使用して Business Process Choreographer Web サービス API と対話します。

このタスクについて

Java Web サービスの Web サービス・プロキシーには、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 JavaBeans クラスが含まれています。 Web サービス・プロキシーは、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 Web サービス・プロキシーは、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: Web サービス・プロキシー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 Web サービス・プロキシーを使用できます。

IBM Web サービス環境では、次のいずれかの方法で Web サービス・プロキシーを生成できます。

- Rational® Application Developer または WebSphere Integration Developer の統合開発環境を使用する。
- wsimport 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、wsimport 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 Web サービス・アプリケーション用 Web サービス・プロキシーの生成

Rational Application Developer の統合開発環境を使用して、Web サービス・クライアント・アプリケーション用の Web サービス・プロキシーを生成できます。以下の一連のステップは、Rational Application Developer バージョン 7.5.3 に適用されます。

始める前に

Web サービス・プロキシー 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 ファイルおよび XSD ファイルを WebSphere 環境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。

手順

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・プロセスの場合:
     a. エクスポート・ファイル `BPEContainer_nodename_servername_WSDLFiles.zip` を一時ディレクトリーに unzip します。このディレクトリーの内容を変更しないでください。また、ビジネス・プロセスと対話するための Web サービス・プロキシー生成に使用されるのは、以下の WSDL ファイルおよび XSD ファイルのみであることに注意してください。
        – BFMJAXWSService.wsdl
        – BFMJAXWSInterface.wsdl
        – BFMJAXWSCallbackService.wsdl
        – BFMJAXWSCallbackInterface.wsdl
        – BFMDataTypes.xsd
        – BPCDataTypes.xsd
        – wsa.xsd
     b. サブディレクトリー `META-INF` を、unzip されたディレクトリー `BPEContainer_`*nodename_servername*`.ear/bfmjaxws.jar` からインポートします。

- ヒューマン・タスクの場合:
  a. エクスポート・ファイル `TaskContainer_nodename_servername_WSDLFiles.zip` を一時ディレクトリーに unzip します。このディレクトリーの内容を変更しないでください。また、ヒューマン・タスクと対話するための Web サービス・プロキシー生成に使用されるのは、以下の WSDL ファイルおよび XSD ファイルのみであることに注意してください。
     – HTMJAXWSService.wsdl
     – HTMJAXWSInterface.wsdl
     – HTMJAXWSCallbackService.wsdl
     – HTMJAXWSCallbackInterface.wsdl
     – HTMDataTypes.xsd
     – BPCDataTypes.xsd
     – wsa.xsd
  b. サブディレクトリー `META-INF` を、unzip されたディレクトリー `TaskContainer_`*nodename_servername*`.ear/htmjaxws.jar` からインポートします。

  新しい `wsdl` ディレクトリーおよびサブディレクトリー構造がプロジェクト内に作成されます。
2. 新たに作成された `wsdl` ディレクトリー内にある `BFMJAXWSService.wsdl` ファイルを選択します。
3. 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。

   残りのステップを続行する前に、サーバーが開始していることを確認します。
4. 「Web サービス」ウィンドウで「**次へ**」をクリックして、デフォルトをすべて受け入れます。
5. 「Web サービスの JAX-WS Web サービス・クライアント構成 (Web Service JAX-WS Web Service Client Configuration)」ウィンドウで、生成される JAX-WS コードのバージョンを 2.0 に変更し、「**終了**」をクリックして他のすべてのデフォルト値を受け入れます。
6. `HTMJAXWSService.wsdl` ファイルに対してこの手順のステップ 2 から 5 を再び実行し、プロンプトが出されたらすべてのファイルを上書きします。

タスクの結果

複数のプロキシー、ロケーター、および JAXB クラスで構成される Web サービス・プロキシーが生成され、プロジェクトに追加されます。

wsimport コマンド行ツールによる Web サービス・アプリケーション用 Web サービス・プロキシーの生成

wsimport コマンド行ツールを使用して、Web サービス・アプリケーションの Web サービス・プロキシーを生成します。

始める前に

Web サービス・プロキシー 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。

手順

1. Business Process Choreographer Web サービス API の Web サービス・プロキシーを生成します。

   注: JAX-WS アプリケーションの wsimport コマンド行ツールの詳細な説明については、WebSphere Application Server の wsimport コマンド行ツールの資料を参照してください。

   ```
   wsimport.bat BFMJAXWSService.wsdl myService1.wsdl myService2.wsdl
    -d proxy-bfm
    -wsdllocation <bfm_location>

   wsimport.bat HTMJAXWSService.wsdl myService1.wsdl myService2.wsdl
    -d proxy-htm
    -wsdllocation <htm_location>
   ```

   この例では、`myService1.wsdl` および `myService2.wsdl` には、カスタム・ビジネス・プロセス、ヒューマン・タスク、またはその両方のインターフェース定義が含まれています。さらに、<*bfm_location*> および <*htm_location*> は、それぞれ `BFMJAXWSService.wsdl` および `HTMJAXWSService.wsdl` 内の WSDL <*port*> エレメントから取得できます。

   両方のプロキシーを、1 つの共通のディレクトリー (例えば proxy-bpc) にマージして、プロンプトが出されたら既存のファイルを上書きします。
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。

関連タスク

『ビジネス・プロセスおよびヒューマン・タスク用クライアント・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、Business Process Choreographer 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。Web サービス・プロキシーを使用して、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。

ビジネス・プロセスおよびヒューマン・タスク用クライアント・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、Business Process Choreographer 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。Web サービス・プロキシーを使用して、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。

始める前に

クライアント・アプリケーションの作成を開始する前に、Web サービス・プロキシーを生成します。

このタスクについて

Web サービス対応の開発ツール (IBM Rational Application Developer など) を使用すると、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。

手順

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. Web サービス・プロキシーを生成します。
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。

例

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。

```
try {
   // create bfm proxy
   BFMJAXWSPortType bfm = new BFMJAXWSService().getBFMJAXWSPort();

   // call getProcessTemplate
   ProcessTemplateType ptt =
     bfm.getProcessTemplate("MY_PROCESS_TEMPLATE_NAME");

   // handle return value
   System.out.println("Process template '" + ptt.getName() +
       "' found, details following:");
   System.out.println("Execution mode: " +
       ptt.getExecutionMode());
   System.out.println("Schema version: " +
       ptt.getSchemaVersion());
 } catch (Exception e) {
   if ( e instanceof ProcessFaultMsg )
   {
     ProcessFaultMsg pfm = (ProcessFaultMsg) e;
     List<FaultStackType> list =
         ( pfm.getFaultInfo() ).getFaultStack();
     FaultStackType fault = list.get( 0 );
     System.out.println( "ProcessFaultMessage: " +
         fault.getMessage() );
   }
   else
   {
     e.printStackTrace( System.out );
   }
 }
```

関連タスク

623 ページの『wsimport コマンド行ツールによる Web サービス・アプリケーション用 Web サービス・プロキシーの生成』
wsimport コマンド行ツールを使用して、Web サービス・アプリケーションの Web サービス・プロキシーを生成します。

621 ページの『Web サービス・プロキシーの生成 (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、*Web サービス・プロキシー* を使用して Business Process Choreographer Web サービス API と対話します。

セキュリティーの追加

Business Process Choreographer Web サービスでは、クライアント・アプリケーションを認証メカニズムに対応するように構成する必要があります。

このタスクについて

デフォルトでは、Business Process Choreographer は以下の認証メカニズムをサポートします。

ユーザー名トークン
Web サービス・コンシューマーは、「ユーザー名」によって要求側を識別 (オプションでパスワードを使用してその ID を認証) する手段として、ユーザー名トークンを Web サービス・プロバイダーに提供します。

バイナリー・セキュリティー・トークン - Lightweight Third-Party Authentication (LTPA) トークン
Web サービス・コンシューマーは、要求元を認証する手段として、LTPA トークンを Web サービス・プロバイダーに提供します。

Business Process Choreographer Web サービス・セキュリティー・ポリシーを、代替の認証メカニズムで置き換えることができます。ただし、Business Process Choreographer Web サービス操作を非認証ユーザーとして呼び出すことはできないため、1 つの認証メカニズムが常に必要となります。

トランザクション・サポートの追加

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。

このタスクについて

Business Process Choreographer は、それぞれの Web サービス操作要求を別個のグローバル・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。

- クライアントのトランザクション・コンテキストを伝搬させる。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るため、クライアントのトランザクションと一緒にコミット (またはバックアウト) されます。逆に、Web サービス操作の実行中にサーバーで問題が発生して、サーバーがロールバックを要求すると、クライアント・アプリケーションのトランザクション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。Business Process Choreographer は、新規グローバル・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。

Business Process Choreographer Web サービスに付加された Web サービス・ポリシーでは、前述のように各要求メッセージに WS-AT トランザクションを含めることを許可します。クライアント・トランザクション・コンテキストを渡さずに Web サービス操作を呼び出すことを選択する場合は、プロバイダー側のトランザクション・ポリシーを無視して、トランザクション・ポリシーなしで Web サービス・クライアントを構成するのが安全です。

第 13 章 Business Process Choreographer JMS API を使用したクライアント・アプリケーションの開発

Java Messaging Service (JMS) API を介してビジネス・プロセス・アプリケーションに非同期で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。

このタスクについて

JMS クライアント・アプリケーションは要求および応答メッセージを JMS API と交換します。要求メッセージを作成するために、クライアント・アプリケーションは JMS TextMessage メッセージ本文に、対応する操作の文書/リテラル・ラッパーを表す XML エレメントを入力します。

ビジネス・プロセスの要件

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・プロセスが、JMS 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、WebSphere Integration Developer 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 サービス操作のビジネス・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

JMS レンダリングの許可

JMS インターフェースの使用を許可するには、WebSphere Application Server でセキュリティー設定が使用可能になっている必要があります。

ビジネス・プロセス・コンテナーがインストールされている場合、ロール **JMSAPIUser** をユーザー ID にマップする必要があります。このユーザー ID を使用して、すべての JMS API 要求を発行します。例えば、**JMSAPIUser** が「User A」にマップされる場合、すべての JMS API 要求はプロセス・エンジンにとって「User A」から発生しているように見えます。

JMSAPIUser ロールには以下の権限を割り当てる必要があります。

要求	必要な許可
forceTerminate	プロセス管理者
sendEvent	可能なアクティビティー所有者またはプロセス管理者

注: その他のすべての要求の場合、特殊権限は必要ありません。

特殊権限は、ビジネス・プロセス管理者のロールを持つ人物に付与されます。ビジネス・プロセス管理者は特殊なロールです。これは、プロセス・インスタンスのプロセス管理者とは異なります。ビジネス・プロセス管理者はすべての特権を持っています。

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できません。このユーザー ID を削除する場合、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>
```

JMS インターフェースへのアクセス

JMS インターフェースを介してメッセージを送受信するには、まずアプリケーションで BPC.cellname.Bus への接続を作成し、セッションを作成し、メッセージ・プロデューサーおよびコンシューマーを生成する必要があります。

このタスクについて

プロセス・サーバーは、point-to-point パラダイムに従う Java Message Service (JMS) メッセージを受け入れます。 JMS メッセージを送受信するアプリケーションは、以下のアクションに従う必要があります。

以下の例では、JMS クライアントは管理対象環境 (EJB、アプリケーション・クライアント、または Web クライアント・コンテナー) で実行していると想定しています。

手順

1. BPC.*cellname*.Bus への接続を作成します。 クライアント・アプリケーションの要求用の事前構成された接続ファクトリーは存在しません。つまり、クライアント・アプリケーションは、JMS API の ReplyConnectionFactory を使用するか、または固有の接続ファクトリーを作成するかのいずれかが可能です。作成する場合は接続ファクトリーを検索するために、Java Naming and Directory Interface (JNDI) 参照を使用できます。 JNDI 参照名は、Business Process Choreographer の外部要求キューの構成時に指定したものと同じ名前である必要があります。以下の例では、クライアント・アプリケーションが「jms/clientCF」という固有の接続ファクトリーを作成すると想定しています。

   ```
   //Obtain the default initial JNDI context.
   Context initialContext = new InitialContext();

   // Look up the connection factory.
   ```

```
// Create a connection factory that connects to the BPC bus.
// Call it, for example, "jms/clientCF".
// Also configure an appropriate authentication alias.
ConnectionFactory connectionFactory =
            (ConnectionFactory)initialcontext.lookup("jms/clientCF");

// Create the connection.
Connection connection = connectionFactory.createConnection();
```

2. セッションを作成し、メッセージ・プロデューサーおよびコンシューマーが作成されるようにします。

```
// Create a transaction session using auto-acknowledgment.
Session session = connection.createSession(true, Session.AUTO_ACKNOWLEDGE);
```

3. メッセージを送信するメッセージ・プロデューサーを作成します。 JNDI 参照名は、Business Process Choreographer の外部要求キューの構成時に指定したものと同じ名前である必要があります。

```
// Look up the destination of the Business Process Choreographer input queue to
// send messages to.
Queue sendQueue = (Queue) initialcontext.lookup("jms/BFMJMSAPIQueue");

// Create a message producer.
MessageProducer producer = session.createProducer(sendQueue);
```

4. 応答を受信するメッセージ・コンシューマーを作成します。 応答宛先の JNDI 参照名は、ユーザー定義の宛先を指定できますが、デフォルトの (Business Process Choreographer 定義の) 応答宛先 jms/BFMJMSReplyQueue も指定できます。どちらの場合も、応答宛先は BPC.<cellname>.Bus にしておく必要があります。

```
// Look up the destination of the reply queue.
Queue  replyQueue = (Queue) initialcontext.lookup("jms/BFMJMSReplyQueue");

// Create a message consumer.
MessageConsumer consumer = session.createConsumer(replyQueue);
```

5. メッセージを送信します。

```
// Start the connection.
connection.start();

// Create a message - see the task descriptions for examples - and send it.
// This method is defined elsewhere ...
String payload = createXMLDocumentForRequest();
TextMessage requestMessage = session.createTextMessage(payload);

// Set mandatory JMS header.
// targetFunctionName is the operation name of JMS API
// (for example, getProcessTemplate, sendMessage)
requestMessage.setStringProperty("TargetFunctionName", targetFunctionName);

// Set the reply queue; this is mandatory if the replyQueue
// is not the default queue (as it is in this example).
requestMessage.setJMSReplyTo(replyQueue);

// Send the message.
producer.send(requestMessage);

// Get the message ID.
String jmsMessageID = requestMessage.getJMSMessageID();

session.commit();
```

6. 返信を受信します。

```
// Receive the reply message and analyse the reply.
TextMessage  replyMessage = (TextMessage) consumer.receive();

// Get the payload.
String payload = replyMessage.getText();

session.commit();
```

7. 接続を閉じ、リソースを解放します。

```
 // Final housekeeping; free the resources.
session.close();
connection.close();
```

注: 各トランザクションの後に接続を閉じることは不要です。接続が開始したら、接続が閉じるまでに、要求および応答メッセージはいくつでも交換できます。例では、単一のビジネス・メソッド内に単一の呼び出しがある単純なケースを示しています。

Business Process Choreographer JMS メッセージの構造

各 JMS メッセージのヘッダーおよび本文には事前定義構造がある必要があります。

Java Message Service (JMS) メッセージは以下のもので構成されます。

- メッセージ識別およびルーティング情報用のメッセージ・ヘッダー。
- 目次を保持するメッセージの本文 (ペイロード)。

Business Process Choreographer はテキスト・メッセージ形式のみサポートします。

メッセージ・ヘッダー

JMS は、クライアントが多くのメッセージ・ヘッダー・フィールドにアクセスできるようにします。

Business Process Choreographer JMS クライアントは以下のヘッダー・フィールドを設定できます。

JMSReplyTo
要求に対する応答を送信する宛先。このフィールドが要求メッセージで指定されていない場合、応答は Export インターフェースのデフォルトの応答宛先に送信されます (Export は、ビジネス・プロセス・コンポーネントのクライアント・インターフェース・レンダリングです)。この宛先は、`initialContext.lookup("jms/BFMJMSReplyQueue");` を使用して取得できます。

TargetFunctionName
WSDL 操作の名前 (例えば、"queryProcessTemplates")。このフィールドは常に設定する必要があります。 TargetFunctionName は、ここで説明されている汎用の JMS メッセージ・インターフェースの操作を指定することに注意してください。これを、例えば call または sendMessage 操作を使用して、間接的に呼び出すことができる具体的なプロセスまたはタスクによって提供される操作と混同しないでください。

また、Business Process Choreographer クライアントは以下のヘッダー・フィールドにアクセスできます。

JMSMessageID

メッセージを一意的に識別します。メッセージが送信されると、JMS プロバイダーによって設定されます。メッセージの送信前にクライアントが JMSMessageID を設定する場合、JMS プロバイダーによって上書きされます。メッセージの ID が認証目的で必要とされる場合、クライアントはメッセージの送信後に JMSMessageID を取得できます。

JMSCorrelationID

メッセージをリンクします。このフィールドは設定しないでください。Business Process Choreographer 応答メッセージには、要求メッセージの JMSMessageID が含まれています。

各応答メッセージには以下の JMS ヘッダー・フィールドが含まれています。

- **IsBusinessException**

  WSDL 出力メッセージの場合は "false" で、WSDL 障害メッセージの場合は "true" です。

ServiceRuntimeExceptions は非同期クライアント・アプリケーションに戻されません。 JMS 要求メッセージの処理中に重大な例外が発生すると、それはランタイム障害となり、この要求メッセージを処理しているトランザクションはロールバックします。その後、JMS 要求メッセージが再度送達されます。障害が早期に (メッセージの非直列化中など、メッセージを SCA エクスポートの一部として処理している間に) 発生した場合は、SCA エクスポートの受信宛先によって指定された送達の失敗最大数まで再試行されます。送達の失敗最大数に達した後、要求メッセージは Business Process Choreographer バスのシステム例外宛先に追加されます。しかし、Business Flow Manager の SCA コンポーネントによる要求を実際に処理している間に障害が発生する場合、失敗した要求メッセージは WebSphere Process Server の失敗イベント管理インフラストラクチャーによって処理されます。つまり、再試行しても例外状態が解決しない場合は、最終的に、失敗イベント管理データベースに入れられることがあります。

メッセージ本文

ビジネス・プロセスまたはヒューマン・タスクによって公開される操作は、文書/リテラル・ラッパー・スタイルに準拠していなければなりません。JMS メッセージ本文は、操作の文書/リテラル・ラッパー・エレメントを表す XML 文書が含まれているストリングです。JMS メッセージ・インターフェースの汎用操作は、ビジネス・プロセスまたはヒューマン・タスクによって実装される操作との間で文書ラッパー・エレメントを伝搬します。

次の例は、単純で有効な要求メッセージ本体を示します。

```
<bfm:queryProcessTemplates
     xmlns:bfm="http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0">
  <whereClause>PROCESS_TEMPLATE.STATE IN (1)</whereClause>
</bfm:queryProcessTemplates>
```

次の例は、より複雑で有効な要求メッセージ本体を示します。クライアント・アプリケーションには、メッセージを特定のプロセスにサブミットするための sendMessage API 操作があります。プロセス入力メッセージは API パラメーターの 1 つであり、このメッセージは、カスタマー・プロセスによって公開されるビジネ

ス操作の入力メッセージです。プロセスには、メッセージをコンシュームする receive アクティビティーが含まれています。

bfm:sendMessage エレメントは、JMS API 操作の文書ラッパー・エレメントです。この中に組み込まれている cns:updateCustomer エレメントは、プロセスによって実装される操作の文書ラッパー・エレメントです。このプロセスには、例えば、cns:customerProcessPortType WSDL ポート・タイプと updateCustomer WSDL 操作を参照する bpel:receive アクティビティーが含まれています。

```
<bfm:sendMessage
     xmlns:bfm="http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0">
  <processTemplateName>customerProcessTemplate</processTemplateName>
  <portType xmlns:cns="http://example.com/customerProcess">cns:customerProcessPortType</portType>
  <operation>updateCustomer</operation>
  <cns:updateCustomer xmlns:cns="http://example.com/customerProcess">
    <street>1600 Pennsylvania Avenue Northwest</street>
    <city>Washington, DC 20006</city>
  </cns:updateCustomer>
</bfm:sendMessage>
```

関連タスク

635 ページの『応答メッセージでのビジネス例外の検査』
JMS クライアント・アプリケーションは、すべての応答メッセージのメッセージ・ヘッダーを検査して、ビジネス例外を調べる必要があります。

JMS クライアント・アプリケーションの成果物のコピー

JMS クライアント・アプリケーションを容易に作成するために、 WebSphere Process Server 環境から多数の成果物をコピーすることができます。

このタスクについて

これらの成果物は、JMS メッセージ本文の作成に BOXMLSerializer を使用する場合にのみ必須です。 JMS API の場合、成果物は以下のとおりです。

```
BFMIF.wsdl
BFMIF.xsd
BPCGen.xsd
wsa.xsd
```

これらのファイルを WebSphere Process Server 環境から開発環境に公開およびエクスポートする必要があります。

JMS アプリケーションのビジネス・プロセス WSDL ファイルの公開

管理コンソールを使用して、WSDL ファイルを公開します。

手順

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**」をクリックします。

   注: 「**アプリケーション**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere 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。

タスクの結果

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、WSDL ファイルと、その WSDL ファイルによって参照される XSD ファイルが含まれています。

応答メッセージでのビジネス例外の検査

JMS クライアント・アプリケーションは、すべての応答メッセージのメッセージ・ヘッダーを検査して、ビジネス例外を調べる必要があります。

このタスクについて

JMS クライアント・アプリケーションはまず、応答メッセージのヘッダーの **IsBusinessException** プロパティーを検査する必要があります。

以下に例を示します。

例

```
// receive response message
Message receivedMessage = ((JmsProxy) getToBeInvokedUponObject().receiveMessage();
String strResponse = ((TextMessage) receivedMessage).getText();

if (receivedMessage.getStringProperty("IsBusinessException") {
  // strResponse is a bussiness fault
  // any api can end w/a processFaultMsg
  // the call api also w/a businessFaultMsg
 }
else {
  // strResponse is the output message
}
```

関連概念

632 ページの『Business Process Choreographer JMS メッセージの構造』
各 JMS メッセージのヘッダーおよび本文には事前定義構造がある必要があります。

例: Business Process Choreographer JMS API を使用して長期実行プロセスを実行する

この例では、長期実行プロセスで処理するために JMS API を使用する汎用のクライアント・アプリケーションの作成方法を示します。

手順

1. 630 ページの『JMS インターフェースへのアクセス』の説明に従って、JMS 環境をセットアップします。
2. 以下のように、インストールされたプロセス定義のリストを取得します。
   - queryProcessTemplates を送信します。
   - これにより、ProcessTemplate オブジェクトのリストが戻されます。
3. 以下のように、開始アクティビティー (createInstance="yes" を指定した receive または pick) のリストを取得します。
   - getStartActivities を送信します。
   - これにより、InboundOperationTemplate オブジェクトのリストが戻されます。
4. 入力メッセージを作成します。 これは環境に固有で、事前に配置された、プロセス固有の成果物を使用しなければならないことがあります。
5. プロセス・インスタンスを作成します。
   - sendMessage を発行します。

   JMS API と共に call 操作を使用して、ビジネス・プロセスによって提供される長期実行の要求/応答操作と対話することもできます。この操作では、長期間たった後でも、操作の結果または障害を指定された応答宛先に戻します。そのため、call 操作を使用する場合、query および getOutputMessage 操作を使用してプロセスの出力または障害メッセージを取得する必要はありません。
6. オプション: 以下のステップを繰り返して、プロセス・インスタンスから出力メッセージを取得します。
   a. query を発行して、終了状態のプロセス・インスタンスを取得します。
   b. getOutputMessage を発行します。
7. オプション: 以下のように、プロセスによって公開された追加の操作を実行します。
   a. getWaitingActivities または getActiveEventHandlers を発行して、InboundOperationTemplate オブジェクトのリストを取得します。
   b. 入力メッセージを作成します。
   c. sendMessage を発行してメッセージを送信します。
8. オプション: プロセスまたはプロセスに含まれるアクティビティーに対して定義されたカスタム・プロパティーを、getCustomProperties および setCustomProperties で取得および設定します。
9. 以下のように、プロセス・インスタンスでの作業を終了します。
   a. delete および terminate を送信して、長期実行プロセスでの作業を終了します。

第 14 章 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 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。

このタスクについて

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。ヒューマン・タスクを含むアプリケーションでは、JSF カスタム・クライアントを生成できます。JSF クライアントの生成について詳しくは、WebSphere Integration Developer のインフォメーション・センターを参照してください。

Business Process Choreographer で提供される JSF コンポーネントを使用して Web クライアントを開発することもできます。

手順

1. 動的プロジェクトを作成し、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・ライブラリー参照を 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。的確なエラー処理が行われるようにするには、JSF 構成ファイルで、エラー Bean とナビゲーション・ターゲットの両方をエラー・ページ用に定義する必要もあります。エラー Bean の名前には `BPCError`、エラー・ページのナビゲーション・ターゲットの名前には `error` が使用されていることを確認します。

```
<faces-config>
...
<managed-bean>
  <managed-bean-name>BPCError</managed-bean-name>
  <managed-bean-class>com.ibm.bpc.clientcore.util.ErrorBeanImpl
  </managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>

...
<navigation-rule>
...
```

```
<navigation-case>
<description>
The general error page.
</description>
<from-outcome>error</from-outcome>
<to-view-id>/Error.jsp</to-view-id>
</navigation-case>
...
</navigation-rule>
</faces-config>
```

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。

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・コードを実装します。

5. アプリケーションをデプロイします。

アプリケーションを Network Deployment 環境でデプロイしている場合、ターゲット・リソースの Java Naming and Directory Interface (JNDI) 名を、Business Flow Manager および Human Task Manager API をセル内で検出するための値に変更してください。

- ビジネス・プロセス・コンテナーが同じ管理対象セル内の別のサーバー上で構成されている場合、名前には以下の構造があります。

  cell/*nodes*/*nodename*/*servers*/*servername*/com/ibm/bpe/api/BusinessManagerHome
  cell/*nodes*/*nodename*/*servers*/*servername*/com/ibm/task/api/HumanTaskManagerHome

- ビジネス・プロセス・コンテナーが同じセル内のクラスターで構成されている場合、名前には以下の構造があります。

  cell/*clusters*/*clustername*/com/ibm/bpe/api/BusinessFlowManagerHome
  cell/*clusters*/*clustername*/com/ibm/task/api/HumanTaskManagerHome

EJB 参照を 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 75. 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
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

タスクの結果

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。

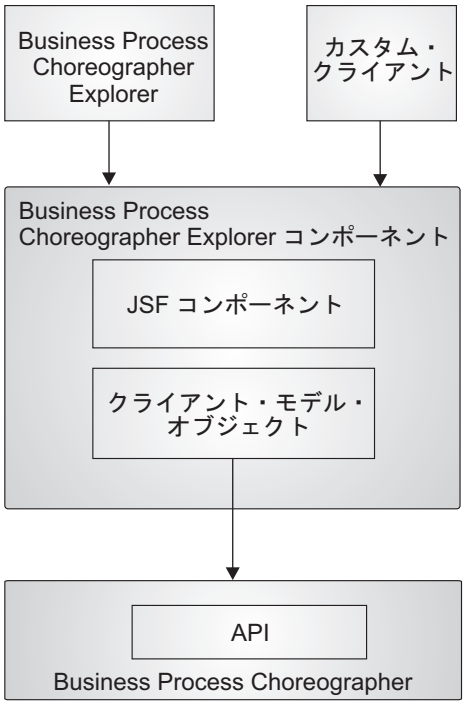
次のタスク

プロセスおよびタスク・メッセージにカスタム JSP を使用している場合、JSP をデプロイするために使用される Web モジュールを、カスタム JSF クライアントがマップされるのと同じサーバーにマップする必要があります。

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実装し、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。

JSF コンポーネントでのエラー処理

JavaServer Faces (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。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。

この Bean は、com.ibm.bpc.clientcore.util.ErrorBean インターフェースを実装します。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実装もしない場合
- 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実装を使用できます。

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();
}
```

関連概念

647 ページの『List コンポーネントでのエラー処理』
List コンポーネントを使用して、JSF アプリケーション内のリストを表示する場合、com.ibm.bpe.jsf.handler.BPCListHandler クラスが提供するエラー処理機能を利用できます。

クライアント・モデル・オブジェクトのデフォルトのコンバーターおよびラベル

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実装します。

List コンポーネントと Details コンポーネントはあらゆる Bean で機能します。Bean のプロパティーをすべて表示できます。ただし、Bean のプロパティーに使用されるコンバーターとラベルを設定する場合は、List コンポーネントの `column` タグまたは Details コンポーネントの `property` タグを使用する必要があります。コンバーターとラベルを設定する代わりに、プロパティーのデフォルトのコンバーターとラベルを定義できます。これらを定義するには、次の静的メソッドを定義します。定義できる静的メソッドを次に示します。

```
static public String getLabel(String property,Locale locale);
static public com.ibm.bpc.clientcore.converter.SimpleConverter
       getConverter(String property);
```

次の表に、対応する Business Flow Manager API クラスと Human Task Manager API クラスを実装し、そのプロパティーにデフォルトのラベルとコンバーターを提供するクライアント・モデル・オブジェクトを示します。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 76. 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

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。

手順

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>TaskPoolQuery</managed-bean-name>
<managed-bean-class>sample.TaskPoolQuery</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>

<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。

照会の結果を強い型の Bean のリストとして表すことができる限り、任意のタイプの照会ロジックを JSF アプリケーションに追加できます。例えば、TaskPoolQuery は com.ibm.task.clientmodel.bean.TaskInstanceBean オブジェクトのリストを使用して実装されます。

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 TaskPoolQuery 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.
    //
    // Add the database columns for all of the properties you want to show
    // in your list to the select statement
    //
    QueryResultSet queryResult = taskService.query(
     "DISTINCT TASK.TKIID, TASK.NAME, TASK.KIND, TASK.STATE, TASK.TYPE,"
      + "TASK.STARTER, TASK.OWNER, 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)",
      (String)null,
      (Integer)null,
      (TimeZone)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実装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。

タスクの結果

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。

リストの処理方法

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、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る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実装されます。このインターフェースの実装は、JavaServer Faces (JSF) アプリケーションの構成ファイルに登録できます。

リスト内のリンクがクリックされると、通知がトリガーされます。 **action** 属性が設定されているすべての列についてリンクがレンダリングされます。 **action** 属性の値は JSF ナビゲーション・ターゲットか、JSF ナビゲーション・ターゲットを戻す JSF アクション・メソッドのいずれかです。

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実装することができます。

照会の実装

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実装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、照会実装の 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()` でメソッドを実装する必要があります。

デフォルトのコンバーターおよびラベル

照会によって戻される項目は Bean である必要があり、そのクラスは BPCListHandler クラスまたは com.ibm.bpc.clientcore.Query インターフェースの定義

でタイプとして指定されたクラスと一致している必要があります。さらに、List コンポーネントは、項目クラスまたはスーパークラスが以下のメソッドを実装しているかどうかを検査します。

```
static public String getLabel(String property,Locale locale);
static public com.ibm.bpc.clientcore.converter.SimpleConverter
       getConverter(String property);
```

これらのメソッドが Bean に対して定義されている場合、List コンポーネントはリストのデフォルト・ラベルとして label を使用し、プロパティーのデフォルト・コンバーターとして SimpleConverter を使用します。これらの設定は、bpe:list タグの **label** および **converterID** 属性で上書きできます。詳しくは、SimpleConverter インターフェースおよび ColumnTag クラスの Javadoc を参照してください。

ユーザー固有の時間帯情報

JavaServer Faces (JSF) コンポーネントは、List コンポーネントのユーザー指定の時間帯情報を処理するためのユーティリティーを提供します。

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実装が **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();
}
```

List コンポーネントでのエラー処理

List コンポーネントを使用して、JSF アプリケーション内のリストを表示する場合、com.ibm.bpe.jsf.handler.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実装に渡します。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実装します。 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。

関連概念

641 ページの『JSF コンポーネントでのエラー処理』
JavaServer Faces (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。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。

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で処理を繰り返し、オブジェクトごとにすべての列をレンダリングします。

表 77. bpe:list 属性

属性	必須	説明
buttonStyleClass	いいえ	フッター領域内のボタン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・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。
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。値が true に設定されている場合は、チェック・ボックス列がレンダリングされます。
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。
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。
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。
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。
selectAll	いいえ	この属性が true に設定されている場合は、リスト内のすべての項目がデフォルトで選択されます。

表 77. bpe:list 属性 (続き)

属性	必須	説明
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の CSS スタイル・クラス。

表 78. bpe:column 属性

属性	必須	説明
action	いいえ	この属性が指定されている場合は、列でリンクがレンダリングされます。リンクがクリックされると、JavaServer Faces アクション・メソッドまたは Faces ナビゲーション・ターゲットが起動されます。シグニチャーが String method() である JavaServer Faces アクション・メソッド。
converterID	いいえ	プロパティー値の変換に使用する Faces コンバーター ID。この属性が設定されていない場合、モデルによりこのプロパティーに設定された Faces コンバーター ID が使用されます。
label	いいえ	列のヘッダーのラベルまたはテーブル・ヘッダー行のセルのラベルとして使用されるリテラルまたは値バインディング式。この属性が設定されていない場合、モデルによりこのプロパティーに設定されたラベルが使用されます。
name	はい	この列に表示されるプロパティーの名前。

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。

手順

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、**model** 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。

   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>
```

model 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。Bean クラスは任意の JavaBeans クラスにすることができます。Bean がデフォルト・コンバーターおよびプロパティー・ラベルを提供する場合、そのコンバーターおよびラベルが、List コンポーネントの場合と同じようにしてレンダリングに使用されます。

タスクの結果

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 79. bpe:details 属性

属性	必須	説明
columnClasses	いいえ	列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) のスタイル・クラスをコンマで区切ったリスト。
id	いいえ	コンポーネントの JavaServer Faces ID。
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。
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・クラスのリスト。
styleClass	いいえ	HTML エレメントのレンダリングに使用される CSS クラス。

表 80. bpe:property 属性

属性	必須	説明
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。
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。
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。

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。

このタスクについて

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。CommandBar コンポーネントは、JavaServer Faces (JSF) アプリケーションに追加して拡張することができます。

手順

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実装し、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実装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 クラスによって実装されます。

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。モデルで BPCListHandler クラスも BPCDetailsHandler クラスも使用しない場合は、 ItemProvider インターフェースを実装したクラスを所有する別のオブジェクトを参照する必要があります。

3. カスタム・コマンドを実装するコードを JSF アプリケーションに追加します。

以下のコード断片は、Command インターフェースを実装するコマンド・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。

```
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実装するカスタム・コマンド実装を参照します。つまり、コマンドは public String execute(List

selectedObjects) throws ClientException メソッドを実装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。

d. **commandClass** 属性にコマンド・クラスが指定されていない場合、アクションが即時に呼び出されます。例えば、例の中のリフレッシュ・コマンドの場合、JSF 値式 `#{TaskInstanceList.refreshList}` がコマンドの代わりに呼び出されます。

タスクの結果

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・バーを実装する JavaServer ページを含むようになります。

コマンドの処理方法

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。

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実装を示します。
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実装すると、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。

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 81. bpe:commandbar 属性

属性	必須	説明
buttonStyleClass	いいえ	コマンド・バー内のボタンのレンダリングに使用されるカスケーディング・スタイル・シート (CSS) スタイル・クラス。
id	いいえ	コンポーネントの JavaServer Faces ID。
model	はい	ItemProvider インターフェースを実装する管理対象 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。

表 81. bpe:commandbar 属性 (続き)

属性	必須	説明
styleClass	いいえ	コマンド・バーのレンダリングに使用される CSS スタイル・クラス。

表 82. bpe:command 属性

属性	必須	説明
action	いいえ	JavaServer Faces アクション・メソッドまたはコマンド・ボタンにより起動される Faces ナビゲーション・ターゲット。アクションにより戻されるナビゲーション・ターゲットは、その他のナビゲーション・ルールをすべて上書きします。このアクションは、例外がスローされない場合、またはコマンドから ErrorsInCommandException 例外がスローされる場合に呼び出されます。
commandClass	いいえ	コマンド・クラスの名前。このクラスのインスタンスは CommandBar コンポーネントにより作成され、コマンド・ボタンが選択されると実行されます。
commandID	はい	コマンドの ID。
context	いいえ	**commandClass** 属性を使用して指定されたコマンドのコンテキストを提供するオブジェクト。コマンド・バーが初めてアクセスされると、コンテキスト・オブジェクトが取得されます。
immediate	いいえ	コマンドが起動される時期を指定します。この属性の値が `true` の場合は、ページの入力が処理される前にコマンドが起動されます。デフォルトは `false` です。
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。
rendered	いいえ	ボタンがレンダリングされるかどうかを判別します。属性の値は、ブール値または値の式のいずれかにすることができます。
styleClass	いいえ	ボタンのレンダリングに使用される CSS スタイル・クラス。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。

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。

このタスクについて

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。

手順

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実装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実装します。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・ハンドラーの itemListener リスト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。

タスクの結果

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 83. bpe:form 属性

属性	必須	説明
id	いいえ	コンポーネントの JavaServer Faces ID。
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。
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。
simplification	いいえ	この属性が true に設定されている場合は、単純型が含まれているプロパティーおよび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。
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。
styleClass4invalidInput	いいえ	無効な入力のレンダリング用の CSS スタイル・クラス名。

表 83. bpe:form 属性 (続き)

属性	必須	説明
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。
styleClass4validInput	いいえ	有効な入力のレンダリング用の CSS スタイル・クラス名。

第 15 章 タスクおよびプロセス・メッセージ用の JSP ページの開発

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 ページを使用して、カスタマイズされた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。

このタスクについて

ユーザー定義の JavaServer Pages (JSP) ページを 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ページを指定する必要があります。例えば、JSP ページの指定先は、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 ページは、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、メッセージのすべてのラベルや入力フィールドのフラグメントがこれに該当します。

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。情報はマップとして提供され、このマップは、無効なプロパティーの XML Path Expression (XPath) を、発生した妥当性検査例外にマップします。

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。

手順

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 ページは、メッセージ・データにアクセスする必要があります。 JSP 内の Java 断片または JSP 実行言語を使用して、メッセージ・データにアクセスします。フォーム内のデータは要求コンテキストを介して使用可能です。
2. JSP ページ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 ページの場所と、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 ページを 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。JSP が個別にデプロイされる場合、Business Process Choreographer Explorer またはカスタム・クライアントがデプロイされるサーバー上で JSP を使用可能にしてください。

   プロセスおよびタスク・メッセージにカスタム JSP を使用している場合、JSP をデプロイするために使用される Web モジュールを、カスタム JSF クライアントがマップされるのと同じサーバーにマップする必要があります。

タスクの結果

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 JavaServer Pages (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>
```

第 16 章 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担当者照会結果の後処理用にカスタマイズしたプラグインを作成することができます。

このタスクについて

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、担当者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。

プラグインを使用するには、プラグインを事前にインストールし、登録する必要があります。担当者照会結果を後処理するためのプラグインを TaskContainer アプリケーションに登録できます。これにより、プラグインはすべてのタスクで使用できるようになります。

Business Process Choreographer 用の 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。

このタスクについて

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。

手順

1. APIEventHandlerPlugin5 インターフェースを実装するクラス、または APIEventHandler 実装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。
   - APIEventHandlerPlugin5 インターフェースを使用する場合は、APIEventHandlerPlugin5 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実装する必要があります。
   - APIEventHandler 実装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 Platform, Enterprise Edition (Java EE) Enterprise 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。

注: 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。このアクションを行うと、データベース内のタスク・データが不整合になる可能性があります。

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。

   以下のいずれかの方法で JAR ファイルを使用可能にすることができます。

   - アプリケーション EAR ファイル内のユーティリティー JAR ファイルとして使用可能にする。
   - アプリケーション EAR ファイルと共にインストールされる共用ライブラリーとして使用可能にする。
   - TaskContainer アプリケーションと共にインストールされる共用ライブラリーとして使用可能にする。この場合、プラグインはすべてのタスクで使用できるようになります。

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 EE 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.APIEventHandlerPlugin5 インターフェースを実装する場合、構成ファイルの名前は `com.ibm.task.spi.`**`Customer`**`APIEventHandlerPlugin` になります。

   b. このファイル内のコメント行 (番号記号 (#) で始まる行) とブランク行を除く最初の行で、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。

タスクの結果

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。

注: API イベント・ハンドラーおよび通知イベント・ハンドラーの両方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・ハンドラーを両方使用する場合、プラグイン実装は同じ名前でなければなりません (例えば、SPI 実装のイベント・ハンドラー名として `Customer` など)。

いずれのプラグインも、単一のクラスまたは 2 つのクラスを使用して実装できます。いずれの場合も、JAR ファイルの META-INF/services/ ディレクトリーに 2 つのファイルを作成する必要があります (com.ibm.task.spi.CustomerNotificationEventHandlerPlugin と com.ibm.task.spi.CustomerAPIEventHandlerPlugin など)。

プラグインの実装とヘルパー・クラスを単一の JAR ファイルにパッケージします。

実装に対する変更を有効にするには、共用ライブラリー内の JAR ファイルを置き換えて、関連 EAR ファイルを再デプロイし、サーバーを再始動します。

次のタスク

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 メソッド実装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、処理を続行できないようにします。

API イベント・ハンドラーを実装するには、APIEventHandlerPlugin インターフェースを拡張する APIEventHandlerPlugin3 インターフェースを実装するか、デフォルトの com.ibm.task.spi.APIEventHandler SPI 実装クラスを拡張します。イベント・ハンドラーは、デフォルトの実装クラスから継承される場合、常に最新バージョンの SPI を実装します。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。

Business Process Choreographer 用の通知イベント・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。

このタスクについて

通知イベント・ハンドラーを実装するには、NotificationEventHandlerPlugin インターフェースを実装する方法と、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) 実装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。

手順

1. NotificationEventHandlerPlugin インターフェースを実装するクラス、または NotificationEventHandler 実装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・メソッドを実装する必要があります。SPI 実装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 Platform, Enterprise Edition (Java EE) Enterprise 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。

   注: 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。このアクションを行うと、データベース内のタスク・データが不整合になる可能性があります。

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。

   以下のいずれかの方法で JAR ファイルを使用可能にすることができます。
   - アプリケーション EAR ファイル内のユーティリティー JAR ファイルとして使用可能にする。
   - アプリケーション EAR ファイルと共にインストールされる共用ライブラリーとして使用可能にする。
   - TaskContainer アプリケーションと共にインストールされる共用ライブラリーとして使用可能にする。この場合、プラグインはすべてのタスクで使用できるようになります。
3. 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 Java EE 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。
4. 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 EE 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実装する場合、構成ファイルの名前は `com.ibm.task.spi.`**`HelpDeskRequest`**`NotificationEventHandlerPlugin` になります。

   b. このファイル内のコメント行 (番号記号 (#) で始まる行) とブランク行を除く最初の行で、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。

タスクの結果

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。

注: API イベント・ハンドラーおよび通知イベント・ハンドラーの両方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・ハンドラーを両方使用する場合、プラグイン実装は同じ名前でなければなりません (例えば、SPI 実装のイベント・ハンドラー名として Customer など)。

いずれのプラグインも、単一のクラスまたは 2 つのクラスを使用して実装できます。いずれの場合も、JAR ファイルの META-INF/services/ ディレクトリーに 2 つのファイルを作成する必要があります (com.ibm.task.spi.CustomerNotificationEventHandlerPlugin と com.ibm.task.spi.CustomerAPIEventHandlerPlugin など)。

プラグインの実装とヘルパー・クラスを単一の JAR ファイルにパッケージします。

実装に対する変更を有効にするには、共用ライブラリー内の JAR ファイルを置き換えて、関連 EAR ファイルを再デプロイし、サーバーを再始動します。

次のタスク

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。

ヒューマン・タスク用の API イベント・ハンドラーおよび通知イベント・ハンドラーのプラグインのインストール

API イベント・ハンドラーまたは通知イベント・ハンドラーのプラグインを使用するには、ヒューマン・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。

このタスクについて

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 Platform, Enterprise Edition (Java EE) 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。

手順

- 単一の Java EE 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 Java EE 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 Java EE 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 Network Deployment 環境で使用できるようにするには、アプリケーションがデプロイされるサーバーまたはクラスター・メンバーをホストする各ノードに手動で JAR ファイルを配布します。アプリケーションのデプロイメント・ターゲット・スコープ (アプリケーションがデプロイされるサーバーまたはクラスター)、またはセル・スコープを使用できます。これによりプラグイン・クラスは、選択されたデプロイメント・スコープ全体で可視になることに注意してください。

次のタスク

これで、プラグインを登録することができます。

API イベント・ハンドラーおよび通知イベント・ハンドラーのプラグインをタスク・テンプレート、タスク・モデル、およびタスクに登録する

API イベント・ハンドラーおよび通知イベント・ハンドラーのプラグインをタスク、タスク・テンプレート、およびタスク・モデルに登録できます。この登録は、臨時タスクの作成、既存タスクの更新、臨時タスク・モデルの作成、またはタスク・テンプレートの定義のときなど、さまざまなタイミングで行うことができます。

このタスクについて

API イベント・ハンドラーおよび通知イベント・ハンドラーのプラグインを以下のレベルのタスクに登録できます。

タスク・テンプレート
このテンプレートを使用して作成されるタスクはすべて同じハンドラーを使用します。

臨時タスク・モデル
このモデルを使用して作成されるタスクは同じハンドラーを使用します。

随時タスク
作成されたタスクは、指定されたハンドラーを使用します。

既存タスク
タスクは、指定されたハンドラーを使用します。

以下のいずれかの方法で、プラグインを登録できます。

手順

- WebSphere Integration Developer でモデル化されたタスク・テンプレートの場合は、タスク・モデルにプラグインを指定します。
- 臨時タスクまたは臨時タスク・モデルの場合は、タスクまたはタスク・モデルを作成するときにプラグインを指定します。

  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。

担当者照会結果の後処理を行うプラグインの使用

Business Process Choreographer の担当者解決は、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返します。担当者解決で返される担当者照会の結果を変更するプラグインを作成できます。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削除できます。

このタスクについて

担当者割り当ておよび担当者の代替によって返される結果を変更するには、プラグイン・インターフェースを実装するクラスを作成し、そのプラグインの JAR ファイルをアセンブルして、それをインストールしアクティブにする必要があります。

担当者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。

手順

1. 担当者照会結果の後処理プラグインを実装します。
   StaffQueryResultPostProcessorPlugin インターフェースまたは StaffQueryResultPostProcessorPlugin2 インターフェースのいずれかを実装するクラスを作成します。
2. インストール可能な JAR ファイルを作成します。
   a. 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。
   b. 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 EE サービス・プロバイダー・インターフェース仕様に準拠している必要があります。
      1) テキスト・エディターで、`com.ibm.task.spi.`*`plug-in_name`*`StaffQueryResultPostProcessorPlugin` という名前のサービス・プロバイダー構成ファイルを作成します。*plug-in_name* はプラグインの名前です。構成ファイルの名前は、実装したインターフェースの名前に依存しません。例えば、`MyHandler` という名前のプラグインが com.ibm.task.spi.StaffQueryResultPostProcessorPlugin2 インターフェースを実装する場合でも、構成ファイルの名前は `com.ibm.task.spi.`**`MyHandler`**`StaffQueryResultPostProcessorPlugin` になります。
      2) このファイルの、コメント行 (番号記号 (#) で始まる行) でもブランク行でもない最初の行で、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。

担当者照会結果を後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。

3. JAR ファイルをアプリケーション・サーバーの共用ライブラリーにインストールし、それを Human Task Manager アプリケーションと関連付けます。
   a. Business Process Choreographer が構成されているサーバーまたはクラスターのスコープで、プラグイン用の WebSphere Application Server 共用ライブラリーを定義します。
   b. 共用ライブラリーを TaskContainer アプリケーションと関連付けます。
   c. 影響を受けるそれぞれの WebSphere Process Server インストール済み環境 (サーバーまたはクラスター・メンバーをホストするもの) で、プラグイン JAR ファイルを使用できるようにします。
4. プラグインを使用するように Human Task Manager を構成します。
   a. 管理コンソールで、Human Task Manager の「カスタム・プロパティー」ページに移動します。

      「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」をクリックします。「**追加プロパティー**」の下の「**カスタム・プロパティー**」を選択します。
   b. プラグインに付けた名前 (例えば MyHandler) を値として持つ、**Staff.PostProcessorPlugin** という名前のカスタム・プロパティーを追加します。

   プラグインを、担当者照会結果の後処理に使用できるようになりました。
5. プラグインを有効にするために、サーバーを再始動します。 後処理プラグインは、担当者の割り当てと担当者の代替の両方を実行した後に呼び出されます。

   注: プラグインを変更する場合は、共用ライブラリー内の JAR ファイルを置き換えて、サーバーを再始動する必要があります。

第 17 章 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) 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。

始める前に

アプリケーションをインストール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、 Business Flow Manager と Human Task Manager がインストールされ、構成されていることを確認します。

このタスクについて

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。

タスクの結果

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・アプリケーションがインストールされると、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・テンプレートは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。

次のタスク

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。

Network Deployment 環境へのビジネス・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方法

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 Network Deployment 環境にインストールするときに、アプリケーションのインストール機能により以下のアクションが自動的に実行されます。

アプリケーションのインストールは段階別に実行されます。次の段階を開始する前に、前の段階が正常に完了している必要があります。

1. アプリケーションのインストールは Deployment Manager で開始されます。

   この段階では、ビジネス・プロセス・テンプレートとヒューマン・タスク・テンプレートが、WebSphere 構成リポジトリーで構成されます。アプリケーションの検証も行われます。エラーが発生した場合、エラーは System.out ファイルまたは System.err ファイルに報告されるか、あるいは Deployment Manager で FFDC エントリーとして報告されます。

2. アプリケーションのインストールはノード・エージェントで続行されます。

   この段階では、1 つのアプリケーション・サーバー・インスタンスでのアプリケーションのインストールが開始されます。このアプリケーション・サーバー・インスタンスは、デプロイメント・ターゲットであるか、またはその一部です。デプロイメント・ターゲットが、複数のクラスター・メンバーで構成されるクラスターの場合は、このクラスターのクラスター・メンバーからサーバー・インスタンスが任意に選択されます。この段階でエラーが発生した場合、エラーは SystemOut.log ファイルまたは SystemErr.log ファイルに報告されるか、あるいはノード・エージェントで FFDC エントリーとして報告されます。

3. サーバー・インスタンスでアプリケーションが実行されます。

   この段階では、プロセス・テンプレートとヒューマン・テンプレートがデプロイメント・ターゲットの Business Process Choreographer データベースに配置されます。エラーが発生した場合、デプロイメント・マネージャーの SDSF ジョブ・データ・セットに報告されます。

ビジネス・プロセスとヒューマン・タスクのデプロイメント

WebSphere Integration Developer または serviceDeploy を使用して、プロセス・コンポーネントまたはタスク・コンポーネントをエンタープライズ・アプリケーション (EAR) ファイルにパッケージ化します。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・プロセスまたはヒューマン・タスクが含まれてい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、これらのビジネス・プロセスまたはヒューマン・タスクは、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・テンプレートとして Business Process Choreographer データベースに格納されます。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。

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。バージョンは、その有効開始日によって区別されます。新規エンタープライズ・アプリケーションのインストール時に、インストールされるテンプレートのバージョンが次のように決定されます。

- テンプレートの名前とターゲット名前空間がまだ存在していない場合は、新規テンプレートがインストールされます。
- テンプレート名とターゲット名前空間が、既存のテンプレートと同じであるが、有効開始日が異なる場合は、既存のテンプレートの新規バージョンがインストールされます。

注: テンプレート名は、ビジネス・プロセスまたはヒューマン・タスクではなく、コンポーネントの名前から派生します。

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。

- WebSphere Integration Developer を使用する場合、有効開始日はヒューマン・タスクまたはビジネス・プロセスがモデル化された日付になります。
- サービス・デプロイメントを使用する場合、有効開始日は、serviceDeploy コマンドが実行された日付になります。アプリケーションがインストールされた日付を有効開始日として取得するのは、コラボレーション・タスクだけです。

ビジネス・プロセス・アプリケーションおよびヒューマン・タスク・アプリケーションの対話式インストール

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。

このタスクについて

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。

手順

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。

このタスクについて

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。

手順

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。
```

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

管理コンソールを使用すると、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できます。

始める前に

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には、以下の条件を適用する必要があります。

- アプリケーションがスタンドアロン・サーバーにインストールされている場合は、そのサーバーが稼働しており、Business Process Choreographer データベースへのアクセス権限を持っている必要があります。
- アプリケーションがクラスターにインストールされている場合は、デプロイメント・マネージャーおよび少なくとも 1 つのクラスター・メンバーが稼働している必要があります。クラスター・メンバーは、Business Process Choreographer データベースへのアクセス権限を所有している必要があります。
- アプリケーションが管理対象サーバーにインストールされている場合は、デプロイメント・マネージャーおよび管理対象サーバーが稼働している必要があります。このサーバーには Business Process Choreographer データベースへのアクセス権限が必要です。
- いずれの状態においてもビジネス・プロセス・テンプレートまたはヒューマン・タスク・テンプレートのインスタンスはありません。
- プロセス・インスタンスが新しいバージョンのプロセスにマイグレーションされたが、サービス呼び出しの応答を待機している場合、以前のバージョンを含むアプリケーションは、応答が受信されるまでアンインストールできません。それ以外のすべての場合は、マイグレーションされたインスタンスは新しいバージョンのインスタンスであると見なされ、古いバージョンのプロセスを含むアプリケーションをアンインストールできます。

このタスクについて

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。

手順

1. 管理コンソールで、「**アプリケーション**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere エンタープライズ・アプリケーション**」をクリックします。
2. 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。アプリケーションのアンインストール前に、Business Process Choreographer Explorer を使用してインスタンスを削除するか、bpcTamplates.jacl 管理スクリプトの **-force** オプションを使用して、これらのインスタンスを停止して削除することができます。

3. アンインストールするアプリケーションを選択し、「**アンインストール**」をクリックします。
4. 「**保管**」をクリックして、変更を保管します。

タスクの結果

アプリケーションはアンインストールされます。

関連タスク

『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』

bpcTemplates.jacl スクリプトの使用には、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。

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

bpcTemplates.jacl スクリプトの使用には、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。

始める前に

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には、以下の条件を適用する必要があります。

- アプリケーションがスタンドアロン・サーバーにインストールされている場合は、そのサーバーが稼働しており、Business Process Choreographer データベースへのアクセス権限を持っている必要があります。
- アプリケーションがクラスターにインストールされている場合は、デプロイメント・マネージャーおよび少なくとも 1 つのクラスター・メンバーが稼働している必要があります。クラスター・メンバーは、Business Process Choreographer データベースへのアクセス権限を所有している必要があります。
- アプリケーションが管理対象サーバーにインストールされている場合は、デプロイメント・マネージャーおよび管理対象サーバーが稼働している必要があります。このサーバーには Business Process Choreographer データベースへのアクセス権限が必要です。
- 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。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。
- WebSphere 管理セキュリティーが使用可能になっていて、ご使用のユーザー ID にオペレーター権限も管理者権限もない場合は、wsadmin `-user` および

-password のオプションを付けて、オペレーターまたは管理者の権限を持つユーザー ID を指定します。-uninstall オプションを使用するにはオペレーター権限が必要であり、-force オプションを使用するには管理者権限が必要です。

- 次のうち、1 つ以上が当てはまります。
  - いずれの状態においてもビジネス・プロセス・テンプレートまたはヒューマン・タスク・テンプレートのインスタンスはありません。
  - **-force** オプションを使用する予定です。
- プロセス・インスタンスが新しいバージョンのプロセスにマイグレーションされたが、サービス呼び出しの応答を待機している場合、以前のバージョンを含むアプリケーションは、応答が受信されるまでアンインストールできません。それ以外のすべての場合は、マイグレーションされたインスタンスは新しいバージョンのインスタンスであると見なされ、古いバージョンのプロセスを含むアプリケーションをアンインストールできます。

このタスクについて

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。

手順

1. アンインストールするアプリケーションのテンプレートにプロセス・インスタンスまたはタスク・インスタンスがまだ関連付けられている場合は、以下のいずれか一方または両方を実行します。
   - Business Process Choreographer Explorer を使用して、インスタンスを削除します。
   - アンインストールするアプリケーションで定義されたプロセス・テンプレートに依存するビジネス・プロセスがほかに存在しないことが確実であれば、**-force** オプションを使用できます。

     注意:
     このオプションと共にスクリプトを使用すると、テンプレートに関連付けられたすべてのインスタンスと実行中のインスタンスに関連付けられたすべてのデータの削除、テンプレートの停止、およびアプリケーションのアンインストールを 1 つのステップで実行します。このオプションを使用する場合は、細心の注意を払ってください。
2. 管理スクリプトがある 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
   ```
3. 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.sh -f bpcTemplates.jacl
                                -uninstall application_name]
                               [-force]
   ```

   各部の意味は、次のとおりです。

   -uninstall *application_name*
   これにより、アンインストールするアプリケーションの名前を指定します。

-force

このオプションにより、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。このオプションを使用する場合は注意が必要です。それは、このオプションを使用すると、実行中のインスタンスに関連するすべてのデータも削除されるからです。

タスクの結果

アプリケーションはアンインストールされます。

関連タスク

681 ページの『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』
管理コンソールを使用すると、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できます。

第 5 部 ビジネス・プロセスとタスクのモニター

第 18 章 ビジネス・プロセスとヒューマン・タスクのモニター

始める前に

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。

このタスクについて

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、モニター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省略します。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。

第 19 章 ビジネス・プロセス・イベントの概要

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。

イベントは、以下のいずれかの形式をとることができます。

WebSphere Business Monitor 6.1、6.2、または 7.0 形式 (スキーマがサポートされた XML)

WebSphere Integration Developer 6.1 以降でモデル化されたプロセスがあり、この形式が選択されている場合は、この形式でイベントが生成されます。

これらのイベントのオブジェクト固有の内容は、Common Base Event (CBE) の eventPointData 部分の `xs:any` スロットに XML エレメントとして書き込まれます。また、有効搭載量のメッセージは *applicationData* セクションに書き込まれます。XML の構造は、スキーマ定義ファイル *install_root*¥ProcessChoreographer¥client¥BFMEvents.xsdで定義されています。 CBE 情報を解析および検証するには、*install_root*¥ProcessChoreographer¥client¥WBIEvent.xsd のスキーマ定義を使用します。

WebSphere Business Monitor 6.0.2 形式 (レガシー XML)

WebSphere Integration Developer 6.0.2 でモデル化されたプロセスがある場合、または WebSphere Integration Developer 6.1 以降において WebSphere Business Monitor 6.0.2 形式が選択されている場合は、この形式でイベントが生成されます。特に明記されていない限り、これらのイベントの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。

レガシー hexBinary

WebSphere Integration Developer でこの形式が選択されている場合は、この形式でイベントが生成されます。

関連資料

709 ページの『ビジネス・プロセス・イベント』
ビジネス・プロセスの Common Base Event は、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に発行されます。プロセスによって、プロセス・イベント、アクティビティー・イベント、アクティビティー・スコープ・イベント、リンク・イベント、および変数イベントが発行される場合があります。

710 ページの『ビジネス・プロセスの Common Base Events』
ビジネス・プロセスの Common Base Event は、WebSphere Integration Developer 内のビジネス・プロセスのモニターが要求された場合に発行されます。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。 これらのイベントは監査ログにも書き込まれます。

716 ページの『アクティビティーの Common Base Event』
アクティビティーの Common Base Event は、WebSphere Integration Developer 内のアクティビティーのモニターが要求された場合に発行されます。ここでは、アクティビティー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。 これらのイベントは監査ログにも書き込まれます。

726 ページの『scope アクティビティーの Common Base Event』
scope アクティビティーの Common Base Event は、WebSphere Integration Developer 内のこれらのアクティビティーのモニターが要求された場合に発行されます。ここでは、アクティビティー・スコープ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。 これらのイベントは監査ログにも書き込まれます。

730 ページの『flow アクティビティーのリンクの Common Base Event』
リンクの Common Base Event は、 WebSphere Integration Developer 内のリンクが定義されている flow アクティビティーのモニターが要求された場合に発行されます。ここでは、リン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。これらのイベントは監査ログにも書き込まれます。

731 ページの『プロセス変数の Common Base Events』
プロセス変数の Common Base Event は、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に発行されます。ここでは、変数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。 これらのイベントは監査ログにも書き込まれます。

ビジネス・プロセス・イベントの拡張子名

拡張子名は、イベントの有効搭載量を示します。ビジネス・プロセス・イベントのすべての拡張子名と、対応する有効搭載量のリストがここに表示されます。

拡張子名には、Common Base Event (CBE) の *extensionName* 属性の値として使用されるストリング値が含まれています。これは、イベントに関する追加データを提供する XML エレメントの名前でもあります。イベント・エレメントの名前は大文字 (例: BPC.BFM.BASE) で、XML エレメントの名前は大/小文字混合 (例: *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。

ビジネス・プロセス・イベントに使用できる拡張子名を以下に示します。

BPC.BFM.ACTIVITY

- 692 ページの『BPC.BFM.ACTIVITY.BASE』
- 694 ページの『BPC.BFM.ACTIVITY.CHILD_PROCESS_TERMINATING』
- 694 ページの『BPC.BFM.ACTIVITY.CLAIM』
- 694 ページの『BPC.BFM.ACTIVITY.CONDITION』
- 695 ページの『BPC.BFM.ACTIVITY.CUSTOMPROPERTYSET』
- 695 ページの『BPC.BFM.ACTIVITY.ESCALATED』
- 696 ページの『BPC.BFM.ACTIVITY.EVENT』
- 696 ページの『BPC.BFM.ACTIVITY.FAILURE』
- 696 ページの『BPC.BFM.ACTIVITY.FOREACH』
- 696 ページの『BPC.BFM.ACTIVITY.JUMPED』
- 697 ページの『BPC.BFM.ACTIVITY.MESSAGE』
- 697 ページの『BPC.BFM.ACTIVITY.SKIP_ON_EXIT_CONDITION_TRUE』
- 697 ページの『BPC.BFM.ACTIVITY.SKIP_REQUESTED』
- 698 ページの『BPC.BFM.ACTIVITY.SKIPPED_ON_REQUEST』
- 698 ページの『BPC.BFM.ACTIVITY.STATUS』
- 698 ページの『BPC.BFM.ACTIVITY.TIMER_RESCHEDULED』
- 699 ページの『BPC.BFM.ACTIVITY.WISTATUS』
- 699 ページの『BPC.BFM.ACTIVITY.WITRANSFER』

BPC.BFM.BASE

- 700 ページの『BPC.BFM.BASE』

BPC.BFM.LINK

- 700 ページの『BPC.BFM.LINK.STATUS』

BPC.BFM.PROCESS

- 701 ページの『BPC.BFM.PROCESS.BASE』
- 701 ページの『BPC.BFM.PROCESS.CORREL』
- 702 ページの『BPC.BFM.PROCESS.CUSTOMPROPERTYSET』
- 703 ページの『BPC.BFM.PROCESS.ESCALATED』
- 703 ページの『BPC.BFM.PROCESS.EVENT』
- 704 ページの『BPC.BFM.PROCESS.FAILURE』
- 704 ページの『BPC.BFM.PROCESS.MIGRATED』
- 705 ページの『BPC.BFM.PROCESS.MIGRATIONTRIGGERED』
- 706 ページの『BPC.BFM.PROCESS.OWNERTRANSFER』
- 706 ページの『BPC.BFM.PROCESS.PARTNER』
- 707 ページの『BPC.BFM.PROCESS.START』
- 707 ページの『BPC.BFM.PROCESS.STATUS』
- 707 ページの『BPC.BFM.PROCESS.WISTATUS』
- 707 ページの『BPC.BFM.PROCESS.WITRANSFER』

BPC.BFM.VARIABLE

- 708 ページの『BPC.BFM.VARIABLE.STATUS』

BPC.BFM.ACTIVITY.BASE

BPC.BFM.ACTIVITY.BASE は、 700 ページの『BPC.BFM.BASE』の XML エレメントを継承します。

表 84. BPC.BFM.ACTIVITY.BASE の XML エレメント

XML エレメント	説明
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。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 40 - KIND_SCOPE 42 - KIND_SCRIPT 43 - KIND_STAFF 44 - KIND_ASSIGN 45 - KIND_CUSTOM 46 - KIND_RETHROW 47 - KIND_FOR_EACH_SERIAL 49 - KIND_FOR_EACH_PARALLEL 52 - KIND_REPEAT_UNTIL 1000 - SQLSnippet 1001 - RetrieveSet 1002 - InvokeInformationService 1003 - AtomicSQLSnippetSequence

表 84. *BPC.BFM.ACTIVITY.BASE の XML エレメント (続き)*

XML エレメント	説明
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*。アクティビティーの場合、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 11 - STATE_WAITING 12 - STATE_EXPIRED 13 - STATE_STOPPED scope アクティビティーの場合、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_COMPENSATION_FAILED 9 - STATE_FAILING 10 - STATE_SKIPPED 11 - STATE_COMPENSATION_FAILING 12 - STATE_FAULTHANDLER_FAILING 13 - STATE_FINISHING 14 - STATE_STOPPED
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。この名前は、表示名とは異なることがあります。
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。
principal	どのユーザーのために現在のアクションが実行されているかを示すユーザー名。

表 84. *BPC.BFM.ACTIVITY.BASE の XML エレメント (続き)*

XML エレメント	説明
taskInstanceId	関連ヒューマン・タスク・インスタンスの ID。これは、スタッフ・アクティビティー・イベントの場合にのみ組み込まれます。
processTemplateId	プロセス・テンプレートの ID。

BPC.BFM.ACTIVITY.CHILD_PROCESS_TERMINATING

BPC.BFM.ACTIVITY.CHILD_PROCESS_TERMINATING は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 85. *BPC.BFM.ACTIVITY.CHILD_PROCESS_TERMINATING の XML エレメント*

XML エレメント	説明
subState	アクティビティーの副状態。副状態は以下のストリングいずれかにすることができます。 SUB_STATE_NONE SUB_STATE_EXPIRING SUB_STATE_SKIPPING SUB_STATE_RESTARTING SUB_STATE_FINISHING SUB_STATE_FAILING
childProcessInstanceID	子プロセスの ProcessInstanceID。

BPC.BFM.ACTIVITY.CLAIM

BPC.BFM.ACTIVITY.CLAIM は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 86. *BPC.BFM.ACTIVITY.CLAIM の XML エレメント*

XML エレメント	説明
username	タスクが要求されたユーザーの名前。

BPC.BFM.ACTIVITY.CONDITION

BPC.BFM.ACTIVITY.CONDITION は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 87. *BPC.BFM.ACTIVITY.CONDITION の XML エレメント*

XML エレメント	説明
branchBpelId	これは BPEL ファイルでの指定に従って、関連する case エレメントの wpc:id 属性の値に設定されます。この情報は、バージョン 6.1.2 以降をインストールしたプロセスについてのみ提供されます。

表 87. *BPC.BFM.ACTIVITY.CONDITION の XML エレメント (続き)*

XML エレメント	説明
condition	これは XPath 条件についてストリングとして条件を指定します。 (このプロパティーはそれ以外の場合または Java 条件については存在しません。)
isForced	イベントを、forceNavigate API (=true) によって起動するか、他の方法 (=false) で起動するかを指定します。
isOtherwise	これは、otherwise 分岐を入力するか (=true)、または case 分岐を入力するか (=false) を指定します。

BPC.BFM.ACTIVITY.CUSTOMPROPERTYSET

BPC.BFM.ACTIVITY.CUSTOMPROPERTYSET は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 88. *BPC.BFM.ACTIVITY.CUSTOMPROPERTYSET の XML エレメント*

XML エレメント	説明
propertyName	カスタム・プロパティーの名前。
propertyValue	カスタム・プロパティーの値。
associatedObjecID	アクティビティー・インスタンス ID である関連オブジェクトの ID。
associatedObjectName	アクティビティー・テンプレート名である関連オブジェクトの名前。
query	isBinary が true の場合、このエレメントはバイナリー・プロパティーの照会ストリングを指定します。そうでない場合、このエレメントは存在しません。
type	isBinary が true の場合、このエレメントはバイナリー・プロパティーのタイプを指定します。そうでない場合、このエレメントは存在しません。
isBinary	ストリング・カスタム・プロパティーの場合は false に設定し、バイナリー・カスタム・プロパティーの場合は true に設定します。バイナリー・カスタム・プロパティーのペイロード・タイプは Empty に制限されています。プロパティー propertyValue は、バイナリー・カスタム・プロパティーでは省略されます。

BPC.BFM.ACTIVITY.ESCALATED

BPC.BFM.ACTIVITY.ESCALATED は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 89. *BPC.BFM.ACTIVITY.ESCALATED の XML エレメント*

XML エレメント	説明
escalationName	エスカレーションの名前。
operation	これは、インライン呼び出しタスクがエスカレートされるイベント・ハンドラーに関連付けられている操作です。

BPC.BFM.ACTIVITY.EVENT

BPC.BFM.ACTIVITY.EVENT は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 90. *BPC.BFM.ACTIVITY.EVENT の XML エレメント*

XML エレメント	説明
operation	受信したイベントでの操作名。

BPC.BFM.ACTIVITY.FAILURE

BPC.BFM.ACTIVITY.FAILURE は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 91. *BPC.BFM.ACTIVITY.FAILURE の XML エレメント*

XML エレメント	説明
activityFailedException	アクティビティーが失敗する原因となった例外
faultNamespace	障害の名前空間 URI。
faultName	障害のローカル・パーツ。

BPC.BFM.ACTIVITY.FOREACH

BPC.BFM.ACTIVITY.FOREACH は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 92. *BPC.BFM.ACTIVITY.FOREACH の XML エレメント*

XML エレメント	説明
parallelBranchesStarted	開始された分岐の数。

BPC.BFM.ACTIVITY.JUMPED

BPC.BFM.ACTIVITY.JUMPED は、 692 ページの『BPC.BFM.ACTIVITY.BASE』 の XML エレメントを継承します。

表 93. *BPC.BFM.ACTIVITY.JUMPED* の *XML* エレメント

XML エレメント	説明
targetName	ジャンプのターゲット・アクティビティーのアクティビティー・テンプレート名が含まれます。イベントの ECSCurrentId に含まれる aiid は、ジャンプのソース・アクティビティーを指します。

BPC.BFM.ACTIVITY.MESSAGE

BPC.BFM.ACTIVITY.MESSAGE は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 94. *BPC.BFM.ACTIVITY.MESSAGE* の *XML* エレメント

XML エレメント	説明
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**」タブで「**互換性があるイベントのモニター**」オプションが選択されたかどうかによって異なります。 この属性は、WebSphere Business Monitor 6.0.2 形式のイベントでのみ使用されます。WebSphere Business Monitor 6.1 形式のイベントの場合、メッセージの内容が *applicationData* セクションに書き込まれます。このセクションには、名前がメッセージ名に設定されている 1 つのコンテンツ・エレメントが含まれています。

BPC.BFM.ACTIVITY.SKIP_ON_EXIT_CONDITION_TRUE

BPC.BFM.ACTIVITY.SKIP_ON_EXIT_CONDITION_TRUE は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。
BPC.BFM.ACTIVITY.SKIP_ON_EXIT_CONDITION_TRUE について、継承されたプロパティー以外に追加して定義する特定のプロパティーはありません。

BPC.BFM.ACTIVITY.SKIP_REQUESTED

BPC.BFM.ACTIVITY.SKIP_REQUESTED は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 95. *BPC.BFM.ACTIVITY.SKIP_REQUESTED* の *XML* エレメント

XML エレメント	説明
cancel	cancel は、アクティビティーをスキップするかどうかを指定して、スキップ (=false) および cancelSkipRequest (=true) の呼び出しを区別します。

BPC.BFM.ACTIVITY.SKIPPED_ON_REQUEST

BPC.BFM.ACTIVITY.SKIPPED_ON_REQUEST は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。この BPC.BFM.ACTIVITY.SKIPPED_ON_REQUEST について、継承されたプロパティー以外に追加して定義する特定のプロパティーはありません。

BPC.BFM.ACTIVITY.STATUS

BPC.BFM.ACTIVITY.STATUS は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 96. BPC.BFM.ACTIVITY.STATUS の XML エレメント

XML エレメント	説明
reason	停止理由コード。停止理由コードが関係するのは、アクティビティーが停止状態の場合のみです。これは、アクティビティーが停止した理由を示します。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 - STOP_REASON_UNSPECIFIED 2 - STOP_REASON_ACTIVATION_FAILED 3 - STOP_REASON_IMPLEMENTATION_FAILED 4 - STOP_REASON_FOLLOW_ON_NAVIGATION_FAILED 5 - STOP_REASON_EXIT_CONDITION_FALSE

注: ペイロードは、ENTRY イベント性質にペイロードを提供するアクティビティーのイベント性質 FRETRIED で使用可能です。同様に、EXIT イベント性質に対応するイベント性質 FCOMPLETED で使用可能です。具体的には以下のとおりです。

- エレメント種類 invoke および staff の FRETRIED と ENTRY (CREATED を参照)。
- エレメント種類 pick、receive、および reply の FCOMPLETED と EXIT。

ペイロードは、イベントのアプリケーション・データ・セクションで提供されますが、これはバージョン 6.1 イベントの場合に限ります。

BPC.BFM.ACTIVITY.TIMER_RESCHEDULED

BPC.BFM.ACTIVITY.TIMER_RESCHEDULED は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 97. BPC.BFM.ACTIVITY.TIMER_RESCHEDULED の XML エレメント

XML エレメント	説明
timestamp	日時は協定世界時 (UTC) で表記します。形式は *yyyy-MM-dd*[T*hh:mm:ss*] であり、それぞれ年、月、日、T、時、分、秒を表します。

BPC.BFM.ACTIVITY.WISTATUS

BPC.BFM.ACTIVITY.WISTATUS は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 98. BPC.BFM.ACTIVITY.WISTATUS の XML エレメント

XML エレメント	説明
username	作業項目に関連付けられたユーザーの名前。
reason	作業項目の割り当て理由。指定可能な整数値の意味は、以下のとおりです。 1 - REASON_POTENTIAL_OWNER 2 - REASON_EDITOR 3 - REASON_READER 4 - REASON_OWNER 5 - REASON_POTENTIAL_STARTER 6 - REASON_STARTER 7 - REASON_ADMINISTRATOR 9 - REASON_ORIGINATOR 10 - REASON_ESCALATION_RECEIVER 11 - REASON_POTENTIAL_INSTANCE_CREATOR

BPC.BFM.ACTIVITY.WITRANSFER

BPC.BFM.ACTIVITY.WITRANSFER は、 692 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 99. BPC.BFM.ACTIVITY.WITRANSFER の XML エレメント

XML エレメント	説明
current	作業項目の現在の所有者のユーザー名。これは、他のユーザーに転送された作業項目があるユーザーです。
target	作業項目の新規所有者のユーザー名。
reason	作業項目の割り当て理由。指定可能な整数値の意味は、以下のとおりです。 1 - REASON_POTENTIAL_OWNER 2 - REASON_EDITOR 3 - REASON_READER 4 - REASON_OWNER 5 - REASON_POTENTIAL_STARTER 6 - REASON_STARTER 7 - REASON_ADMINISTRATOR 9 - REASON_ORIGINATOR 10 - REASON_ESCALATION_RECEIVER 11 - REASON_POTENTIAL_INSTANCE_CREATOR

BPC.BFM.BASE

BPC.BFM.BASE は、WBIMonitoringEvent の XML エレメントを継承します。

表 100. BPC.BFM.BASE の XML エレメント

XML エレメント	説明
BPCEventCode	イベントの特性を示す Business Process Choreographer イベント・コード。
processTemplateName	プロセス・テンプレートの名前。この名前は、表示名とは異なることがあります。
processTemplateValidFrom	プロセス・テンプレートの有効開始日属性。
eventProgressCounter	イベント進行カウンターは、同じプロセス・インスタンスのすべてのナビゲーション・ステップの実行順序の中での現在のナビゲーション・ステップの位置を示すために使用されます。 イベント進行カウンターは長期実行プロセスの場合に必要であり、イベント・ローカル・カウンターと併用して、同じプロセス・インスタンスに属するイベントの順序 (不完全な可能性もあります) を再作成するために使用できます。microflow では、イベント進行カウンターはゼロに設定されます。
eventLocalCounter	ローカル・カウンターは、同一トランザクション内で発生する 2 つのイベントの順序を検出するときに使用されます。Microflow インスタンスでは、このカウンターによりすべての発行済みイベントの順序が再構成されます。長期実行プロセスの場合、ローカル・カウンターは現行ナビゲーション・トランザクションにおける順序を示します。
processInstanceName	API 呼び出しによって提供されるプロセス・インスタンス名は、それがプロセス・インスタンス ID とは異なる場合にのみ存在します。
processInstanceId	プロセス・インスタンスの ID。

BPC.BFM.LINK.STATUS

BPC.BFM.LINK.STATUS は、『BPC.BFM.BASE』の XML エレメントを継承します。

表 101. BPC.BFM.LINK.STATUS の XML エレメント

XML エレメント	説明
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。

表 101. BPC.BFM.LINK.STATUS の XML エレメント (続き)

XML エレメント	説明
sourceBpelId	ナビゲート対象のリンクに対応するソース・アクティビティーの wpc:id 属性。
targetBpelId	ナビゲート対象のリンクに対応するターゲット・アクティビティーの wpc:id 属性。
isForced	イベントを、forceNavigate API (=true) によって起動するか、他の方法 (=false) で起動するかを指定します。
processTemplateId	プロセス・テンプレートの ID。

BPC.BFM.PROCESS.BASE

BPC.BFM.PROCESS.BASE は、 700 ページの『BPC.BFM.BASE』の XML エレメントを継承します。

表 102. BPC.BFM.PROCESS.BASE の XML エレメント

XML エレメント	説明
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*> 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 11 - STATE_SUSPENDED 12 - STATE_COMPENSATION_FAILED
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。
principal	このイベントに関連したユーザーの名前。

BPC.BFM.PROCESS.CORREL

BPC.BFM.PROCESS.CORREL は、『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 103. BPC.BFM.PROCESS.CORREL の XML エレメント

XML エレメント	説明

correlationSet	これは hexBinary ストリングです。これをストリングに変換すると、以下の形式になります。
	`<?xml version="1.0"?>`
	`<correlationSet name="correlation_set_name">`
	`  <property name="property_name"`
	`  value="property_value"/>*`
	`</correlationSet>`

action	次のいずれかのストリングが含まれます。 **init** 相関セット・プロパティー *correlationSet* が初期化されたことを示します。 **set** API を使用して相関セット・プロパティー *correlationSet* の値が設定されたことを示します。 **unset** API を使用して相関セット・プロパティー *correlationSet* が削除または設定解除されたため、このプロパティーには値が含まれていないことを示します。

BPC.BFM.PROCESS.CUSTOMPROPERTYSET

BPC.BFM.PROCESS.CUSTOMPROPERTYSET は、 701 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 104. BPC.BFM.PROCESS.CUSTOMPROPERTYSET の XML エレメント

XML エレメント	説明
propertyName	カスタム・プロパティーの名前。
propertyValue	カスタム・プロパティーの値。
associatedObjectID	プロセス・インスタンス ID である関連オブジェクトの ID。
associatedObjectName	プロセス・テンプレート名である関連オブジェクトの名前。
query	isBinary が true の場合、このエレメントはバイナリー・プロパティーの照会ストリングを指定します。そうでない場合、このエレメントは存在しません。
type	isBinary が true の場合、このエレメントはバイナリー・プロパティーのタイプを指定します。そうでない場合、このエレメントは存在しません。

表 104. BPC.BFM.PROCESS.CUSTOMPROPERTYSET の XML エレメント (続き)

XML エレメント	説明
isBinary	ストリング・カスタム・プロパティーの場合は false に設定し、バイナリー・カスタム・プロパティーの場合は true に設定します。バイナリー・カスタム・プロパティーのペイロード・タイプは Empty に制限されています。プロパティー propertyValue は、バイナリー・カスタム・プロパティーでは省略されます。

BPC.BFM.PROCESS.ESCALATED

BPC.BFM.PROCESS.ESCALATED は、 701 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 105. BPC.BFM.PROCESS.ESCALATED の XML エレメント

XML エレメント	説明
escalationName	エスカレーションの名前。
operation	これは、インライン呼び出しタスクがエスカレートされるイベント・ハンドラーに関連付けられている操作です。
portTypeName	インライン呼び出しタスクがエスカレートされるイベント・ハンドラーに関連付けられている操作のポート・タイプ名です。
portTypeNamespace	インライン呼び出しタスクがエスカレートされるイベント・ハンドラーに関連付けられている操作のポート・タイプ名前空間です。

BPC.BFM.PROCESS.EVENT

BPC.BFM.PROCESS.EVENT は、 701 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 106. BPC.BFM.PROCESS.EVENT の XML エレメント

XML エレメント	説明
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メッセージ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**」タブで「**互換性があるイベントのモニター**」オプションが選択されたかどうかによって異なります。 この属性は、WebSphere Business Monitor 6.0.2 形式のイベントでのみ使用されます。WebSphere Business Monitor 6.1 形式のイベントの場合、メッセージの内容が *applicationData* セクションに書き込まれます。このセクションには、名前がメッセージ名に設定されている 1 つのコンテンツ・エレメントが含まれています。
operation	受信したイベントでの操作名。

表 106. BPC.BFM.PROCESS.EVENT の XML エレメント (続き)

XML エレメント	説明
portTypeName	イベント・ハンドラーに関連付けられている操作のポート・タイプ名です。
portTypeNamespace	イベント・ハンドラーに関連付けられている操作のポート・タイプ名前空間です。

BPC.BFM.PROCESS.FAILURE

BPC.BFM.PROCESS.FAILURE は、 701 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 107. BPC.BFM.PROCESS.FAILURE の XML エレメント

XML エレメント	説明
processFailedException	プロセスの失敗につながる例外メッセージ。
faultNamespace	障害の名前空間 URI。
faultName	障害のローカル・パーツ。

BPC.BFM.PROCESS.MIGRATED

BPC.BFM.PROCESS.MIGRATED は、 701 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 108. BPC.BFM.PROCESS.MIGRATED の XML エレメント

XML エレメント	説明
migratedFromPTID	マイグレーション元となるプロセス・テンプレートの ID。
migratedFromValidFrom	マイグレーション元となるプロセス・テンプレートの validFrom 日付。
マイグレーション後は、プロセス内のアクティビティー・インスタンスについての情報が Common Base Event (CBE) のアプリケーション・データ・セクションのビジネス・オブジェクト (BO) に提供されます。このビジネス・オブジェクトは *install_root*/ProcessChoreographer/client/BFMEvent_Data_V7.xsd ファイルに定義されます。 ビジネス・オブジェクトには、マイグレーションされた各アクティビティーの以下の情報が含まれます。	
activityInstanceID	アクティビティー・インスタンスの ID。
activityState	アクティビティーの現在の実行状態。形式は次のとおりです。*state_code-state_name*
activitySubState	アクティビティーの現在の実行副状態。形式は次のとおりです。*substate_code-substate_name*

表 108. BPC.BFM.PROCESS.MIGRATED の XML エレメント (続き)

XML エレメント	説明
activityStopReason	停止理由コード。停止理由コードが関係するのは、アクティビティーが停止状態の場合のみです。これは、アクティビティーが停止した理由を示します。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 - STOP_REASON_UNSPECIFIED 2 - STOP_REASON_ACTIVATION_FAILED 3 - STOP_REASON_IMPLEMENTATION_FAILED 4 - STOP_REASON_FOLLOW_ON_NAVIGATION_FAILED 5 - STOP_REASON_EXIT_CONDITION_FALSE
bpelId	アクティビティーの BPEL ID。
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 ID。

BPC.BFM.PROCESS.MIGRATIONTRIGGERED

BPC.BFM.PROCESS.MIGRATIONTRIGGERED は、 701 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 109. BPC.BFM.PROCESS.MIGRATIONTRIGGERED の XML エレメント

XML エレメント	説明
migrateToPTID	マイグレーション先となるプロセス・テンプレートの ID。
migrateToValidFrom	マイグレーション先となるプロセス・テンプレートの validFrom 日付。
マイグレーション前は、プロセス内のアクティビティー・インスタンスについての情報が Common Base Event (CBE) のアプリケーション・データ・セクションのビジネス・オブジェクト (BO) に提供されます。このビジネス・オブジェクトは、*install_root*/ProcessChoreographer/client/BFMEvent_Data_V7.xsdファイルに定義されます。 ビジネス・オブジェクトには、マイグレーションされる各アクティビティーの以下の情報が含まれます。	
activityInstanceID	アクティビティー・インスタンスの ID。
activityState	アクティビティーの現在の実行状態。形式は次のとおりです。*state_code-state_name*
activitySubState	アクティビティーの現在の実行副状態。形式は次のとおりです。*substate_code-substate_name*

表 109. BPC.BFM.PROCESS.MIGRATIONTRIGGERED の XML エレメント (続き)

XML エレメント	説明
activityStopReason	停止理由コード。停止理由コードが関係するのは、アクティビティーが停止状態の場合のみです。これは、アクティビティーが停止した理由を示します。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 - STOP_REASON_UNSPECIFIED 2 - STOP_REASON_ACTIVATION_FAILED 3 - STOP_REASON_IMPLEMENTATION_FAILED 4 - STOP_REASON_FOLLOW_ON_NAVIGATION_FAILED 5 - STOP_REASON_EXIT_CONDITION_FALSE
bpelId	アクティビティーの BPEL ID。
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 ID。

BPC.BFM.PROCESS.OWNERTRANSFER

BPC.BFM.PROCESS.OWNERTRANSFER は、 701 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 110. BPC.BFM.PROCESS.OWNERTRANSFER の XML エレメント

XML エレメント	説明
current	プロセスの現在の所有者のユーザー名。これは、他のユーザーに転送されるプロセスを持つユーザーです。
target	プロセスの新規所有者のユーザー名。

BPC.BFM.PROCESS.PARTNER

BPC.BFM.PROCESS.PARTNER は、 701 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 111. BPC.BFM.PROCESS.PARTNER の XML エレメント

XML エレメント	説明
partnerLinkName	パートナー・リンクの名前。

注: BPC.BFM.PROCESS.PARTNER イベントのエンドポイント参照は、バージョン 6.1 イベントの CBE のアプリケーション・データ・セクションにのみ書き込まれます。このペイロードは、Web Services Addressing (WS-Addressing) EndpointReferenceType エレメントを含む Web Services Business Process Execution Language (WS-BPEL) ServiceRefType ラッパー・エレメントです。ServiceRefType アーチファクト (スキーマ) は、標準的なシナリオの場合であるアプリケーションのコンテキストでは、使用可能であるはずです。しかし、静的に定義されたパートナー・リンク間でエンドポイントを動的に割り当てる場合、このスキーマは使用できず、エンドポイント参照は組み込まれません。

BPC.BFM.PROCESS.START

BPC.BFM.PROCESS.START は、 701 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 112. BPC.BFM.PROCESS.START の XML エレメント

XML エレメント	説明
username	プロセスの開始または再開を要求したユーザーの名前。

BPC.BFM.PROCESS.STATUS

BPC.BFM.PROCESS.STATUS は、 701 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

BPC.BFM.PROCESS.WISTATUS

BPC.BFM.PROCESS.WISTATUS は、 701 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 113. BPC.BFM.PROCESS.WISTATUS の XML エレメント

XML エレメント	説明
username	作業項目が作成または削除されたユーザーの名前。
reason	作業項目の割り当て理由。指定可能な整数値の意味は、以下のとおりです。 1 - REASON_POTENTIAL_OWNER 2 - REASON_EDITOR 3 - REASON_READER 4 - REASON_OWNER 5 - REASON_POTENTIAL_STARTER 6 - REASON_STARTER 7 - REASON_ADMINISTRATOR 9 - REASON_ORIGINATOR 10 - REASON_ESCALATION_RECEIVER 11 - REASON_POTENTIAL_INSTANCE_ CREATOR

BPC.BFM.PROCESS.WITRANSFER

BPC.BFM.PROCESS.WITRANSFER は、 701 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 114. BPC.BFM.PROCESS.WITRANSFER の XML エレメント

XML エレメント	説明
current	作業項目の現在の所有者のユーザー名。これは、他のユーザーに転送された作業項目があるユーザーです。
target	作業項目の新規所有者のユーザー名。

表 114. *BPC.BFM.PROCESS.WITRANSFER の XML エレメント (続き)*

XML エレメント	説明
reason	作業項目の割り当て理由。指定可能な整数値の意味は、以下のとおりです。 1 - REASON_POTENTIAL_OWNER 2 - REASON_EDITOR 3 - REASON_READER 4 - REASON_OWNER 5 - REASON_POTENTIAL_STARTER 6 - REASON_STARTER 7 - REASON_ADMINISTRATOR 9 - REASON_ORIGINATOR 10 - REASON_ESCALATION_RECEIVER 11 - REASON_POTENTIAL_INSTANCE_CREATOR

BPC.BFM.VARIABLE.STATUS

BPC.BFM.VARIABLE.STATUS は、 700 ページの『BPC.BFM.BASE』の XML エレメントを継承します。

表 115. *BPC.BFM.VARIABLE.STATUS の XML エレメント*

XML エレメント	説明
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* エレメントも *VariableData_BO* エレメントも存在しません。変数のデータは、ストリングまたはビジネス・オブジェクト (BO) のいずれかとして表現されます。形式は、WebSphere Integration Developer の「**イベント・モニター**」タブで「**互換性があるイベントのモニター**」オプションが選択されたかどうかによって異なります。 この属性は、WebSphere Business Monitor 6.0.2 形式のイベントでのみ使用されます。WebSphere Business Monitor 6.1 形式のイベントの場合、変数の内容が *applicationData* セクションに書き込まれます。このセクションには、名前が変数名に設定されている 1 つのコンテンツ・エレメントが含まれています。
bpelId	変数の Business Process Choreographer ID。
principal	どのユーザーのために現在のアクションが実行されているかを示すユーザー名。
processTemplateId	プロセス・テンプレートの ID。

関連資料

『ビジネス・プロセス・イベント』
ビジネス・プロセスの Common Base Event は、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に発行されます。プロセスによって、プロセス・イベント、アクティビティー・イベント、アクティビティー・スコープ・イベント、リンク・イベント、および変数イベントが発行される場合があります。

710 ページの『ビジネス・プロセスの Common Base Events』
ビジネス・プロセスの Common Base Event は、WebSphere Integration Developer 内のビジネス・プロセスのモニターが要求された場合に発行されます。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。 これらのイベントは監査ログにも書き込まれます。

716 ページの『アクティビティーの Common Base Event』
アクティビティーの Common Base Event は、WebSphere Integration Developer 内のアクティビティーのモニターが要求された場合に発行されます。ここでは、アクティビティー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。 これらのイベントは監査ログにも書き込まれます。

726 ページの『scope アクティビティーの Common Base Event』
scope アクティビティーの Common Base Event は、WebSphere Integration Developer 内のこれらのアクティビティーのモニターが要求された場合に発行されます。ここでは、アクティビティー・スコープ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。 これらのイベントは監査ログにも書き込まれます。

730 ページの『flow アクティビティーのリンクの Common Base Event』
リンクの Common Base Event は、 WebSphere Integration Developer 内のリンクが定義されている flow アクティビティーのモニターが要求された場合に発行されます。ここでは、リン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。これらのイベントは監査ログにも書き込まれます。

731 ページの『プロセス変数の Common Base Events』
プロセス変数の Common Base Event は、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に発行されます。ここでは、変数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。 これらのイベントは監査ログにも書き込まれます。

ビジネス・プロセス・イベント

ビジネス・プロセスの Common Base Event は、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に発行されます。プロセスによって、プロセス・イベント、アクティビティー・イベント、アクティビティー・スコープ・イベント、リンク・イベント、および変数イベントが発行される場合があります。

プロセス・テンプレート・イベントを除くすべてのビジネス・プロセス・イベントは、CEI と監査証跡の両方で発行できます。プロセス・テンプレート・イベント PROCESS_INSTALLED および PROCESS_UNINSTALLED は、監査証跡でのみ発行できます。

注: ヒューマン・タスク・アクティビティーには、関連するインライン・ヒューマン・タスクがあります。ビジネス・プロセスを定義するときに、アクティビティーと、それに関連するインライン・ヒューマン・タスクの両方がイベントを発行することを指定できます。

イベントの構造は、XML スキーマ定義 (XSD) ファイル BFMEvents.xsd に記述されています。このファイルは、*install_root*¥ProcessChoreographer¥client ディレクトリーにあります。

関連資料

689 ページの『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。

732 ページの『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。

690 ページの『ビジネス・プロセス・イベントの拡張子名』
拡張子名は、イベントの有効搭載量を示します。ビジネス・プロセス・イベントのすべての拡張子名と、対応する有効搭載量のリストがここに表示されます。

ビジネス・プロセスの Common Base Events

ビジネス・プロセスの Common Base Event は、WebSphere Integration Developer 内のビジネス・プロセスのモニターが要求された場合に発行されます。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。 これらのイベントは監査ログにも書き込まれます。

状態遷移およびプロセス・イベント

以下の図は、ビジネス・プロセスで発生する可能性のある状態遷移、およびこれらの状態変更が発生する場合に発行されるイベントを示しています。各状態間のリンクは、イベントの性質と、状態遷移で発行されるイベントのイベント・コードを示しています。

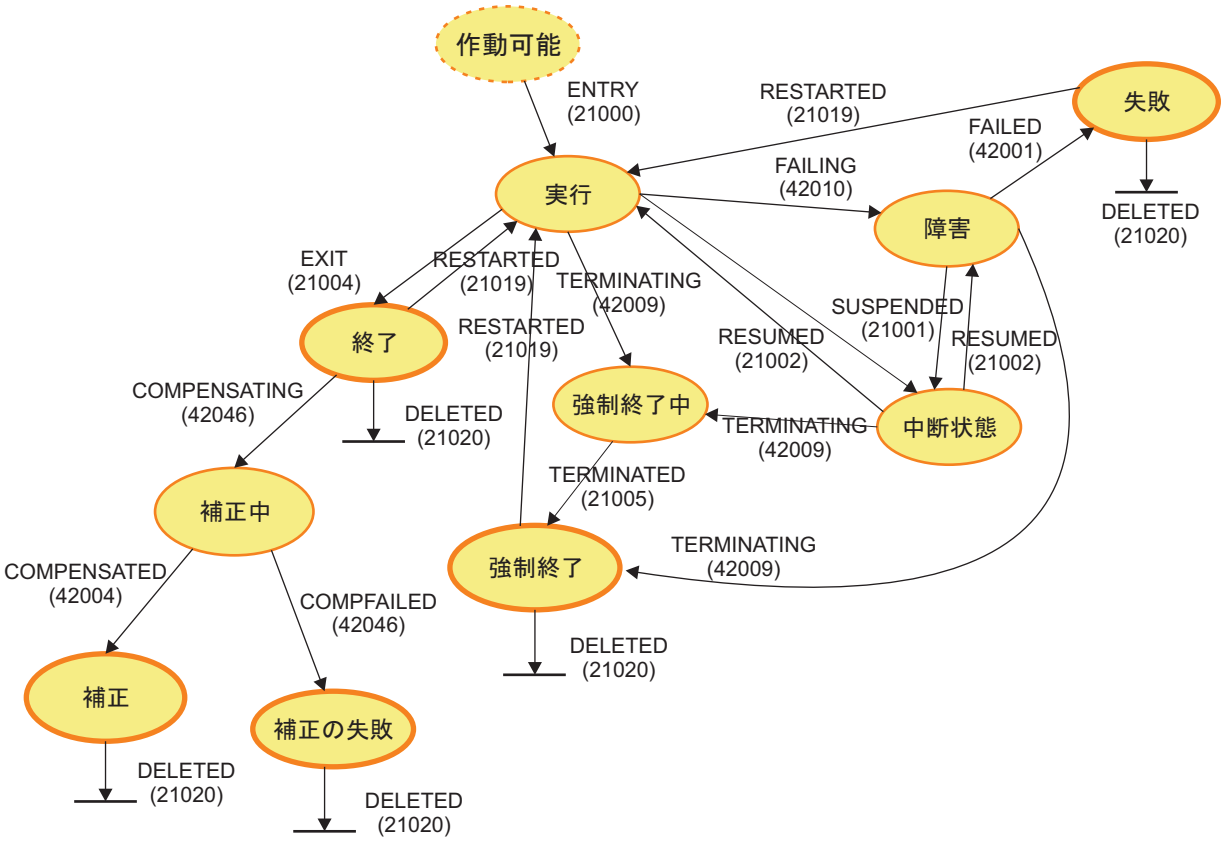


図 10. 状態遷移およびプロセス・イベント

プロセス・イベント

以下のテーブルの列の内容は、次のとおりです。

コード イベントの番号。WebSphere Business Monitor 6.0.2 形式のイベントの場合、この値は *BPCEventCode* 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。WebSphere Business Monitor 6.1 形式のイベントの場合、この値は Common Base Event の xs:any スロットに書き込まれます。

イベント名および拡張子名
この列には 2 つの値があります。これは、イベントの名前と、Common Base Event の *extensionName* 属性に設定された値です。拡張子名は、Common Base Event に含まれるイベント固有情報を識別するものであり、イベントに関する追加データを提供する XML エレメントの名前でもあります。

状態 ビジネス・プロセス・イベントの状態名を指します。

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。

一部のプロセス・イベントは、状態変更を伴わずに発行されます。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。

表 116. プロセス・イベント

コード	イベント名および拡張子名	状態	イベント性質	説明
21000	PROCESS_STARTED BPC.BFM.PROCESS.START	開始	ENTRY	プロセスが開始された。
21001	PROCESS_SUSPENDED BPC.BFM.PROCESS.STATUS	レポート	SUSPENDED	プロセスが中断された。プロセス・インスタンスを中断するには、Business Process Choreographer Explorer を使用します。
21002	PROCESS_RESUMED BPC.BFM.PROCESS.STATUS	レポート	RESUMED	プロセスが再開された。中断されたプロセスのみを再開できます。プロセス・インスタンスを再開するには、Business Process Choreographer Explorer を使用します。
21004	PROCESS_COMPLETED BPC.BFM.PROCESS.STATUS	停止	EXIT	プロセスが完了した。
21005	PROCESS_TERMINATED BPC.BFM.PROCESS.STATUS	停止	TERMINATED	プロセスが強制終了した。プロセス・インスタンスを強制終了するには、Business Process Choreographer Explorer を使用します。
21019	PROCESS_RESTARTED BPC.BFM.PROCESS.START	レポート	RESTARTED	プロセスが再始動された。例えば、Business Process Choreographer Explorer を使用するなどして、要求に対してプロセスが再始動されます。
21020	PROCESS_DELETED BPC.BFM.PROCESS.STATUS	破棄	DELETED	プロセスが削除された。
42001	PROCESS_FAILED BPC.BFM.PROCESS. FAILURE	失敗	FAILED	プロセスが失敗した。
42003	PROCESS_COMPENSATING BPC.BFM.PROCESS.STATUS	レポート	COMPENSATING	プロセスが補正中。子プロセスのみを補正できます。子プロセスの補正は、親プロセスに関連付けられている障害ハンドラーまたは補正ハンドラーによってトリガーされます。
42004	PROCESS_COMPENSATED BPC.BFM.PROCESS.STATUS	停止	COMPENSATED	プロセスが補正された。

表 116. プロセス・イベント (続き)

コード	イベント名および拡張子名	状態	イベント性質	説明
42006	PROCESS_INSTALLED	レポート	INSTALLED	これらはプロセス・インスタンス・イベントであり、監査証跡でのみ発行されます。 Common Base Event としては発行されませんが、完全を期すためにこの表に入っています。
42007	PROCESS_UNINSTALLED	レポート	UNINSTALLED	
42009	PROCESS_TERMINATING BPC.BFM.PROCESS.STATUS	レポート	TERMINATING	プロセスが強制終了中。
42010	PROCESS_FAILING BPC.BFM.PROCESS.STATUS	レポート	FAILING	プロセスが失敗する。
42027	PROCESS_CORRELATION_ SET_INITIALIZED BPC.BFM.PROCESS.CORREL	レポート	CORRELATION	このイベントは、プロセス・インスタンスの新しい相関セットが初期化された場合、(例えば、開始相関セットを持つ receive アクティビティーがメッセージを受信した場合など) に発行されます。 このイベントは、状態変更に関連付けられていません。
42041	PROCESS_WORKITEM_ DELETED BPC.BFM.PROCESS. WISTATUS	レポート	WI_DELETED	プロセス作業項目が削除された。このイベントは、API 要求によって作業項目が明示的に削除される場合にのみ発行されます。対応するプロセス・インスタンスが削除されたために作業項目が削除される場合、イベントは発行されません。 このイベントは、状態変更に関連付けられていません。
42042	PROCESS_WORKITEM_ CREATED BPC.BFM.PROCESS. WISTATUS	レポート	WI_CREATED	プロセス作業項目が作成された。このイベントは、プロセス用の追加の作業項目が API 要求などによって作成された場合に発行されます。 このイベントは、状態変更に関連付けられていません。

表 116. プロセス・イベント (続き)

コード	イベント名および拡張子名	状態	イベント性質	説明
42046	PROCESS_COMPENSATION_FAILED BPC.BFM.PROCESS.STATUS	失敗	COMPFAILED	プロセス補正が失敗した。
42047	PROCESS_EVENT_RECEIVED BPC.BFM.PROCESS.EVENT	レポート	EV_RECEIVED	プロセス・イベントを受信した。このイベントは、プロセスに関連付けられているイベント・ハンドラーがアクティブになると発行されます。 このイベントは、状態変更に関連付けられていません。
42049	PROCESS_EVENT_ESCALATED BPC.BFM.PROCESS.ESCALATED	レポート	EV_ESCALATED	プロセス・イベントがエスカレートされた。このイベントは、プロセスの onEvent イベント・ハンドラーに関連付けられているインライン呼び出しタスクがエスカレートされると発行されます。 このイベントは、状態変更に関連付けられていません。
42056	PROCESS_WORKITEM_ TRANSFERRED BPC.BFM.PROCESS. WITRANSFER	レポート	WI_ TRANSFERRED	プロセス作業項目が転送された。 このイベントは、状態変更に関連付けられていません。
42058	PROCESS_PARTNER_CHANGED BPC.BFM.PROCESS.PARTNER	レポート	PA_CHANGE	プロセス・パートナーが変更された。このイベントは、新規エンドポイント参照がパートナー・リンクに割り当てられると発行されます。 このイベントは、状態変更に関連付けられていません。

表 116. プロセス・イベント (続き)

コード	イベント名および拡張子名	状態	イベント性質	説明
42059	PROCESS_CUSTOMPROPERTY_SET BPC.BFM.PROCESS. CUSTOMPROPERTYSET	レポート	CP_SET	プロセスのカスタム・プロパティーが設定された。このイベントは、プロセス・インスタンスのカスタム・プロパティーが変更されると発行されます。 このイベントは、状態変更に関連付けられていません。
42071	PROCESS_OWNER_TRANSFERRED BPC.BFM.PROCESS.OWNERTRANSFER	レポート	OWNER_ TRANSFERRED	このイベントは、プロセスの所有権があるユーザーから別のユーザーに転送されると発行されます。 このイベントは、状態変更に関連付けられていません。
42077	PROCESS_CORRELATION_SET_SET BPC.BFM.PROCESS.CORREL	レポート	CORRELATION	このイベントは、プロセス・インスタンスの相関セットの値が設定された場合に発行されます。 このイベントは、状態変更に関連付けられていません。
42078	PROCESS_CORRELATION_SET_UNSET BPC.BFM.PROCESS.CORREL	レポート	CORRELATION	このイベントは、プロセス・インスタンスの相関セットの値が削除または設定解除された場合に発行されます。 このイベントは、状態変更に関連付けられていません。
42079	PROCESS_MIGRATED BPC.BFM.PROCESS.MIGRATED	レポート	MIGRATED	このイベントは、新規テンプレートを使用するためにプロセスがマイグレーションされた場合に発行されます。 このイベントは、状態変更に関連付けられていません。

表 116. プロセス・イベント (続き)

コード	イベント名および拡張子名	状態	イベント性質	説明
42080	PROCESS_MIGRATION_TRIGGERED BPC.BFM.PROCESS. MIGRATIONTRIGGERED	レポート	MIGRATION_ TRIGGERED	このイベントは、新規テンプレートを使用するためのプロセス・インスタンスのマイグレーションが開始した場合に発行されます。 このイベントは、状態変更に関連付けられていません。

プロセス・イベントの場合、以下のイベント相関範囲 ID の内容は次のとおりです。

- 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。

関連資料

689 ページの『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。

732 ページの『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。

690 ページの『ビジネス・プロセス・イベントの拡張子名』
拡張子名は、イベントの有効搭載量を示します。ビジネス・プロセス・イベントのすべての拡張子名と、対応する有効搭載量のリストがここに表示されます。

アクティビティーの Common Base Event

アクティビティーの Common Base Event は、WebSphere Integration Developer 内のアクティビティーのモニターが要求された場合に発行されます。ここでは、アクティビティー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。 これらのイベントは監査ログにも書き込まれます。

状態遷移およびアクティビティー・イベント

状態変更と発行されるイベントは、以下のアクティビティーのタイプによって異なります。

- invoke、assign、empty、reply、rethrow、throw、terminate、および Java snippet アクティビティー

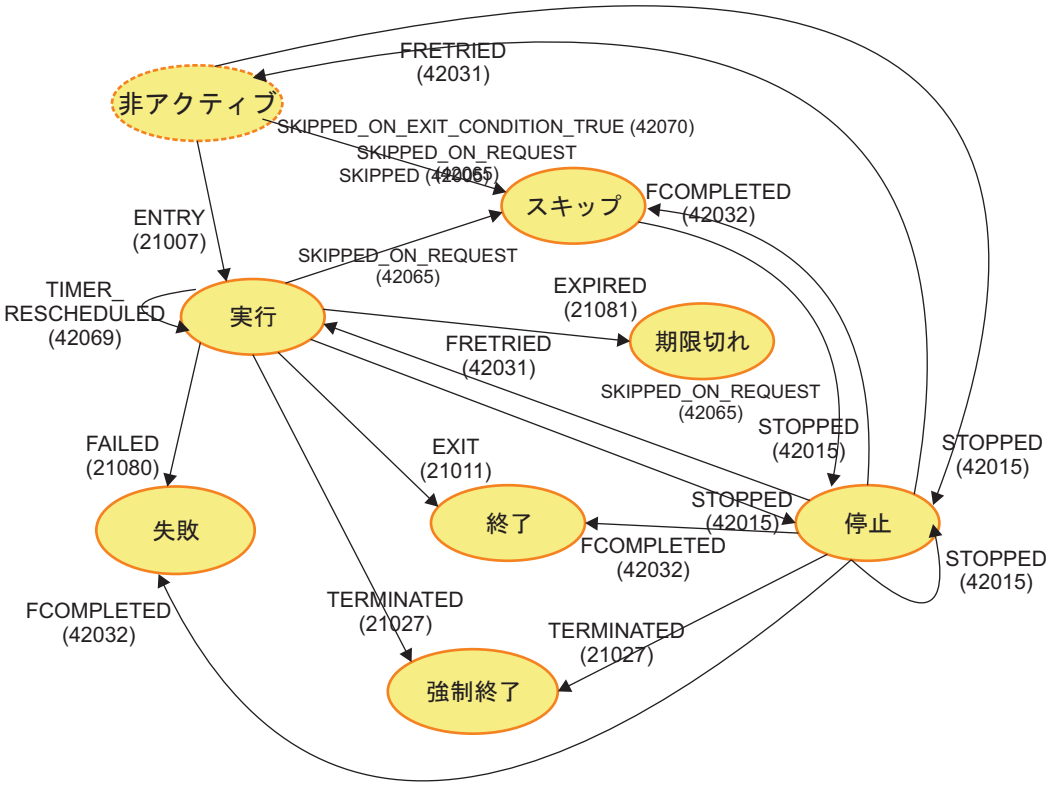


図 11. invoke アクティビティーと short-lived アクティビティーの状態遷移およびイベント

- pick (receive choice)、wait、および receive アクティビティー

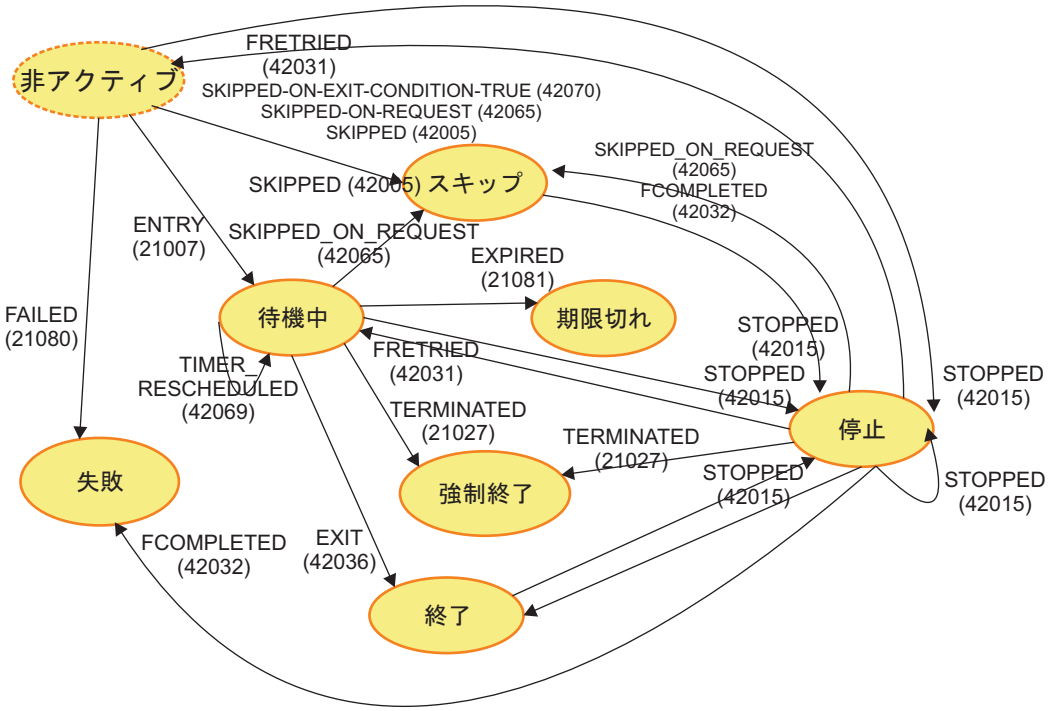


図 12. wait、および receive アクティビティーの状態遷移およびイベント

- ヒューマン・タスク・アクティビティー

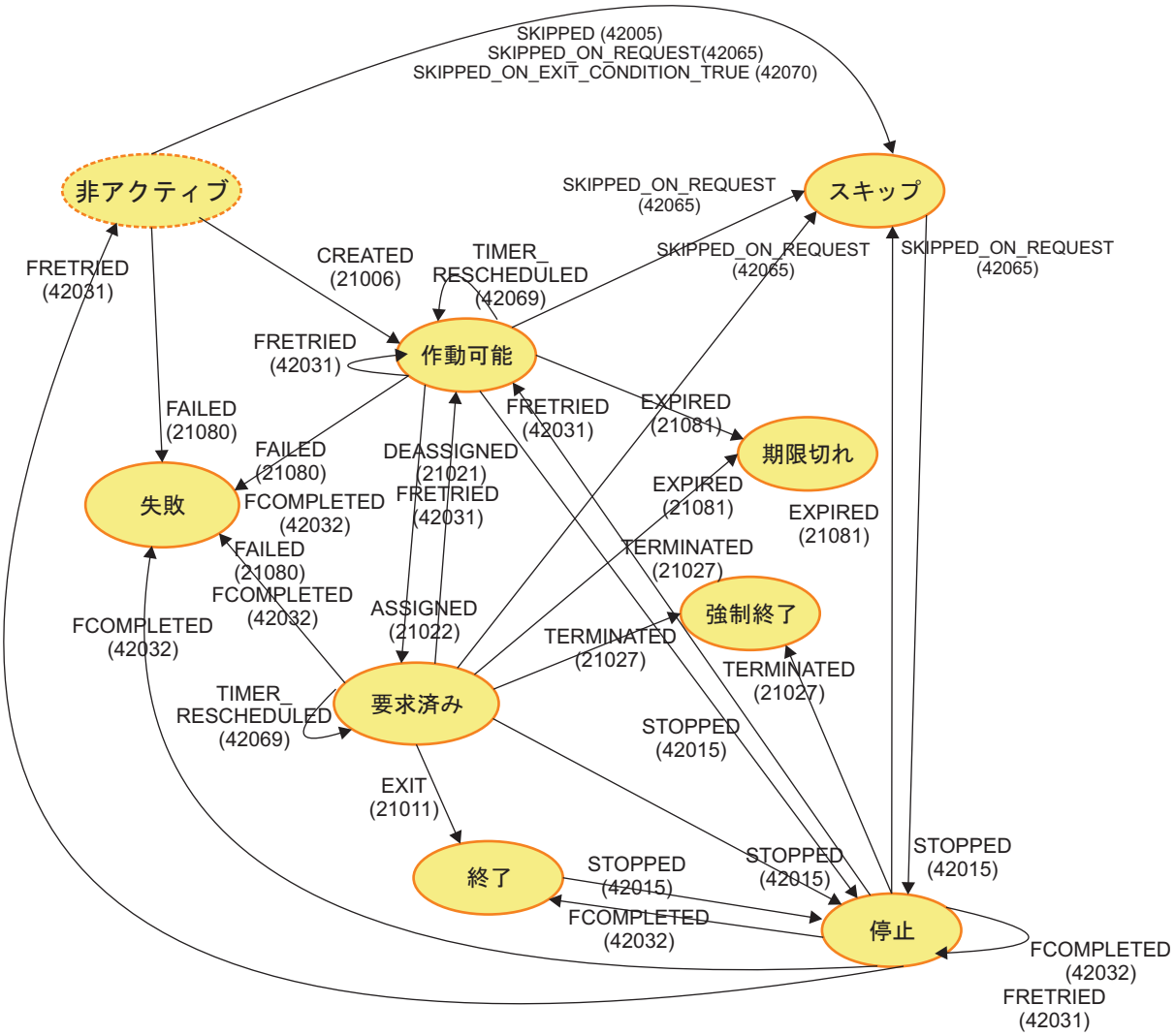


図 13. ヒューマン・タスク・アクティビティーの状態遷移およびイベント

- 構造化アクティビティー (flow または sequence アクティビティーなど)

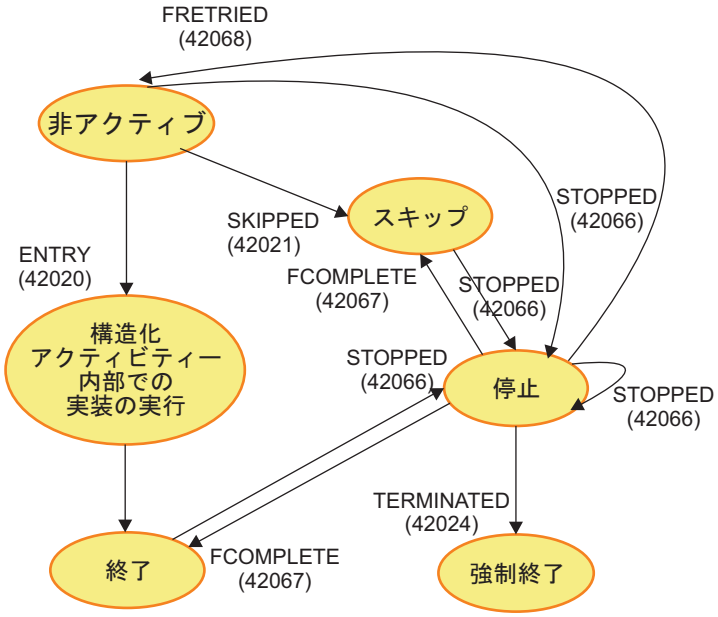


図 14. 構造化アクティビティーの状態遷移およびイベント

アクティビティー・イベント

以下のテーブルの列の内容は、次のとおりです。

コード イベントの番号。WebSphere Business Monitor 6.0.2 形式のイベントの場合、この値は *BPCEventCode* 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。WebSphere Business Monitor 6.1 形式のイベントの場合、この値は Common Base Event の xs:any スロットに書き込まれます。

イベント名および拡張子名
この列には 2 つの値があります。これは、イベントの名前と、Common Base Event の *extensionName* 属性に設定された値です。拡張子名は、Common Base Event に含まれるイベント固有情報を識別するものであり、イベントに関する追加データを提供する XML エレメントの名前でもあります。

状態 ビジネス・プロセス・イベントの状態名を指します。

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。

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。

表 117. アクティビティー・イベント

コード	イベント名および拡張子名	状態	イベント性質	説明
21006	ACTIVITY_READY BPC.BFM.ACTIVITY. MESSAGE	開始	CREATED	アクティビティーが実行可能である。このイベントは、ヒューマン・タスク・アクティビティーの開始時に発行されます。
21007	ACTIVITY_STARTED 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	アクティビティーが開始した。invoke アクティビティーでは、ビジネス・オブジェクト・ペイロードが使用可能です。
21011	ACTIVITY_COMPLETED invoke、human task、receive、および reply アクティビティーの場合: BPC.BFM.ACTIVITY. MESSAGE pick アクティビティーの場合: BPC.BFM.ACTIVITY.EVENT 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	アクティビティーが完了した。invoke、ヒューマン・タスク、receive、および reply アクティビティーでは、ビジネス・オブジェクト・ペイロードが使用可能です。
21021	ACTIVITY_CLAIM_ CANCELED BPC.BFM.ACTIVITY. STATUS	レポート	DEASSIGNED	要求がキャンセルされた。このイベントは、ヒューマン・タスク・アクティビティーに対する要求が取り消された時点で発行されます。
21022	ACTIVITY_CLAIMED BPC.BFM.ACTIVITY. CLAIM	レポート	ASSIGNED	アクティビティーが要求された。このイベントは、ヒューマン・タスク・アクティビティーが要求された時点で発行されます。
21027	ACTIVITY_ TERMINATED BPC.BFM.ACTIVITY. STATUS	停止	TERMINATED	アクティビティーが強制終了した。長期実行アクティビティーが、割り当てられている有効範囲またはプロセスでの障害処理の影響で強制終了することがあります。

表 117. アクティビティー・イベント (続き)

コード	イベント名および拡張子名	状態	イベント性質	説明
21080	ACTIVITY_FAILED BPC.BFM.ACTIVITY. FAILURE	失敗	FAILED	アクティビティーが失敗した。このイベントは、アクティビティーの実行時に障害が発生し、その障害がエンクロージング・スコープまたはプロセスに定義されている障害ハンドラーに伝搬する場合に発行されます。
21081	ACTIVITY_EXPIRED BPC.BFM.ACTIVITY. STATUS	レポート	EXPIRED	アクティビティーが期限切れになった。このイベントは、invoke アクティビティーおよびヒューマン・タスク・アクティビティーにのみ適用します。
42005	ACTIVITY_SKIPPED BPC.BFM.ACTIVITY. STATUS	レポート	SKIPPED	アクティビティーがスキップされた。このイベントは、結合動作が定義されているアクティビティーにのみ適用します。結合動作が false と評価された場合、アクティビティーはスキップされ、スキップされたイベントが発行されます。
42012	ACTIVITY_OUTPUT_ MESSAGE_SET BPC.BFM.ACTIVITY. MESSAGE	レポート	OUTPUTSET	アクティビティー出力メッセージが設定された。ビジネス・オブジェクト・ペイロードが使用可能です。 このイベントは、アクティビティーが完了していない状態で、要求されたヒューマン・タスク・アクティビティーの出力メッセージが設定される場合に (例えば、中間結果を保管するために) 発行されます。アクティビティーの状態は変更されません。 このイベントは、ヒューマン・タスク・アクティビティーの完了時には発行されません。

表 117. アクティビティー・イベント (続き)

コード	イベント名および拡張子名	状態	イベント性質	説明
42013	ACTIVITY_FAULT_ MESSAGE_SET BPC.BFM.ACTIVITY. MESSAGE	レポート	FAULTSET	アクティビティー障害メッセージが設定された。ビジネス・オブジェクト・ペイロードが使用可能です。 このイベントは、アクティビティーが完了していない状態で、要求されたヒューマン・タスク・アクティビティーの障害メッセージが設定される場合に発行されます。このイベントは、ヒューマン・タスク・アクティビティーが完了して障害が発生する場合は発行されません。
42015	ACTIVITY_STOPPED BPC.BFM.ACTIVITY. STATUS	停止	STOPPED	アクティビティーが停止した。アクティビティーの実行時に未処理の障害が発生する場合は、アクティビティーを停止できます。
42031	ACTIVITY_FORCE_ RETRIED BPC.BFM.ACTIVITY. STATUS	レポート	FRETRIED	アクティビティーが強制的に再試行された。アクティビティーを強制的に再試行するには、Business Process Choreographer Explorer を使用します。
42032	ACTIVITY_FORCE_ COMPLETED BPC.BFM.ACTIVITY. STATUS	停止	FCOMPLETED	アクティビティーが強制終了された。アクティビティーを強制終了するには、Business Process Choreographer Explorer を使用します。
42036	ACTIVITY_MESSAGE_ RECEIVED BPC.BFM.ACTIVITY. MESSAGE	レポート	EXIT	pick (receive choice) アクティビティーでメッセージが受信された
42037	ACTIVITY_LOOP_ CONDITION_TRUE BPC.BFM.ACTIVITY. STATUS	レポート	CONDTRUE	ループ条件が true。
42038	ACTIVITY_LOOP_ CONDITION_FALSE BPC.BFM.ACTIVITY. STATUS	レポート	CONDFALSE	ループ条件が false。

表 117. アクティビティー・イベント (続き)

コード	イベント名および拡張子名	状態	イベント性質	説明
42039	ACTIVITY_WORKITEM_ DELETED BPC.BFM.ACTIVITY. WISTATUS	レポート	WI_DELETED	作業項目が削除された。このイベントは、pick、ヒューマン・タスク、および受信イベントにのみ適用します。 このイベントは、API 要求によって作業項目が明示的に削除される場合にのみ発行されます。対応するプロセス・インスタンスが削除されたために作業項目が削除される場合、イベントは発行されません。
42040	ACTIVITY_WORKITEM_ CREATED BPC.BFM.ACTIVITY. WISTATUS	レポート	WI_CREATED	作業項目が作成された。このイベントは、pick、ヒューマン・タスク、および受信イベントにのみ適用します。
42050	ACTIVITY_ESCALATED BPC.BFM.ACTIVITY. ESCALATED	レポート	ESCALATED	アクティビティーがエスカレートされた。このイベントは、ヒューマン・タスク・アクティビティーに関連したエスカレーションが発生する場合に、 pick、ヒューマン・タスク、および受信イベントにのみ適用します。
42054	ACTIVITY_WORKITEM_ REFRESHED BPC.BFM.ACTIVITY. WISTATUS	レポート	WI_REFRESHED	アクティビティー作業項目が更新された。このイベントは、pick、ヒューマン・タスク、および受信イベントにのみ適用します。
42055	ACTIVITY_WORKITEM_ TRANSFERRED BPC.BFM.ACTIVITY. WITRANSFER	レポート	WI_TRANSFERRED	作業項目が転送された。このイベントは、pick、ヒューマン・タスク、および受信イベントにのみ適用します。
42057	ACTIVITY_PARALLEL_ BRANCHES_STARTED BPC.BFM.ACTIVITY. FOREACH	レポート	BRANCHES_ STARTED	このイベントは、forEach アクティビティーのブランチの開始時に発行されます。
42060	ACTIVITY_ CUSTOMPROPERTY_SET BPC.BFM.ACTIVITY. CUSTOMPROPERTYSET	レポート	CP_SET	このイベントは、アクティビティー・インスタンスのカスタム・プロパティーが変更された時点で発行されます。

表 117. アクティビティー・イベント (続き)

コード	イベント名および拡張子名	状態	イベント性質	説明
42061	ACTIVITY_BRANCH_ CONDITION_TRUE BPC.BFM.ACTIVITY. CONDITION	レポート	CONDTRUE	このイベントは、choice アクティビティーの case 条件が true に評価された場合に発行されます。ナビゲートされた各 choice アクティビティー・インスタンスで、case エレメント条件が true に設定されるイベントは多くても 1 つです。つまり、case エレメントに入らない場合、case エレメントはイベントの対象外となり、otherwise エレメントが条件 case エレメントと同様のイベントを発生させます。
42062	ACTIVITY_ALL_BRANCH_ CONDITIONS_FALSE BPC.BFM.ACTIVITY. STATUS	レポート	ALLCONDFALSE	このイベントは、case エレメントが使用されず、otherwise エレメントが存在しない場合に発行されます。この場合、ナビゲーションは choice 構文の最後まで続行します。
42063	ACTIVITY_JUMPED BPC.BFM.ACTIVITY. JUMPED	レポート	JUMPED	このイベントは、ジャンプ・アクションのソース・アクティビティーの最終アクティビティー・イベント後、ターゲット・アクティビティーの最初のイベント前に発行されます。
42064	ACTIVITY_SKIP_ REQUESTED BPC.BFM.ACTIVITY. SKIP_REQUESTED	レポート	SKIP_REQUESTED	スキップ・アクティビティーが要求された。このイベントは、対応するアクティビティーがアクティブ状態ではなく、skip または cancelSkipRequest API が呼び出された場合に発行されます。この場合、要求はすぐにはナビゲーションに影響しません。イベントには、skip と cancelSkipRequest 呼び出しを区別するためのフラグが含まれます。 スキップするイベントの ECSCurrentID は、関連したアクティビティーの AIID に設定されません。

表 117. アクティビティー・イベント (続き)

コード	イベント名および拡張子名	状態	イベント性質	説明
42065	ACTIVITY_SKIPPED_ ON_REQUEST BPC.BFM.ACTIVITY. SKIPPED_ON_REQUEST	レポート	SKIPPED_ON_ REQUEST	要求によりイベントがスキップされた。このイベントは、スキップ対象のマークが付いているアクティビティーの後のナビゲーションが続行された場合に発行されます。
42069	ACTIVITY_TIMER_ RESCHEDULED BPC.BFM.ACTIVITY. TIMER_RESCHEDULED	レポート	TIMER_ RESCHEDULED	このイベントは、rescheduleTimer 要求が処理された場合に発行されます。wait、human task、invoke、および pick の各アクティビティーに対してこのイベントを生成できます。
42070	ACTIVITY_SKIPPED_ ON_EXIT_ CONDITION BPC.BFM.ACTIVITY.SKIP_ ON_EXIT_CONDITION_TRUE	レポート	SKIPPED_ON_ EXIT_ CONDITION_ TRUE	このイベントは、onEntry タイプの出口条件が true に評価されたためにアクティビティーがスキップされる場合に発行されます。
42072	ACTIVITY_CHILD_PROCESS_ TERMINATING BPC.BFM.ACTIVITY.CHILD_ PROCESS_TERMINATING	レポート	CHILD_ PROCESS_ TERMINATING	このイベントは、対応するアクティビティーが実行状態の invoke アクティビティーであり、子プロセスを保有し、なおかつ forceRetry、forceComplete、または skip API が呼び出されるかアクティビティーが期限切れになった場合に発行されます。
42073	ACTIVITY_CONDITION_ FORCED BPC.BFM.ACTIVITY. CONDITIONFORCED	レポート	ACTIVITY_ CONDITION_ FORCED	結合条件の評価エラーが原因で停止したアクティビティーは強制的にナビゲーションが続行されました。
42074	ACTIVITY_LOOP_ CONDITION_FORCED BPC.BFM.ACTIVITY. CONDITIONFORCED	レポート	ACTIVITY_LOOP_ CONDITION_ FORCED	repeat-until または while ループ条件の評価エラーが原因で停止したアクティビティーは強制的にナビゲーションが続行されました。
42075	ACTIVITY_FOR_EACH_ COUNTERS_FORCED BPC.BFM.ACTIVITY. COUNTERSFORCED	レポート	ACTIVITY_ FOR_EACH_ COUNTERS_ FORCED	for-each ループ条件の評価エラーが原因で停止したアクティビティーは強制的にナビゲーションが続行されました。

ほとんどのアクティビティー・イベントの場合、イベント相関範囲 ID の内容は次のとおりです。

- *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。

カスタム・プロパティーが設定されたイベントの場合、イベント相関範囲 ID は、カスタム・プロパティーが設定されているコンテキストを示します。例えば、API 要求を使用してカスタム・プロパティーが設定された場合、イベント相関範囲 ID はプロセス・イベントの場合と同様に設定されます。カスタム・プロパティーが Java 断片に設定された場合、*ECSCurrentID* にはその Java 断片のアクティビティー・インスタンス ID が設定され、*ECSParentID* にはプロセス・インスタンス ID が設定されます。

関連資料

689 ページの『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。

732 ページの『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。

690 ページの『ビジネス・プロセス・イベントの拡張子名』
拡張子名は、イベントの有効搭載量を示します。ビジネス・プロセス・イベントのすべての拡張子名と、対応する有効搭載量のリストがここに表示されます。

scope アクティビティーの Common Base Event

scope アクティビティーの Common Base Event は、WebSphere Integration Developer 内のこれらのアクティビティーのモニターが要求された場合に発行されます。ここでは、アクティビティー・スコープ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。 これらのイベントは監査ログにも書き込まれます。

scope アクティビティーの状態遷移およびイベント

以下の図は、scope アクティビティーで発生する可能性のある状態遷移、およびこれらの状態変更が発生する場合に発行されるイベントを示しています。

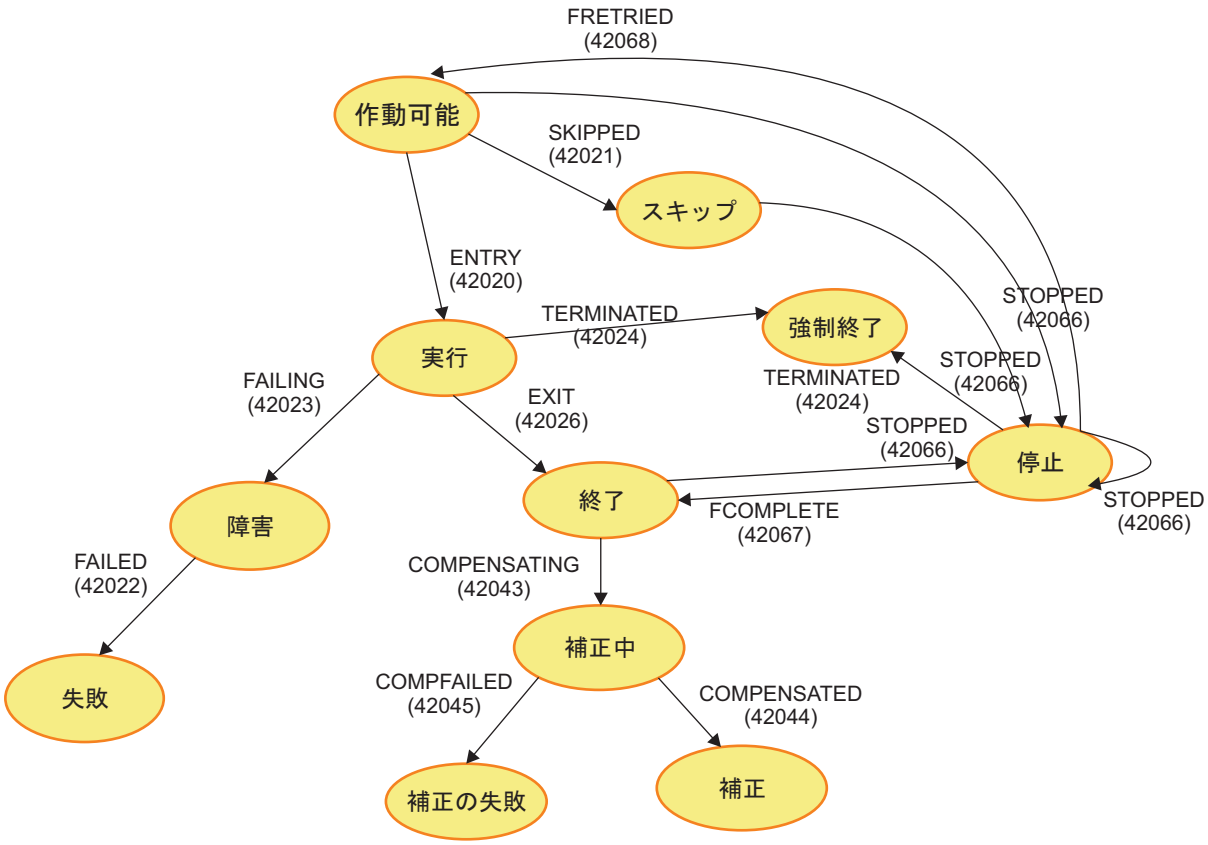


図 15. scope アクティビティーの状態遷移およびイベント

scope アクティビティーのイベント

以下のテーブルの列の内容は、次のとおりです。

コード イベントの番号。WebSphere Business Monitor 6.0.2 形式のイベントの場合、この値は *BPCEventCode* 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。WebSphere Business Monitor 6.1 形式のイベントの場合、この値は Common Base Event の xs:any スロットに書き込まれます。

イベント名および拡張子名
この列には 2 つの値があります。これは、イベントの名前と、Common Base Event の *extensionName* 属性に設定された値です。拡張子名は、Common Base Event に含まれるイベント固有情報を識別するものであり、イベントに関する追加データを提供する XML エレメントの名前でもあります。

状態 ビジネス・プロセス・イベントの状態名を指します。

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。

以下の表は、scope アクティビティーのすべてのイベントを説明しています。

表 118. scope アクティビティーのイベント

コード	イベント名および拡張子名	状態	イベント性質	説明
42020	SCOPE_STARTED BPC.BFM.ACTIVITY. STATUS	開始	ENTRY	スコープが開始した。このイベントは、ナビゲーションがスコープ・インスタンスに入った場合に発行されます。
42021	SCOPE_SKIPPED BPC.BFM.ACTIVITY. STATUS	レポート	SKIPPED	スコープがスキップされた。イベントは、結合動作が定義されている scope アクティビティーにのみ適用されます。このイベントは、スコープの結合条件が false に評価された場合に発行されます。プロセスのナビゲーションは、デッド・パスを除去しながら、スコープの終端まで継続します。
42022	SCOPE_FAILED BPC.BFM.ACTIVITY. FAILURE	失敗	FAILED	スコープで障害が発生した。このイベントは、プロセス・ナビゲーションが、スコープの障害ハンドラーを出た場合に発行されます。
42023	SCOPE_FAILING BPC.BFM.ACTIVITY. STATUS	レポート	FAILING	スコープで障害が発生している。このイベントは、プロセス・ナビゲーションが、スコープの障害処理パスに入った場合に発行されます。
42024	SCOPE_TERMINATED BPC.BFM.ACTIVITY. STATUS	停止	TERMINATED	スコープが強制終了した。scope アクティビティーは、関連付けられたプロセスが scope アクティビティーと並列になっているブランチ内の terminate アクティビティーなどによって終了されると、終了します。
42026	SCOPE_COMPLETED BPC.BFM.ACTIVITY. STATUS	停止	EXIT	スコープが完了した。このイベントは、スコープの通常のナビゲーション・パスと、活動化されたイベント・ハンドラーのすべてのパスが完了したときに発行されます。
42043	SCOPE_COMPENSATING BPC.BFM.ACTIVITY. STATUS	レポート	COMPENSATING	スコープが補正中。このイベントは、プロセス・ナビゲーションが、スコープの補正ハンドラー (デフォルトの補正ハンドラーを含む) に入った場合に発行されます。

表 118. scope アクティビティーのイベント (続き)

コード	イベント名および拡張子名	状態	イベント性質	説明
42044	SCOPE_COMPENSATED BPC.BFM.ACTIVITY. STATUS	停止	COMPENSATED	スコープが補正された。このイベントは、スコープの補正ハンドラー (デフォルトの補正ハンドラーを含む) が完了したときに発行されます。
42045	SCOPE_COMPENSATION_ FAILED BPC.BFM.ACTIVITY. STATUS	失敗	COMPFAILED	スコープの補正が失敗した。このイベントは、スコープの補正ハンドラーの実行時に障害が発生した場合に発行されます。
42048	SCOPE_EVENT_ RECEIVED BPC.BFM.ACTIVITY. EVENT	レポート	EV_RECEIVED	このイベントは、スコープの新しいイベント・ハンドラー・インスタンスが開始された場合に発行されます。
42051	SCOPE_EVENT_ ESCALATED BPC.BFM.ACTIVITY. ESCALATED	レポート	EV_ESCALATED	スコープ・イベントがエスカレートされた。このイベントは、スコープのアクティブ・イベント・ハンドラーのインライン・ヒューマン・タスクに関連付けられているエスカレーションが開始されたときに発行されます。
42066	SCOPE_STOPPED BPC.BFM.ACTIVITY.STATUS	停止	STOPPED	スコープが停止した。アクティベーション中またはスコープの後続のナビゲーション中に未処理の障害が発生すると、スコープ・インスタンスが停止する場合があります。
42067	SCOPE_FORCE_ COMPLETED BPC.BFM.ACTIVITY. STATUS	レポート	FCOMPLETED	スコープが強制的に完了した
42068	SCOPE_FORCE_RETRIED BPC.BFM.ACTIVITY. STATUS	レポート	FRETRIED	スコープが強制的に再試行された
42076	SCOPE_CONDITION_ FORCED BPC.BFM.ACTIVITY. CONDITIONFORCED	レポート	SCOPE_ CONDITION_ FORCED	結合条件の評価エラーが原因で停止したアクティビティーは強制的にナビゲーションが続行されました。

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。

アクティビティー有効範囲イベントの場合、以下のイベント相関範囲 ID の内容は次のとおりです。

- ECSCurrentID は、スコープの ID です。
- ECSParentID は、収容プロセスの ID です。

関連資料

689 ページの『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。

732 ページの『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。

690 ページの『ビジネス・プロセス・イベントの拡張子名』
拡張子名は、イベントの有効搭載量を示します。ビジネス・プロセス・イベントのすべての拡張子名と、対応する有効搭載量のリストがここに表示されます。

flow アクティビティーのリンクの Common Base Event

リンクの Common Base Event は、 WebSphere Integration Developer 内のリンクが定義されている flow アクティビティーのモニターが要求された場合に発行されます。ここでは、リン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。これらのイベントは監査ログにも書き込まれます。

flow アクティビティーのリン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。

リンク・イベント

以下のテーブルの列の内容は、次のとおりです。

コード イベントの番号。WebSphere Business Monitor 6.0.2 形式のイベントの場合、この値は *BPCEventCode* 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。WebSphere Business Monitor 6.1 形式のイベントの場合、この値は Common Base Event の xs:any スロットに書き込まれます。

イベント名および拡張子名
この列には 2 つの値があります。これは、イベントの名前と、Common Base Event の *extensionName* 属性に設定された値です。拡張子名は、Common Base Event に含まれるイベント固有情報を識別するものであり、イベントに関する追加データを提供する XML エレメントの名前でもあります。

状態 ビジネス・プロセス・イベントの状態名を指します。

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。

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。

表 119. リンク・イベント

コード	イベント名および拡張子名	状態	イベント性質	説明
21034	LINK_EVALUATED_ TO_TRUE BPC.BFM.LINK.STATUS	レポート	CONDTRUE	リンクが true と評価された
42000	LINK_EVALUATED_ TO_FALSE BPC.BFM.LINK.STATUS	レポート	CONDFALSE	リンクが false と評価された

リンク・イベントの場合、以下のイベント相関範囲 ID の内容は次のとおりです。

- 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。

関連資料

689 ページの『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。

732 ページの『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。

690 ページの『ビジネス・プロセス・イベントの拡張子名』
拡張子名は、イベントの有効搭載量を示します。ビジネス・プロセス・イベントのすべての拡張子名と、対応する有効搭載量のリストがここに表示されます。

プロセス変数の Common Base Events

プロセス変数の Common Base Event は、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に発行されます。ここでは、変数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。 これらのイベントは監査ログにも書き込まれます。

プロセス変数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。

変数イベント

以下のテーブルの列の内容は、次のとおりです。

コード イベントの番号。WebSphere Business Monitor 6.0.2 形式のイベントの場合、この値は *BPCEventCode* 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。WebSphere Business Monitor 6.1 形式のイベントの場合、この値は Common Base Event の xs:any スロットに書き込まれます。

イベント名および拡張子名
この列には 2 つの値があります。これは、イベントの名前と、Common Base Event の *extensionName* 属性に設定された値です。拡張子名は、

Common Base Event に含まれるイベント固有情報を識別するものであり、イベントに関する追加データを提供する XML エレメントの名前でもあります。

状態　ビジネス・プロセス・イベントの状態名を指します。

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。

以下の表で、変数イベントについて説明します。

表 120. 変数イベント

コード	イベント名および拡張子名	状態	イベント性質	説明
21090	VARIABLE_UPDATED BPC.BFM.VARIABLE. STATUS	レポート	CHANGED	変数が更新された。ビジネス・オブジェクト・ペイロードが使用可能です。

変数イベントの場合、以下のイベント相関範囲 ID の内容は次のとおりです。

- ECSCurrentID 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。

関連資料

689 ページの『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。

『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。

690 ページの『ビジネス・プロセス・イベントの拡張子名』
拡張子名は、イベントの有効搭載量を示します。ビジネス・プロセス・イベントのすべての拡張子名と、対応する有効搭載量のリストがここに表示されます。

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

関連資料

709 ページの『ビジネス・プロセス・イベント』
ビジネス・プロセスの Common Base Event は、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に発行されます。プロセスによって、プロセス・イベント、アクティビティー・イベント、アクティビティー・スコープ・イベント、リンク・イベント、および変数イベントが発行される場合があります。

710 ページの『ビジネス・プロセスの Common Base Events』
ビジネス・プロセスの Common Base Event は、WebSphere Integration Developer 内のビジネス・プロセスのモニターが要求された場合に発行されます。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。 これらのイベントは監査ログにも書き込まれます。

716 ページの『アクティビティーの Common Base Event』
アクティビティーの Common Base Event は、WebSphere Integration Developer 内のアクティビティーのモニターが要求された場合に発行されます。ここでは、アクティビティー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。 これらのイベントは監査ログにも書き込まれます。

726 ページの『scope アクティビティーの Common Base Event』
scope アクティビティーの Common Base Event は、WebSphere Integration Developer 内のこれらのアクティビティーのモニターが要求された場合に発行されます。ここでは、アクティビティー・スコープ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。 これらのイベントは監査ログにも書き込まれます。

730 ページの『flow アクティビティーのリンクの Common Base Event』
リンクの Common Base Event は、 WebSphere Integration Developer 内のリンクが定義されている flow アクティビティーのモニターが要求された場合に発行されます。ここでは、リン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。これらのイベントは監査ログにも書き込まれます。

731 ページの『プロセス変数の Common Base Events』
プロセス変数の Common Base Event は、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に発行されます。ここでは、変数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。 これらのイベントは監査ログにも書き込まれます。

第 20 章 ヒューマン・タスク・イベントの概要

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。

イベントは、以下のいずれかの形式をとることができます。

WebSphere Business Monitor 6.0.2 形式

WebSphere Business Monitor 6.0.2 形式のイベントは、WebSphere Integration Developer 6.0.2 でモデリングされているタスクがある場合、または WebSphere Integration Developer 6.1 以降で WebSphere Business Monitor 6.0.2 形式 (レガシー XML) が使用可能になっている場合に発生します。特に明記されていない限り、これらのイベントの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。

WebSphere Business Monitor 6.1 形式

WebSphere Business Monitor 6.1 形式のイベントは、WebSphere Integration Developer 6.1 以降でモデリングされているタスクがあって、WebSphere Business Monitor 6.1 形式 (XML スキーマのサポート) が使用可能になっている場合に発生します。これらのイベントのオブジェクト固有の内容は、Common Base Event の eventPointData フォルダーの `xs:any` スロットに XML エレメントとして書き込まれます。 XML の構造は、XML スキーマ定義 (XSD) ファイル HTMEvents.xsd で定義されています。このファイルは、*install_root*¥ProcessChoreographer¥client ディレクトリーにあります。

関連資料

745 ページの『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。ヒューマン・タスクが発行できるすべてのイベント (つまりタスク・イベントおよびエスカレーション・イベント) の詳細説明については、以下に示す情報を参照してください。

ヒューマン・タスク・イベントの拡張子名

拡張子名は、ヒューマン・タスク・イベントの有効搭載量を示します。ヒューマン・タスク・イベントのすべての拡張子名と、対応する有効搭載量のリストがここに表示されます。

拡張子名には、Common Base Event の *extensionName* 属性の値として使用されるストリング値が含まれています。これは、イベントに関する追加データを提供する XML エレメントの名前でもあります。イベント・エレメントの名前は大文字 (例えば、BPC.HTM.BASE) で、XML 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。

ヒューマン・タスク・イベントに使用できる拡張子名を以下に示します。

BPC.HTM.BASE

- 『BPC.HTM.BASE』

BPC.HTM.ESCALATION

- 738 ページの『BPC.HTM.ESCALATION.BASE』
- 738 ページの『BPC.HTM.ESCALATION.CUSTOMPROPERTYSET』
- 738 ページの『BPC.HTM.ESCALATION.STATUS』
- BPC.HTM.ESCALATION.UPDATED
- 739 ページの『BPC.HTM.ESCALATION.WISTATUS』
- 739 ページの『BPC.HTM.ESCALATION.WITRANSFER』

BPC.HTM.TASK

- 740 ページの『BPC.HTM.TASK.BASE』
- 740 ページの『BPC.HTM.TASK.CUSTOMPROPERTYSET』
- 741 ページの『BPC.HTM.TASK.FAILURE』
- 741 ページの『BPC.HTM.TASK.FOLLOW』
- 742 ページの『BPC.HTM.TASK.INTERACT』
- 742 ページの『BPC.HTM.TASK.MESSAGE』
- 742 ページの『BPC.HTM.TASK.STATUS』
- 742 ページの『BPC.HTM.TASK.UPDATED』
- 744 ページの『BPC.HTM.TASK.WISTATUS』
- 744 ページの『BPC.HTM.TASK.WITRANSFER』

BPC.HTM.BASE

BPC.HTM.BASE は、WBIMonitoringEvent の XML エレメントを継承します。

表 121. BPC.HTM.BASE の XML エレメント

XML エレメント	説明
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。
activityInstanceId	アクティビティー・インスタンスの ID。
displayName	タスク・インスタンスまたはエスカレーション・インスタンスの表示名。

表 121. BPC.HTM.BASE の XML エレメント (続き)

XML エレメント	説明
expirationDate	協定世界時 (UTC) でのタスクの有効期限。ISO 8601 形式 `yyyyMMdd HHmmssZ` で表記します。
isAdHoc	実行時にタスクが作成された場合、この値は true です。
isEscalated	タスクがエスカレートされる場合、この値は true です。
isFollowOn	後続タスクの場合、この値は true です。
isSubTask	サブタスクの場合、この値は true です。
isSuspended	タスクが中断されている場合、この値は true です。
isWaitingForSubTask	タスクがサブタスクを待っている場合、この値は true です。
kind	タスクの種類を示す、次のいずれかの値が含まれます。 101 (ヒューマン・タスク) 105 (参加タスク) 106 (管理用タスク)
parentTaskId	親タスクの ID。親タスクがない場合、これは空のままです。
principal	このイベントに関連したユーザーの名前。
processInstanceId	プロセス・インスタンスの ID。
processTemplateId	プロセス・テンプレートの ID。
state	タスク・インスタンスの現在の状態を示す、次のいずれかの値が含まれます。 1 - INACTIVE 2 - READY 3 - RUNNING 5 - FINISHED 6 - FAILED 7 - TERMINATE 8 - CLAIMED 12 - EXPIRED 101 - FORWARDED
taskInstanceId	タスク・インスタンスの ID。
taskTemplateId	テンプレートの ID。
taskTemplateName	名前空間を含むタスク・テンプレートの名前。この名前は、表示名とは異なることがあります。並列ルーティング・タスクのサブタスクの場合、この値は、親タスク・テンプレートの名前にストリング `$Child` を付加したものになります。
taskTemplateValidFrom	タスク・テンプレートが有効になる日時。

BPC.HTM.ESCALATION.BASE

BPC.HTM.ESCALATION.BASE は、 736 ページの『BPC.HTM.BASE』の XML エレメントを継承します。

表 122. BPC.HTM.ESCALATION.BASE の XML エレメント

XML エレメント	説明
escalationName	エスカレーションの名前。
escalationInstanceDescription	エスカレーションの説明。
escalationTemplateId	エスカレーションのテンプレート ID。

BPC.HTM.ESCALATION.CUSTOMPROPERTYSET

BPC.HTM.ESCALATION.CUSTOMPROPERTYSET は、
『BPC.HTM.ESCALATION.BASE』の XML エレメントを継承します。

表 123. BPC.HTM.ESCALATION.CUSTOMPROPERTYSET の XML エレメント

XML エレメント	説明
username	カスタム・プロパティーを設定したユーザーの名前。
propertyName	カスタム・プロパティーの名前。
propertyValue	カスタム・プロパティーの値。
associatedObjectID	エスカレーション・インスタンス ID である関連オブジェクトの ID。
query	isBinary が true の場合、このエレメントはバイナリー・プロパティーの照会ストリングを指定します。そうでない場合、このエレメントは存在しません。
type	isBinary が true の場合、このエレメントはバイナリー・プロパティーのタイプを指定します。そうでない場合、このエレメントは存在しません。
isBinary	ストリング・カスタム・プロパティーの場合は false に設定し、バイナリー・カスタム・プロパティーの場合は true に設定します。バイナリー・カスタム・プロパティーのペイロード・タイプは Empty に制限されています。プロパティー propertyValue は、バイナリー・カスタム・プロパティーでは省略されます。

BPC.HTM.ESCALATION.STATUS

BPC.HTM.ESCALATION.STATUS は、『BPC.HTM.ESCALATION.BASE』の XML エレメントを継承します。BPC.HTM.ESCALATION.STATUS について、継承されたプロパティー以外に追加して定義する特定のプロパティーはありません。

BPC.HTM.ESCALATION.UPDATED

BPC.HTM.ESCALATION.UPDATED は、 738 ページの
『BPC.HTM.ESCALATION.BASE』の XML エレメントを継承します。

表 124. BPC.HTM.ESCALATION.UPDATED の XML エレメント

XML エレメント	説明
durationUntilEscalated	タスク状態が検査され、それに応じてエスカレーションが発生するか、過剰状態になるまでのカレンダー固有期間。
durationUntilRepeated	エスカレーション・アクションが再実行されるまでのカレンダー固有期間。
escalationTime	このエスカレーションが起動される時刻。
name	エスカレーションの名前。

BPC.HTM.ESCALATION.WISTATUS

BPC.HTM.ESCALATION.WISTATUS は、 738 ページの
『BPC.HTM.ESCALATION.BASE』の XML エレメントを継承します。

表 125. BPC.HTM.ESCALATION.WISTATUS の XML エレメント

XML エレメント	説明
username	作業項目がエスカレートされたユーザーの名前。
reason	作業項目がユーザーに割り当てられた理由。この整数値は、以下のいずれかを意味します。 REASON_NONE (0) REASON_POTENTIAL_OWNER (1) REASON_EDITOR (2) REASON_READER (3) REASON_OWNER (4) REASON_POTENTIAL_STARTER (5) REASON_STARTER (6) REASON_ADMINISTRATOR (7) REASON_ORIGINATOR (9) REASON_ESCALATION_RECEIVER (10) REASON_POTENTIAL_INSTANCE_ CREATOR (11)

BPC.HTM.ESCALATION.WITRANSFER

BPC.HTM.ESCALATION.WITRANSFER は、 738 ページの
『BPC.HTM.ESCALATION.BASE』の XML エレメントを継承します。

表 126. *BPC.HTM.ESCALATION.WITRANSFER の XML エレメント*

XML エレメント	説明
current	現在のユーザーの名前。これは、他のユーザーに転送された作業項目があるユーザーです。
target	作業項目の受取人となるユーザーの名前。
reason	作業項目が転送された理由。この整数値は、以下のいずれかを意味します。 REASON_NONE (0) REASON_POTENTIAL_OWNER (1) REASON_EDITOR (2) REASON_READER (3) REASON_OWNER (4) REASON_POTENTIAL_STARTER (5) REASON_STARTER (6) REASON_ADMINISTRATOR (7) REASON_ORIGINATOR (9) REASON_ESCALATION_RECEIVER (10) REASON_POTENTIAL_INSTANCE_ CREATOR (11)

BPC.HTM.TASK.BASE

BPC.HTM.TASK.BASE は、 736 ページの『BPC.HTM.BASE』の XML エレメントを継承します。

表 127. *BPC.HTM.TASK.BASE の XML エレメント*

XML エレメント	説明
taskInstanceDescription	タスクの説明。
taskInstanceName	タスク・インスタンスの名前。 インライン・タスクの場合は、プロセス・テンプレート名とドル記号からなる接頭部が付きます。 並列ルーティング・タスクのサブタスクの場合、この値は、親タスク・インスタンスの名前に、ストリング `$p` と、そのサブタスクを識別する整数を結合することにより構成されます。例えば、5 番目のサブタスクの場合は *parentTaskName*`$p5` となります。

BPC.HTM.TASK.CUSTOMPROPERTYSET

BPC.HTM.TASK.CUSTOMPROPERTYSET は、『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 128. BPC.HTM.TASK.CUSTOMPROPERTYSET の XML エレメント

XML エレメント	説明
username	カスタム・プロパティーを設定したユーザーの名前。
propertyName	カスタム・プロパティーの名前。
propertyValue	カスタム・プロパティーの値。
associatedObjectID	タスク・インスタンス ID である関連オブジェクトの ID。
query	isBinary が true の場合、このエレメントはバイナリー・プロパティーの照会ストリングを指定します。そうでない場合、このエレメントは存在しません。
type	isBinary が true の場合、このエレメントはバイナリー・プロパティーのタイプを指定します。そうでない場合、このエレメントは存在しません。
isBinary	ストリング・カスタム・プロパティーの場合は false に設定し、バイナリー・カスタム・プロパティーの場合は true に設定します。バイナリー・カスタム・プロパティーのペイロード・タイプは Empty に制限されています。プロパティー propertyValue は、バイナリー・カスタム・プロパティーでは省略されます。

BPC.HTM.TASK.FAILURE

BPC.HTM.TASK.FAILURE は、 740 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 129. BPC.HTM.TASK.FAILURE の XML エレメント

XML エレメント	説明
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。
faultName	障害の名前。

BPC.HTM.TASK.FOLLOW

BPC.HTM.TASK.FOLLOW は、 740 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 130. BPC.HTM.TASK.FOLLOW の XML エレメント

XML エレメント	説明
followTaskId	後続タスクとして開始されたタスクの ID。

BPC.HTM.TASK.INTERACT

BPC.HTM.TASK.INTERACT は、 740 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 131. BPC.HTM.TASK.INTERACT の XML エレメント

XML エレメント	説明
username	タスクに関連したユーザーの名前。

BPC.HTM.TASK.MESSAGE

BPC.HTM.TASK.MESSAGE は、 740 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 132. BPC.HTM.TASK.MESSAGE の XML エレメント

XML エレメント	説明
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**」タブで「**互換性があるイベントのモニター**」オプションが選択されたかどうかによって異なります。

BPC.HTM.TASK.STATUS

BPC.HTM.TASK.STATUS は、 740 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。BPC.HTM.TASK.STATUS について、継承されたプロパティー以外に追加して定義する特定のプロパティーはありません。

BPC.HTM.TASK.UPDATED

BPC.HTM.TASK.UPDATED は、 740 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 133. BPC.HTM.TASK.UPDATED の XML エレメント

XML エレメント	説明
businessRelevant	ビジネス関連タスクと「補助」タスクとを区別できます。
contextAuthorizationOfOwner	指定可能な値は、以下のとおりです。 • 0 = AUTH_NONE。関連コンテキストに対して実行できる操作がないことを示します。 • 3 = AUTH_READER。関連コンテキスト・オブジェクトに対して、読者権限を必要とする操作 (例えば、プロセス・インスタンスのプロパティーの読み取り) を実行できることを示します。
name	タスクの名前。

表 133. *BPC.HTM.TASK.UPDATED* の *XML* エレメント *(続き)*

XML エレメント	説明
namespace	タスクのカテゴリー化に使用される名前空間。
description	タスクの説明。
displayName	タスク・インスタンスの表示名。
priority	タスクの優先順位。
type	タスクのカテゴリー化に使用されるタイプ。
eventHandlerName	アプリケーション・コンポーネントに送信された vetoable イベントを処理する Java オブジェクト。
durationUntilDeleted	タスク・インスタンスが終了状態に到達してから、そのインスタンスが削除されるまでの期間。
deletionTime	スケジュールされた削除時刻、または NULL (削除がスケジュールされていない場合)。
durationUntilDue	このタスクの予想される所要時間を示すカレンダー固有期間。
dueTime	このタスクの予想される終了時刻。
durationUntilExpires	タスクの有効期限が切れるまでのカレンダー固有期間。
expirationTime	このタスクの有効期限が切れる実際の日付。
escalated	このタスクのエスカレーションが発生したかどうかを示します。
parentContextID	このタスクの親コンテキスト。これは、タスクが従属する ID です。 • トップレベル・タスク (サブタスク・ツリーのルート、または後続タスク・チェーンのルート) の場合、このエレメントは、アプリケーション・コンポーネントを作成するタスクによって作成時に設定され、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。例えば、Business Flow Manager の場合、これは PIID、EIID、SIID、または AIID となります。 • サブタスクの場合、これは、すぐ上のレベルにあるタスク・インスタンスの ID です。 • 非インライン・タスクの場合、これは ACOID です。
supportsClaimIfSuspended	中断されたタスクを要求できるかどうかを示します。
supportsDelegation	このタスクを代行できるかどうかを示します。

表 133. BPC.HTM.TASK.UPDATED の XML エレメント (続き)

XML エレメント	説明
supportsFollowOnTasks	後続タスクがサポートされるかどうかを示します。
supportsSubTasks	このタスクのサブタスクを呼び出せるかどうかを示します。

BPC.HTM.TASK.WISTATUS

BPC.HTM.TASK.WISTATUS は、 740 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 134. BPC.HTM.TASK.WISTATUS の XML エレメント

XML エレメント	説明
username	作業項目が作成または削除されたユーザーの名前。
reason	作業項目がユーザーに割り当てられた理由。この整数値は、以下のいずれかを意味します。 REASON_NONE (0) REASON_POTENTIAL_OWNER (1) REASON_EDITOR (2) REASON_READER (3) REASON_OWNER (4) REASON_POTENTIAL_STARTER (5) REASON_STARTER (6) REASON_ADMINISTRATOR (7) REASON_ORIGINATOR (9) REASON_ESCALATION_RECEIVER (10) REASON_POTENTIAL_INSTANCE_ CREATOR (11)

BPC.HTM.TASK.WITRANSFER

BPC.HTM.TASK.WITRANSFER は、 740 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 135. BPC.HTM.TASK.WITRANSFER の XML エレメント

XML エレメント	説明
current	現在のユーザーの名前。これは、他のユーザーに転送された作業項目があるユーザーです。
target	作業項目の受取人となるユーザーの名前。

表 135. BPC.HTM.TASK.WITRANSFER の XML エレメント (続き)

XML エレメント	説明
reason	作業項目が転送された理由。この整数値は、以下のいずれかを意味します。 REASON_NONE (0) REASON_POTENTIAL_OWNER (1) REASON_EDITOR (2) REASON_READER (3) REASON_OWNER (4) REASON_POTENTIAL_STARTER (5) REASON_STARTER (6) REASON_ADMINISTRATOR (7) REASON_ORIGINATOR (9) REASON_ESCALATION_RECEIVER (10) REASON_POTENTIAL_INSTANCE_ CREATOR (11)

関連資料

『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。ヒューマン・タスクが発行できるすべてのイベント (つまりタスク・イベントおよびエスカレーション・イベント) の詳細説明については、以下に示す情報を参照してください。

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。ヒューマン・タスクが発行できるすべてのイベント (つまりタスク・イベントおよびエスカレーション・イベント) の詳細説明については、以下に示す情報を参照してください。

タスクの状態が変更された時点でイベントが発行されます。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。

- 746 ページの『タスク・イベント』
- 749 ページの『エスカレーション・イベント』

注: 実行時に作成されたタスクについては、タスク・モデルでビジネス関連性フラグが true に設定されている場合のみイベントが発行されます。

インライン・タスクは、ヒューマン・タスク・イベントとアクティビティー・イベントの両方を発行できます。アクティビティー・イベントのリストについては、716 ページの『アクティビティーの Common Base Event』を参照してください。

タスク・テンプレート・イベントを除くすべてのヒューマン・タスク・イベントは、CEI と監査証跡の両方で発行できます。タスク・テンプレート・イベント TASK_TEMPLATE_INSTALLED および TASK_TEMPLATE_UNINSTALLED は、監査証跡でのみ発行できます。

XML スキーマ定義 (XSD) ファイル

CEI に送信されるイベントの構造は、スキーマ定義ファイル *install_root*¥ProcessChoreographer¥client¥HTMEvents.xsd に記述されています。

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。WebSphere Business Monitor 6.0.2 形式のイベントの場合、この値は *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。 WebSphere Business Monitor 6.1 形式のイベントの場合、この値は Common Base Event の xs:any スロットに書き込まれます。

イベント名および拡張子名
この列には 2 つの値があります。これは、イベントの名前と、Common Base Event の *extensionName* 属性に設定された値です。拡張子名は、Common Base Event に含まれるイベント固有情報を識別するものであり、イベントに関する追加データを提供する XML エレメントの名前でもあります。

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、750 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	イベント名および拡張子名	状態	イベント性質	説明
51001	TASK_CREATED BPC.HTM.TASK. INTERACT	レポート	CREATED	タスクが作成された
51002	TASK_DELETED BPC.HTM.TASK.STATUS	破棄	DELETED	タスクが削除された

コード	イベント名および拡張子名	状態	イベント性質	説明
51003	TASK_STARTED BPC.HTM.TASK.STATUS	開始	ENTRY	タスクが開始された
51004	TASK_COMPLETED BPC.HTM.TASK.STATUS	停止	EXIT	タスクが完了した
51005	TASK_CLAIM_ CANCELLED BPC.HTM.TASK.STATUS	レポート	DEASSIGNED	要求がキャンセルされた
51006	TASK_CLAIMED BPC.HTM.TASK. INTERACT	レポート	ASSIGNED	タスクが要求された
51007	TASK_TERMINATED BPC.HTM.TASK.STATUS	停止	TERMINATED	タスクが強制終了された
51008	TASK_FAILED BPC.HTM.TASK. FAILURE	失敗	FAILED	タスクが失敗した
51009	TASK_EXPIRED BPC.HTM.TASK.STATUS	レポート	EXPIRED	タスクの期限切れ
51010	TASK_WAITING_FOR_ SUBTASK BPC.HTM.TASK.STATUS	レポート	WAITFORSUBTASK	サブタスクを待機中
51011	TASK_SUBTASKS_ COMPLETED BPC.HTM.TASK.STATUS	停止	SUBTASKCOMPLETED	サブタスクが完了した
51012	TASK_RESTARTED BPC.HTM.TASK.STATUS	レポート	RESTARTED	タスクが再始動した
51013	TASK_SUSPENDED BPC.HTM.TASK.STATUS	レポート	SUSPENDED	タスクが一時停止した
51014	TASK_RESUMED BPC.HTM.TASK.STATUS	レポート	RESUMED	タスクが再開した
51015	TASK_COMPLETED_ WITH_FOLLOW_ON BPC.HTM.TASK. FOLLOW	レポート	COMPLETEDFOLLOW	タスクが完了し、追加タスクが開始した

コード	イベント名および拡張子名	状態	イベント性質	説明
51101	TASK_UPDATED BPC.HTM.TASK.UPDATED	レポート	UPDATED	タスクが更新された
51102	TASK_INPUT_MESSAGE_ UPDATED BPC.HTM.TASK. MESSAGE	レポート	INPUTSET	入力メッセージが更新された。ビジネス・オブジェクト・ペイロードが使用可能です。
51103	TASK_OUTPUT_ MESSAGE_UPDATED BPC.HTM.TASK. MESSAGE	レポート	OUTPUTSET	出力メッセージが更新された。ビジネス・オブジェクト・ペイロードが使用可能です。
51104	TASK_FAULT_ MESSAGE_UPDATED BPC.HTM.TASK. MESSAGE	レポート	FAULTSET	障害メッセージが更新された。ビジネス・オブジェクト・ペイロードが使用可能です。
51201	TASK_WORKITEM_ DELETED BPC.HTM.TASK. WISTATUS	破棄	WI_DELETED	作業項目が削除された
51202	TASK_WORKITEM_ CREATED BPC.HTM.TASK. WISTATUS	レポート	WI_CREATED	作業項目が作成された
51204	TASK_WORKITEM_ TRANSFERRED BPC.HTM.TASK. WITRANSFER	レポート	WI_TRANSFERRED	作業項目が転送された
51205	TASK_WORKITEM_ REFRESHED BPC.HTM.TASK. WISTATUS	レポート	WI_REFRESHED	作業項目が更新された
51301	TASK_CUSTOMPROPERTY_ SET BPC.HTM.TASK. CUSTOMPROPERTYSET	レポート	CP_SET	カスタム・プロパティーが設定された。このイベントは、タスク・インスタンスのカスタム・プロパティーが変更されたときに生成されます。

コード	イベント名および拡張子名	状態	イベント性質	説明
52001	TASK_TEMPLATE_ INSTALLED	レポート	INSTALLED	これらはタスク・テンプレート・イベントであり、監査証跡でのみ発行されます。 Common Base Event としては発行されませんが、完全を期すためにこの表に入っています。
52002	TASK_TEMPLATE_ UNINSTALLED	レポート	UNINSTALLED	

タスク・イベントの場合、以下のイベント相関範囲の ID の内容は次のとおりです。

- 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	イベント名および拡張子名	状態	イベント性質	説明
53001	ESCALATION_UPDATED BPC.HTM.ESCALATION. UPDATED	レポート	UPDATED	エスカレーションが更新された
53201	ESCALATION_WORKITEM_ DELETED BPC.HTM.ESCALATION. WISTATUS	破棄	WI_DELETED	作業項目が削除された
53202	ESCALATION_WORKITEM_ CREATED BPC.HTM.ESCALATION. WISTATUS	レポート	WI_CREATED	作業項目が作成された
53204	ESCALATION_WORKITEM_ TRANSFERRED BPC.HTM.ESCALATION. WITRANSFER	レポート	WI_TRANS- FERRED	エスカレーションが転送された
53205	ESCALATION_WORKITEM_ REFRESHED BPC.HTM.ESCALATION. WISTATUS	レポート	WI_REFRESH- ED	作業項目が更新された

コード	イベント名および拡張子名	状態	イベント性質	説明
51302	ESCALATION_CUSTOMPROPERTY_SET BPC.HTM.ESCALATION.CUSTOMPROPERTYSET	レポート	CP_SET	カスタム・プロパティーが設定された。このイベントは、エスカレーション・インスタンスのカスタム・プロパティーが変更されたときに生成されます。

タスク・イベントの場合、以下のイベント相関範囲の ID の内容は次のとおりです。

- 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。

関連資料

735 ページの『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。

735 ページの『ヒューマン・タスク・イベントの拡張子名』
拡張子名は、ヒューマン・タスク・イベントの有効搭載量を示します。ヒューマン・タスク・イベントのすべての拡張子名と、対応する有効搭載量のリストがここに表示されます。

『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

状態名	Common Base Event の内容	
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

関連資料

745 ページの『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。ヒューマン・タスクが発行できるすべてのイベント (つまりタスク・イベントおよびエスカレーション・イベント) の詳細説明については、以下に示す情報を参照してください。

第 6 部 チューニング

第 21 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。

始める前に

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。

手順

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・ユーザーのために応答時間を短縮することが望ましいことがあります。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。例えばプロセッサー負荷が 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. アプリケーションを調整します。

   アプリケーションには複数のプロセスを含めることができます。 Microflow は長期実行プロセスよりもパフォーマンスが優れているため、パーシスタンスが不要な状況で、機能を 1 つのトランザクションの 1 つのスレッドで処理できる場合は、長期実行プロセスではなく Microflow をモデル化する方式を選択してください。また、長期実行プロセスの分岐を Microflow に分離することを検討してください。さらに、同期サービス呼び出しの速度は通常、非同期サービス呼び出しより速くなります。そのためパフォーマンス上の理由で、同期サービス呼び出しが長期実行プロセスでのデフォルトの動作でないとしても、同期サービス呼び出しを使用してください。

   長期実行プロセスでは、トランザクション境界を変更できます。ほとんどの場合、トランザクション境界の数を減らすことによって、パフォーマンスを向上できます。ただし、トランザクション境界の最適の数はパフォーマンス・テストによってのみ検出できます。データの直列化と非直列化にもコストがかかるため、アクティビティーを直列化する代わりにプロセス内で並列実行パスを使用し、プロセスの一部であるデータのサイズや複雑さを最小化することも検討してください。さらに、発行されるイベントの数を最小にします。

4. プロセスを調整します。

個々のプロセスの調整方法は、そのプロセスが長期実行プロセスか microflow かによって異なります。

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、Apache Derby 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。

タスクの結果

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

長期実行ビジネス・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。

関連タスク

759 ページの『microflow のチューニング』
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、監査ロギングまたは Common Event Infrastructure (CEI) が microflow で使用可能になっている場合にのみ、データベースを使用してテンプレート情報を取得します。microflow のパフォーマンスは、主に、microflow が呼び出すサービスに左右されます。

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。

このタスクについて

長期実行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ング・システムを調整します。データ・ストア・バックエンド・タイプの場合、ファイル・ストアは適切に実行しているた

め、そのファイル・ストアが優先されます。ご使用の環境がクラスターで稼働しており、ファイル・ストアを使用できない場合は、データベース・データ・ストアを使用します。

WebSphere Application Server のサービス統合機能を使用する場合は、WebSphere Application Server インフォメーション・センターにある説明に従って、メッセージング・エンジンのデータ・ストアをセットアップおよび調整してください。

タスクの結果

ご使用のメッセージング・エンジンは最適な状態で作動しています。

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。

手順

WebSphere Application Server のサービス統合機能を使用する場合は、WebSphere Application Server インフォメーション・センターにある、メッセージング・エンジンの JDBC データ・ソースの調整に関する情報を参照してください。

タスクの結果

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。

ビジネス・プロセス・ナビゲーションのパフォーマンスの向上

パフォーマンスの最適化を使用可能にし、各種構成パラメーターを調整することによって、長期実行プロセスのパフォーマンスを調整することができます。

このタスクについて

長期実行プロセスは、複数のトランザクションにわたっています。バージョン 7.0 よりも前のバージョンでは、トランザクションが Java Messaging Service (JMS) メッセージによってトリガーされるのがデフォルトになっていました。バージョン 7.0 では、作業マネージャー・ベースのナビゲーションがデフォルトになっており、これによりパフォーマンスが向上します。ただし、以前のバージョンからマイグレーションし、JMS ベースのナビゲーションを使用していた場合は、作業マネージャー・ベースの実装を使用してトランザクションをトリガーするように Business Flow Manager を構成することによって、プロセス・ナビゲーションのパフォーマンスを向上させることができます。使用するナビゲーション・モードが JMS か作業マネージャーかに関係なく、トランザクション間キャッシュのサイズを調整できます。

以下に、2 つのプロセス・ナビゲーション・モードの特性を要約します。

JMS メッセージ・ベースのナビゲーション
プロセス・ナビゲーションのメッセージ駆動型 Bean (MDB) によって制御される JMS メッセージで処理されるプロセス・ナビゲーション。

- メッセージング・エンジンがアプリケーションに対してローカルになるようにトポロジーがセットアップされている場合、プロセスはサーバー・ア

フィニティーを維持してナビゲートします。ただし、非同期メッセージやヒューマン・タスクなどの外部イベントによってプロセスが起動される場合は除きます。

- 単一アプリケーション・クラスター内の複数のサーバーが単一のリモート・メッセージング・エンジンを使用するようにトポロジーがセットアップされている場合、プロセス内のナビゲーションはクラスター内の複数のサーバーに分散されます。

作業マネージャー・ベースのナビゲーション

プロセス・ナビゲーションは、作業マネージャーが制御するスレッド・プールによって処理されます。プロセス・インスタンスの正常なナビゲーションは、サーバー・アフィニティーを維持して完了します。

トランザクションの整合性を確保するために、ナビゲーション・ステップをトリガーするメッセージは Business Process Choreographer データベースに保管されます。バックグラウンド・リカバリー・スレッドは、定期的にこれらのメッセージをスキャンし、指定された最大経過時間よりも古いメッセージが存在する場合、プロセス・ナビゲーション MDB によって選出させるためにそれらのメッセージを JMS キューに送信します。Business Process Choreographer により、各メッセージが 1 回だけ実行されることが保証されます。

エラーが発生し、ナビゲーション・ステップがロールバックされた場合は、プロセスのナビゲーションは JMS 制御ナビゲーションに戻ります。

サーバー・アフィニティーは、プロセス・インスタンス内のナビゲーションが単一の WebSphere Application Server で行われることを意味します (非同期サービスが起動された場合、待機状態またはタイムアウト状態が発生した場合、receive アクティビティーまたは pick アクティビティーがアクティブ化された場合、ヒューマン・タスクが実行された場合を除く)。これらのイベントの結果、あるプロセス内のナビゲーションが別の WebSphere Application Server で続行される場合があります。

手順

1. Business Flow Manager を構成して、作業マネージャー・ベースのプロセス・ナビゲーションを使用します。

   管理コンソールで、以下のステップを実行します。

   a. 「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「**クラスター名**」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「**サーバー名**」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Business Flow Manager**」をクリックします。

   b. 「**構成**」タブで、「**高度なパフォーマンスの最適化を使用可能にする**」オプションを選択します。 以下の構成パラメーターの値を変更できます。

      - メッセージ・プール・サイズ
      - 停止したメッセージの最大存続期間
      - 停止したメッセージのリカバリー間隔
      - スレッドでの最大処理時間

- トランザクション間キャッシュ・サイズ

2. オプション: ワークフロー・マネージャーが使用可能なスレッドの最大数を増やします。

   Business Flow Manager では、内部処理のために 2 つのスレッドが必要です。残りのスレッドは、プロセス・ナビゲーションで使用可能です。最初に各プロセッサーにスレッドを 1 つ追加します。スレッド・プール・サイズを増加させる場合は、Business Process Choreographer データベース (BPEDB) および接続ファクトリー (BPECFC) の接続プール・サイズも増加させる必要があります。

   スレッドの最大数を変更するには、管理コンソールを使用して以下を実行します。

   a. 「**リソース**」 → 「**非同期 Bean**」 → 「**作業マネージャー**」 → 「**BPENavigationWorkManager**」をクリックします。
   b. 「**スレッド・プールのプロパティー**」の下の、「**スレッドの最大数**」の値を変更します。
   c. 「**作業要求キュー・サイズ**」の値を「**スレッドの最大数**」と同じ値に設定します。

3. 変更を保管します。
4. 変更を有効にするために、サーバーを再始動します。

タスクの結果

これで作業マネージャーがプロセス・ナビゲーションを制御します。

microflow のチューニング

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、監査ロギングまたは Common Event Infrastructure (CEI) が microflow で使用可能になっている場合にのみ、データベースを使用してテンプレート情報を取得します。microflow のパフォーマンスは、主に、microflow が呼び出すサービスに左右されます。

このタスクについて

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。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。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。

手順

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。

2. JVM のガーベッジ・コレクションを調整します。 世代別ガーベッジ・コレクター・ポリシーによって最適なスループットを実現できます。そのポリシーは、

JVM 設定の汎用 JVM 引数としてアクティブ化されます。収集の初期値は、合計ヒープ・サイズの半分に設定してください。例えば、`-Xgcpolicy:gencon -Xmn512M` の場合は、ヒープ・サイズ 1024 MB に対応するポリシーがアクティブ化されます。

3. 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、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。「**アプリケーション・サーバー**」 → 「***server_name***」 → 「**ORB サービス**」 → 「**z/OS 追加設定**」 → 「**ワークロード・プロファイル**」にナビゲートします。
4. 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「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」 → 「**プロパティーの追加**」 → 「**スレッド・プール**」 → 「**デフォルト**」にナビゲートします。

タスクの結果

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。

関連タスク

756 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』
長期実行ビジネス・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。

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、例えば、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。
- 動的担当者照会 (置換変数を使用する照会) を最小限に抑えるか、または回避します。
- 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。

タスクおよびプロセス照会の最適化

タスク・リストおよびプロセス・リストを取得するための query および queryAll API 呼び出しを実行すると、複数のデータベース表の組み合わせを含む複雑な SQL 照会を生成できます。データ表示を最適化することは、パフォーマンス要件、特に複数のユーザーがタスク・リストに同時にアクセスするヒューマン・ワークフロー・アプリケーションのパフォーマンス要件に取り組むのに役立ちます。

このタスクについて

Business Process Choreographer が照会用に調整される場合、応答時間は通常、高負荷であっても、適合サイズのシステム上のサブ秒の領域にあります。照会の応答時間を計算するには、標準のデータベース計算を適用できます。

大容量のヒューマン・ワークフローのシナリオは、照会テーブルでの使用に合わせて最適に調整されています。照会テーブルは、特定の照会に関係のある事前計算データのセットを提供します。例えば、照会プロパティーは照会の実行時にデータベースによってタスクまたはプロセス・インスタンスと結合される必要があります。照会テーブルが使用される場合、こうした SQL 結合は照会実行時にさらに計算される必要はありません。

照会テーブルの実装および保守の取り組みは、標準のデータベース・チューニング技法と比べて大きくなります。照会テーブルを使用する前に、索引、ログ・ファイル配布、およびメモリーなどの、標準のデータベース最適化手法を注意深く検討してください。

テーブルを照会するための 2 つの方法がサポートされています。それは、実体化ビューとカスタム・テーブルです。実体化ビューとカスタム・テーブルのどちらを使用するかは、保守コスト、開発コスト、さらにタスクおよびプロセス・リストの照会によって戻されるデータの現行性に関する要件に基づいて決定してください。

手順

- 実体化ビューは、非同期更新メカニズムを利用する場合に使用します。これは、最適の照会とプロセス・ナビゲーション・パフォーマンスを提供します。
  - 更新は、実体化ビューが使用される場合にのみ行われます。
  - セットアップ、使用、および保守が比較的簡単です。
  - アプリケーション・ソース・コードを変更しないで実装できます。
- カスタム・テーブルは、query または queryAll インターフェースを使用して、他のアプリケーションからのデータを標準照会に組み込む場合に使用します。さらに、カスタム・テーブルは、タスクおよびプロセス照会に必要なデータの表示を最適化するためにも使用できます。
  - データベース・トリガーまたはその他の技法を使用して、タスクおよびプロセス・リストの照会用に最適化されたカスタム・テーブルを同期で更新できます。
  - カスタム・テーブルで提供されたデータを照会するために、照会を変更する必要があります。

関連情報

Business Process Choreographer の query() および queryAll メソッド: ベスト・プラクティス

ヒューマン・ワークフローの調整

DB2 インフォメーション・センター: マテリアライズ照会表

第 22 章 Business Process Choreographer Explorer の調整

以下の提案は、Business Process Choreographer Explorer 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ることを示しています。

手順

1. カスタマイズ・ビューに照会テーブルを使用します。

   Business Process Choreographer Explorer には、事前定義ビューごとにデフォルトの照会テーブル定義が用意されています。しかし、照会テーブルの能力を最大限に活用するためには、照会テーブルをカスタマイズ・ビューの基盤としても使用する必要があります。照会テーブルを使用しないカスタマイズ・ビューがある場合は、ビジネス・シナリオに必要なプロパティーおよびフィルターが正確に含まれている照会テーブルを作成し、この新規照会テーブルに基づいてそれらのビューを再定義することを検討してください。

   照会テーブルに基づかないカスタマイズ・ビューを使用しないようにしてください。照会テーブルに基づかない検索を使用するのは、1 回限りの検索で、特定のフィルター基準を定義するための柔軟性が必要となる場合に限定してください。
2. サーバーの最大ヒープ・サイズを増やすことを検討します。

   Web クライアントは、システム上の負荷を自然に増やします。サーバーに接続されるクライアントが増えると、メモリー内に保持しなければならないオブジェクトが増えます。そのため、サーバーの最大ヒープ・サイズを増やすことを検討してください。これにより、アプリケーションの応答時間が改善され、アプリケーションと並行して作業できるユーザーの最大数が増えます。
3. Web コンテナー・スレッド・プールを調整します。

   スレッド・プールのサイズおよびスレッド非アクティビティー・タイムアウトが Web コンテナーのパフォーマンスに影響を与える可能性があります。これらの設定を変更するには、管理コンソールで「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」をクリックし、次に「**追加プロパティー**」セクションで「**スレッド・プール**」 → 「**WebContainer**」の領域にナビゲートします。

   a. 最大プール・サイズおよび最小プール・サイズを調整します。

      Web クライアント・アプリケーションの HTTP 要求はすべて、Web コンテナー・スレッド・プールからのスレッドを使用して処理されます。 Web クライアントのパフォーマンスを反映するように、最小および最大プール・サイズを調整できます。

      プール内の最大スレッドの数は、アプリケーション・サーバーが同時に処理できる要求の数を表すわけではありません。プール内のすべてのスレッドが使用中の場合、追加要求は、スレッドに割り当てることができるまでキューに入れられます。クライアント要求がスレッドの割り当てを待機する場合、クライアントの応答時間は大きくなります。しかし、最大数の設定が高すぎ

る場合、システムは過負荷になり、その結果クライアントの応答時間はさらに悪くなります。また、他のアプリケーションが著しくスローダウンする可能性もあります。

コンテナーのサイズ変更がパフォーマンス向上につながるかどうかを判別するには、Tivoli® Performance Viewer を使用して、スレッド上の負荷 (PercentMaxed カウンター) および Web コンテナー・モジュールのアクティブ・スレッドの数 (ActiveThreads カウンター) をモニターします。PercentMaxed カウンターの値が常に 2 桁の数字になっている場合、Web コンテナーがボトルネックである可能性があります。この場合、スレッドの数を増やしてください。アクティブ・スレッドの数がプール内のスレッドの数より小さい場合、スレッド・プール・サイズを小さくするとパフォーマンス向上につながることがあります。

b. スレッド非アクティビティー・タイムアウトを調整します。

スレッド非アクティビティー・タイムアウトは、スレッドが再利用される前に経過しなければならない非アクティビティーの期間をミリ秒で定義します。このタイムアウトに小さい値 (1 など) を設定すれば、多数のユーザーがスレッド・プールで空きスレッドを待たずに同時に処理を実行できるようになります。値 0 は待機時間がないことを示します。

4. 大規模なリストのために検索限界の値を小さくします。

大規模タスクまたはプロセス・リストで作業している場合、リストの検索限界を小さくして、ユーザーがアクセスしないデータを収集しないようにすることもできます。この設定を変更するには、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Business Process Choreographer Explorer**」をクリックして、設定を変更します。

Business Choreographer Explorer レポート作成機能の調整

レポート生成に要する時間は、さまざまな要因によって変動します。レポート生成のパフォーマンスを向上させるためのさまざまな方法を以下に示します。

データベース統計を更新する

DB2 データベースでは、実動データベースにデータを取り込んだ時点でデータベース統計を更新すると、パフォーマンスが大幅に向上します。

- DB2 データベースの統計を更新するには、以下のコマンドを入力します。

```
RUNSTATS ON TABLE schema_prefix.EVENT_ACT_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.EVENT_PRC_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.INST_ACT_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.INST_PRC_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.OPEN_EVENTS_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.QUERY_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.SLICES_T WITH DISTRIBUTION AND DETAILED INDEXES ALL;
```

データベースが大規模な (例えばプロセス・インスタンスが 500 000 件を超える) 場合は、RUNSTATS ユーティリティーを実行するときに WITH DISTRIBUTION AND SAMPLED DETAILED INDEXES ALL ステートメントを使用してください。

ここで *schema_prefix* は、Business Choreographer Explorer レポート作成機能用データベースの作成時に使用されたデータベース・スキーマの名前です。データベース統計の更新についての詳細は、ご使用のデータベースの資料を参照してください。

発行イベントの数を削減する

WebSphere Integration Developer では、アクティビティーまたはプロセスのログ記録を非常に詳細なレベルで定義できます。アクティビティー監査イベントがレポート作成の対象になるのは、アクティビティーを含むプロセスに対してもイベントが生成される場合に限られます。プロセスに関連付けることができないアクティビティー・イベントは、イベント・コレクター・アプリケーションでは無視され、データベースに格納されません。発行イベント数を減らすには、以下の手順を実行します。

1. 監査するプロセス・テンプレートを選択し、関心のないプロセスに関するイベントの発行を無効にします。
2. 監査するプロセス・テンプレートのアクティビティーを選択します。レポート結果に影響を与えずに一部のイベントを省略できるかどうかを確認します。

アクティビティーまたはプロセスの正確な情報を取得するには、すべてのイベント・タイプを監査するか、またはイベント・タイプを一切監査しないかのどちらかが必要です。

SQL ユーザー定義関数実装を使用する

レポートを作成するには、 Business Process Choreographer Explorer レポート・データベースに特定のユーザー定義関数 (UDF) をいくつかインストールする必要があります。UDF は SQL ベースの実装と Java ベースの実装として提供されます。SQL 実装は、Java 実装よりも処理速度が高速ですが、欠点もあります。 Java 実装を使用している場合は、SQL 実装への切り替えを検討してください。

SQL 実装と Java 実装の長所と短所について詳しくは、ユーザー定義関数の選択に関するトピックを参照してください。

タイムアウト値を増加させる

レポート生成には長時間かかることがあります。生成にかかる時間が長すぎると、JDBC ドライバーのトランザクション・タイムアウトまたは接続タイムアウトが発生することがあります。この状況が発生した場合は、次のようにタイムアウト値を増加させます。

1. 管理コンソールで、「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「**サーバー名**」をクリックし、次に「**コンテナー設定**」セクションで「**コンテナー・サービス**」を展開し、「**トランザクション・サービス**」をクリックします。
2. 「**合計トランザクション存続時間タイムアウト**」の値が「**最大トランザクション・タイムアウト**」の値よりも小さい場合は、同じ値にします。

3. それでもまだパフォーマンスの問題が発生する場合は、「**合計トランザクション存続時間タイムアウト**」の値を 0 に設定し、「**最大トランザクション・タイムアウト**」の値を増やします。
4. それでもまだパフォーマンスの問題が発生する場合は、「**合計トランザクション存続時間タイムアウト**」と「**最大トランザクション・タイムアウト**」の両方の値を 0 に設定し、JDBC ドライバーの接続タイムアウトの値を増加させます。この操作を実行するには、「**リソース**」 → 「**JDBC**」 → 「**JDBC プロバイダー**」> 「***JDBC プロバイダー***」 → 「**データ・ソース**」 → 「***データ・ソース名***」 → 「**接続プール・プロパティー**」 をクリックしてデータ・ソースの接続プール・プロパティーにナビゲートし、「**接続タイムアウト**」の値を増加させます。

サーバー・クラスターでは、すべてのクラスター・メンバーのトランザクション・タイムアウト値を調整する必要があります。

不要なデータを削除する

レポートのパフォーマンスは、レポート・データベース内のインスタンスとイベント・データの容量に応じて変化します。レポート生成のために大量のデータを照会すると、パフォーマンスは低下します。レポート・データベース内のプロセス・インスタンスやアクティビティー・インスタンスの数を減らすと、レポートのパフォーマンスを向上させることができます。不要な情報や古い情報を定期的に削除すると、パフォーマンスの向上に役立ちます。

第 7 部 トラブルシューティング

第 23 章 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、Business Process Choreographer とその Business Flow Manager、または Human Task Manager コンポーネントの構成に関連した問題を解決します。

このタスクについて

このセクションの目的は、Business Flow Manager または Human Task Manager 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、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 に対するプロファイル・アクションでは、プロファイル・ツールの logs ディレクトリーにある bpcaugment.log ファイルに書き込みが行われます。より詳細なトレースは、同じディレクトリーにある bpcaugment.wsadmin.log ファイルに書き込まれます。

これらのファイルは、*install_root*¥logs¥manageprofiles¥*profileName*¥logs にあります。

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、プロファイルの logs ディレクトリーの bpeconfig.log ファイルにアクションが記録されます。このディレクトリーは、*profile_root* ディレクトリーにあります。

管理スクリプト

ProcessChoreographer ディレクトリーの admin 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・ファイルを作成しません。wsadmin を使用して実行されるすべての Business Process Choreographer スクリプトは、プロファイル・ツールの logs ディレクトリー内の wsadmin.traceout ファイルに記録されます。ただし、このファイルは wsadmin が起動されるたびに上書きされるので、必ず、-tracefile オプションと -appendtrace オプションのどちらかを使用するか、または、wsadmin を再度起動する前にログ・ファイルを保管してください。

構成関連スクリプト

bpeconfig.jacl、bpeupgrade.jacl、clientconfig.jacl、および bpeunconfig.jacl のスクリプト・ファイルは、それぞれ bpeconfig.log、bpeupgrade.log、clientconfig.log、および bpeunconfig.log という名前でそれぞれのログ・ファイルを logs ディレクトリーに書き込みます。

以下の構成スクリプトは、logs ディレクトリー内のログ・ファイルを setupEventCollector.log ファイルに書き込みます。

- setUpEventCollector.sh

また、wsadmin.traceout ファイルも確認します。

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。

このタスクについて

Business Flow Manager および Human Task Manager はどちらもデータベースを必要としています。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。

手順

- 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しているとき、またはサーバーが始動するときに、"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、DB2 の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てください。
    1. データ・ソースのクラス・パス設定を確認します。デフォルトのセットアップでは、WebSphere 変数 ${DB2UNIVERSAL_JDBC_DRIVER_PATH} は、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・ソースの実装クラス COM.ibm.db2.jdbc.DB2XADataSource について述べるエラーが発生した場合:

  - 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を確認します。

  - コンポーネントで管理される認証別名が、Business Process Choreographer がサーバー上で構成されている場合は、BPCDB_*nodeName.serverName_Auth_Alias* に設定され、Business Process Choreographer がクラスター上で構成されている場合は BPCDB_*clusterName_Auth_Alias* に設定されていることを確認します。

– 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。

• Derby 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。

  – ij コマンド行プロセッサーを呼び出そうとして「Java クラスが見つかりません (Java class not found)」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こと、および classpath 環境変数に以下の JAR ファイルが含まれていることを確認してください。

    - derby.jar

    - derbytools.jar

– 組み込み Derby ドライバーを使用しているのに、(ij などの) Derby ツールを使用して Derby 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 Derby may have already booted the database
  c:¥WebSphere¥AppServer¥profiles¥profile_name¥databases¥BPEDB.
```

Derby データベースには一度に 1 つのアプリケーションしかアクセスしていない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は、ビジネス・プロセス・コンテナーが使用するデータベース・システムが稼働しており、アクセス可能であることを確認します。 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。
- 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。
- データベース内にテーブルおよびビューが見つからず、スキーマ作成オプションが使用可能になっていない場合は、以下を確認してください。
  – データベース・スキーマ修飾子が構成されている場合は、以下を確認してください。
    - スキーマ修飾子はデータベース内のスキーマに一致している必要があります。スキーマは、スクリプトで使用されているスキーマと同じである必要があります。
    - データベースのテーブルとビューを操作する特権をユーザーに付与する必要があります。
  – スキーマ修飾子が構成されていない場合は、以下を確認してください。
    - ユーザーの認証別名は、スクリプトの実行に使用されるのと同じユーザー ID であるか、またはスクリプトで使用されているスキーマ修飾子に一致している必要があります。
    - データベースのテーブルとビューを操作する特権をユーザーに付与する必要があります。
- スキーマ作成オプションが使用可能になっており、データベースのテーブルとビューが見つからない場合は、以下の条件を使用してデータベース表とオブジェクトが自動的に作成されます。
  – スキーマ修飾子が構成されている場合は、そのスキーマ修飾子を使用してテーブルとビューが作成されます。
  – スキーマ修飾子が構成されていない場合は、ユーザー ID を使用してテーブルとビューが作成されます。

REST API: URL が正しく構成されていない

Representational State Transfer (REST) API を正しく構成する必要があります。そうでないと、Business Process Choreographer Explorer または Business Space 内のプロセス状態ビュー・ウィジェットを使用しようとすると、エラーが発生します。

理由

以下の原因が考えられます。

- クラスター環境でグラフィカル・プロセス・ウィジェットを使用する場合、Business Flow Manager REST API と Human Task Manager REST API のエンドポイントを手動で設定する必要があります。
- 単一クラスターで Business Process Choreographer Explorer を構成済みの場合、ロード・バランシングを達成するために Web サーバーについて正しいホスト名とポートを構成する必要があります。
- コンテキスト・ルートを変更、または Web サーバーに Web モジュールをマップする場合、REST API の URL を変更する必要があります。

解決方法

この問題を訂正するには、次の指示に従います。

- Business Process Choreographer Explorer インスタンスを構成した場合、メッセージ CWWBZ0052W または CWWBZ0053W がないか、ログ・ファイルを確認します。これらのメッセージには、インスタンスの構成時に指定された、使用 URL に関する情報が含まれています。
- 単一セルに複数の Business Process Choreographer 構成を保有しており、 Business Flow Manager (BPEContainer アプリケーション) と Human Task Manager (TaskContainer アプリケーション) の REST API Web モジュールが同じ Web サーバーにマップされている場合、これらの Web モジュールのコンテキスト・ルートは固有でなければなりません。
  1. Business Flow Manager に対してコンテキスト・ルートを設定するには、「**アプリケーション**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere エンタープライズ・アプリケーション**」をクリックし、次に「**BPEContainer_*suffix***」 → 「**Web モジュールのコンテキスト・ルート**」をクリックします。ここで、*suffix* は、Business Process Choreographer が構成されている *node_name_server_name* または *cluster_name* です。次に、Web モジュールの BFMRESTAPI および BFMJAXWSAPI のコンテキスト・ルートが正しく固有であることを確認します。
  2. Human Task Manager に対してコンテキスト・ルートを設定するには、「**アプリケーション**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere エンタープライズ・アプリケーション**」をクリックし、次に「**TaskContainer_*suffix***」 → 「**Web モジュールのコンテキスト・ルート**」をクリックします。ここで、*suffix* は、Business Process Choreographer が構成されている *node_name_server_name* または *cluster_name* です。次に、Web モジュールの HTMRESTAPI および HTMJAXWSAPI のコンテキスト・ルートが正しく固有であることを確認します。

6.0.x Business Process Choreographer API クライアントが バージョン 7.0 環境で失敗する

WebSphere Process Server バージョン バージョン 7.0 にアップグレードしたときに、6.0.x Business Process Choreographer API クライアントをマイグレーションしていませんでした。バージョン 7.0 環境でクライアントを実行しようとすると、そのクライアントは失敗します。

症状

以下のような例外が SystemOut.log ファイルに書き込まれます。

```
[9/6/07 21:05:27:093 PDT] 00000045 ExceptionUtil E CNTR0020E: EJB threw an unexpected
(non-declared) exception during invocation of method "processMessage" on
bean "BeanId(validateDataApp#validateDataEJB.jar#component.validateItem, null)".
Exception data: javax.ejb.AccessLocalException: ;
nested exception is: com.ibm.websphere.csi.CSIAccessException:
SECJ0053E: Authorization failed for /UNAUTHENTICATED while invoking
(Home)com/ibm/bpe/api/BusinessFlowManagerHome create:4
securityName: /UNAUTHENTICATED;accessID: UNAUTHENTICATED is not granted any of the required
roles: BPEAPIUser
com.ibm.websphere.csi.CSIAccessException: SECJ0053E: Authorization failed for
/UNAUTHENTICATED while invoking (Home)com/ibm/bpe/api/BusinessFlowManagerHome
create:4 securityName: /UNAUTHENTICATED;accessID: UNAUTHENTICATED is not granted any of the required
roles: BPEAPIUser
at com.ibm.ws.security.core.SecurityCollaborator.performAuthorization(SecurityCollaborator.java:484)
at com.ibm.ws.security.core.EJSSecurityCollaborator.preInvoke(EJSSecurityCollaborator.java:218)
at com.ibm.ejs.container.EJSContainer.preInvokeForStatelessSessionCreate(EJSContainer.java:3646)
at com.ibm.ejs.container.EJSContainer.preInvoke(EJSContainer.java:2868)
at com.ibm.bpe.api.EJSLocalStatelessGenericBusinessFlowManagerEJBHome_a412961d.create(Unknown Source)
```

理由

ユーザーを最初に認証せずに Business Process Choreographer API を使用するクライアントを作成していた場合は、それらの API を使用する前にログインを実行するよう、そのクライアントを修正してください。マイグレーション後、Java EE ロール BPEAPIUser および TaskAPIUser は、値 Everyone に設定されます。これにより、アプリケーション・セキュリティーが有効な場合にログインを要求しないという 6.0.x の振る舞いを維持して、以前のバージョンとの互換性が保たれます。新規インストールでは、これらのロールはデフォルト値の AllAuthenticated に設定されます。Everyone を使用した Java EE ロール BPEAPIUser と TaskAPIUser のマッピングは、非推奨になっています。

解決方法

ユーザーを強制的に API クライアントにログオンさせるよう、API クライアントを修正して、API を使用します。

一時的な回避策として、BPEAPIUser ロールおよび TaskAPIUser ロールに対するマッピングを変更することができます。このマッピングを変更するには、以下のようにします。

1. 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**」 → 「**BPEContainer_*suffix***」をクリックし、「**詳細プロパティー**」の下の「**ユーザー/グループ・マッピングへのセキュリティー・ロール**」をクリックします。

2. BPEAPIUser ロールを AllAuthenticated から Everyone に変更し、「**OK**」をクリックします。
3. TaskContainer_suffix および TaskAPIUser ロールについてステップ 2 を繰り返します。
4. クライアントの変更が完了したら、これらのロールを AllAuthenticated に戻して、非認証ユーザーが API にアクセスできないようにします。

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。

```
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担当者ディレクトリー・プロバイダーは、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。

- WebSphere Application Server FFDC ログは、ffdc フォルダーにあります。
- 開始されたタスクの SDSF ジョブ・データ・セットに書き込まれる出力。

第 24 章 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。

このタスクについて

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

このトピックでは、ビジネス・プロセス、ヒューマン・タスク、またはその両方が含まれるアプリケーションをインストールするときに発生する可能性のある問題の症状および解決方法について説明します。

症状: ビジネス・プロセスまたはヒューマン・タスクのインストール後に例外が発生する

ビジネス・プロセス、ヒューマン・タスク、またはその両方が含まれる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・マネージャーまたはスタンドアロン・サーバーの `SystemOut.log` ファイルに以下のような例外が記録されます。

- CWWBF0064E: server1 がビジネス・プロセス・アプリケーションを実行するように構成されていません。
- CWTCO0017E: server1 がヒューマン・タスク・アプリケーションを実行するように構成されていません。

理由

デプロイメント・ターゲット上にビジネス・プロセス・コンテナーもヒューマン・タスク・コンテナーも構成されていません。

解決方法

ビジネス・プロセスおよびヒューマン・タスクの機能を使用するには、ビジネス・プロセス・コンテナーとヒューマン・タスク・コンテナーの両方を構成する必要があります。コンテナーの構成について詳しくは、「*関連情報*」のセクションを参照してください。

症状: インストールおよび構成リポジトリーの更新が正常終了した後にアプリケーションが開始しない

ビジネス・プロセス、ヒューマン・タスク、またはその両方が含まれるアプリケーションが、正常にインストールされた後に開始しません。これは、構成変更が管理

コンソールで保管されたか、wsadmin ツールを使用して保管されたことを意味します。

理由

ビジネス・プロセスまたはヒューマン・タスクが含まれるアプリケーションのインストールは、2 段階に分かれています。第 1 段階は、構成変更が構成リポジトリーに保管された時点までです。その後は、次の段階になります。この第 2 段階 (デプロイメントと呼ばれる) では、アプリケーション内で検出されたビジネス・プロセス・テンプレートとヒューマン・タスク・テンプレートが Business Process Choreographer データベースに格納されます。Network Deployment 環境内で構成リポジトリーを同期化するとき、またはこのアプリケーションを開始しようとしたときに、このステップが始まります。

アプリケーション内のテンプレート数および使用するハードウェアによっては、デプロイメント段階で時間がかかっているためにアプリケーションが開始しないことがあります。

Business Process Choreographer データベースへのデプロイメント時に問題があることも考えられます。この場合は、ログ、トレース、および FFDC を調べて詳細情報を得る必要があります。

解決方法

デプロイメント環境に応じて、SystemOut.log ファイル、SystemErr.log ファイル、および FFDC を調べます。

デプロイメント環境がスタンドアロン・サーバーの場合は、このサーバーにあるこれらのログおよび FFDC を探します。

デプロイメント環境が Network Deployment 環境の場合は、デプロイメント・ターゲットに含まれるすべてのサーバーと、それらのサーバーを管理するすべてのノード・エージェントにあるこれらのログおよび FFDC を探します。

これらのログおよび FFDC に問題の徴候が見られない場合は、以下のトレースを有効にして、サポート担当員に連絡してください。

```
*=info: com.ibm.bpe.*=all: com.ibm.task.*=all: com.ibm.ws.staffsupport.*=all
```

デプロイメント環境がスタンドアロン・サーバーの場合は、このサーバーでトレースを有効にします。

デプロイメント環境が Network Deployment 環境の場合は、デプロイメント・ターゲットに含まれるすべてのサーバーと、それらのサーバーを管理するすべてのノード・エージェントでトレースを有効にします。

症状: 前のレベルの WebSphere Process Server にアプリケーションがデプロイされない

新しいバージョンの WebSphere Integration Developer で作成したアプリケーションが WebSphere Process Server にインストールされません。

理由

WebSphere Process Server ランタイム・バージョンは、インストールしようとしている .EAR ファイルのバージョンと同じかそれ以降のものでなければなりません。

解決方法

WebSphere Integration Developer によって生成された .EAR ファイルのバージョンと同じかそれ以降の WebSphere Process Server バージョンを使用してください。あるいは、適切なバージョンの WebSphere Integration Developer を使用してください。

症状: 混合バージョンのクラスターにアプリケーションがデプロイされない

複数のバージョンのメンバーが混在している (クラスター・メンバーの一部を最近マイグレーションした) クラスターでは、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両方が含まれているアプリケーションを、インストール、更新、またはアンインストールすることはできません。

理由

混合バージョン環境では、ビジネス・プロセスまたはヒューマン・タスクが含まれるアプリケーションのインストール、更新、またはアンインストールは、インストールするバージョンにかかわらず、サポート**されません**。

解決方法

マイグレーションを完了してから、これらのアプリケーションをインストール、更新、またはアンインストールします。

症状: 共用ライブラリーを使用している場合、コード生成が機能しない

ビジネス・プロセスが含まれるアプリケーションから共用ライブラリーにアクセスしている場合は、アプリケーションがインストールされない場合があり、以下のようなエラーが発生します。

```
com.ibm.bpe.plugins.DeploymentCodeGenerationCompileFailedException:
CWWBD0338E: BPEL ファイル com/ibm/test/bpel/DeployTestBpel.bpel' の
Java コードのコンパイルが失敗しました。
```

理由

アプリケーションのインストールと共用ライブラリーに関する既知の制限があります。詳しくは、技術情報 1268185 を参照してください。

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アンインストールのトラブルシューティング

このトピックでは、ビジネス・プロセス、ヒューマン・タスク、またはその両方が含まれるアプリケーションをアンインストールするときに発生する可能性のある問題の症状および解決方法について説明します。

症状: インスタンスが存在することが原因でアプリケーションのアンインストールが失敗した

ビジネス・プロセス、ヒューマン・タスク、またはその両方が含まれるアプリケーションをアンインストールするときに、デプロイメント・マネージャーまたはスタンドアロン・サーバーの SystemOut.log ファイルに以下のような例外が記録されます。

- CWTCO0006E: ヒューマン・タスク *task_name* がインスタンスを取得しました。インスタンスを除去してからアプリケーションをアンインストールしてください。
- CWWBF0025E: プロセス *process_name* には依然としてインスタンスがあります。プロセス・アプリケーションを更新またはアンインストールする前に、すべてのプロセス・インスタンスを終了して削除してください。

デプロイメント・マネージャーまたはスタンドアロン・サーバーの SystemErr.log ファイルにも、同様の例外が記録されます。

理由

アンインストールしようとしているアプリケーションには、ビジネス・プロセス、ヒューマン・タスク、またはその両方が含まれています。このビジネス・プロセスまたはヒューマン・タスクのテンプレートのうち、関連インスタンスが存在するものが 1 つ以上あります。ビジネス・プロセス、ヒューマン・タスク、またはその両方が含まれるアプリケーションをアンインストールするためには、関連インスタンスが存在してはなりません。

このルールの唯一の例外は、スタンドアロン・サーバーを使用していて、このサーバーの「**開発モードでの実行**」オプションが有効になっている場合です。この場合、既存のインスタンスがありますが、アプリケーションをアンインストールすることは可能です。

解決方法

これらのビジネス・プロセスまたはヒューマン・タスクのインスタンスがアプリケーションの一部として存在しないようにします。 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・インスタンスを表示し、これらを削除します。

アプリケーションをアンインストールするには、681 ページの『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』または 682 ページの『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』の説明に従ってください。

症状: インスタンスが存在することが原因でアプリケーションのアンインストールが失敗したが、それらのインスタンスが見つからない

ビジネス・プロセス、ヒューマン・タスク、またはその両方が含まれるアプリケーションをアンインストールするとき、このアプリケーションに関連するビジネス・プロセス・インスタンスまたはヒューマン・タスク・インスタンスが存在すること

が原因でアンインストールが失敗しましたが、これらのインスタンスを照会できません。アプリケーションはアンインストールされませんでした。

理由

このような問題が発生する可能性がありますが、この障害の一般的の理由を判別することは困難な場合があります。

解決方法

Business Flow Manager システム管理者および Human Task Manager システム管理者と一緒に、アプリケーションに属するすべてのビジネス・プロセス・インスタンスおよびヒューマン・タスク・インスタンスが削除されていることを確認してください。これは、実稼働環境でも推奨される方法です。完了したプロセス・インスタンスを削除するには、Business Process Choreographer Explorer を使用するか、327 ページの『完了したプロセス・インスタンスの削除』のトピックで説明するスクリプトを使用します。

-force オプションを使用してアプリケーションを強制的にアンインストールするには、bpcTemplates.jacl スクリプトを使用します。**注意: 実稼働環境での -force オプションの使用は推奨されません。** bpcTemplates.jacl スクリプトを使用するには、682 ページの『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』の説明に従ってください。このアクションによって、アプリケーションのアンインストール中に既存のすべてのプロセス・インスタンスおよびタスク・インスタンスが削除されます。

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。

このタスクについて

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。

手順

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。

次のタスク

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。しかし、これらのクラスを必要とする Microflow インスタンスは、まだ実行し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。

processMessage メソッドの呼び出し中に予期しない例外が発生しました (メッセージ: CNTR0020E)

ビジネス・プロセス・コンテナーが停止していたため、クライアントがサーバーに接続できませんでした。

解決方法

ビジネス・プロセス・コンテナーが実行していることを確認してください。

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

以下のすべての条件が発生した場合は、アクティビティーが停止状態になります。

- アクティビティーの実装によって障害が生成された場合、あるいはアクティビティーに関連した条件、タイマー、またはカウンターの値 (例えば、アクティビティーの結合条件や、アクティビティーの発信リンクの遷移条件のいずれかなど) の評価時に障害が生成された場合。
- エンクロージング・スコープで障害が処理されない。
- 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 断片については、以下のいずれかの条件が発生した場合。
  - プロセスの continueOnError 属性が `no` に設定されていて、アクティビティーの continueOnError 属性が `inherit` または `no` に設定されている場合。
  - プロセスの continueOnError 属性が `yes` に設定されていて、アクティビティーの continueOnError 属性が `no` に設定されている場合。
- 他のすべてのアクティビティーについては、プロセスの continueOnError 属性が `no` に設定されている場合。

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。

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. Failed Event Manager コンソールを使用して、失敗メッセージの詳細を表示し、失敗イベントを再生します。
2. 管理コンソールの失敗イベント管理ビューに失敗メッセージが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イベントがある場合は、そのメッセージを再アクティブ化します。
   - 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

サブプロセスを呼び出すとリモートの EJB メソッドが呼び出されるため、別の EAR ファイル内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しておく必要があります。

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。

長期実行プロセスが同期的に呼び出された場合のハング・スレッド (メッセージ: WSVR0605W)

長期実行プロセスによって、別の長期実行プロセスが同期的に呼び出されます。作業負荷の高い条件下では、スレッド・モニターによって、ハング・スレッドが SystemOut.log ファイル (メッセージ WSVR0605W) で報告されます。

理由

同期的に呼び出される長期実行プロセスが原因で、ハング・スレッドが発生することがよくあります。長期実行プロセスは、通常、複数のトランザクションにまたがるため、ナビゲーションを続けるには、空きスレッドが必要になります。使用可能なすべてのスレッドが、サブプロセスを呼び出した親プロセスのナビゲーション・ステップに関与している場合、システムが応答しなくなります。空きスレッドがないために、サブプロセスが完了できません。

解決方法

長期実行プロセスが別の長期実行プロセスを呼び出す場合は、これらのプロセスが別のコンポーネントで分離されていたとしても、常に非同期に呼び出すようにする必要があります。例えば、長期実行プロセスがメディエーションを呼び出し、このメディエーションが別の長期実行プロセスを呼び出す場合、メディエーションの優先対話形式が非同期であることを確認してください。

実行時バインディングによって誤ったバージョンのサブプロセスが呼び出される

親プロセスが、実行時バインディングを使用してサブプロセスを呼び出します。両方のプロセスは、同じモジュール内にあります。モジュールをコピーし、有効開始タイム・スタンプを変更することにより、新しいバージョンのサブプロセスが作成されます。そのモジュールがデプロイされた後、親プロセスの実行中のインスタンスが、サブプロセスの新しいバージョンではなく古いバージョンを引き続き呼び出します。

理由

実行時バインディングでは、サブプロセスのプロセス・テンプレート名が、親プロセスの invoke アクティビティーの参照パートナー・プロパティーの一部として指定されます。Business Process Choreographer は、現在有効なプロセスのバージョンを実行時に判別します。

誤ったバージョンのサブプロセスを使用して実行時バインディングが行われる一般的な理由は、サブプロセスを含むモジュールに Service Component Architecture (SCA) エクスポートがないことです。エクスポートがないと、他のモジュール内の

プロセスが親プロセスに対して可視にならず、親プロセスでは、同じモジュール内に存在するバージョンのサブプロセスを常に呼び出します。

解決方法

WebSphere Integration Developer のアセンブリー・エディターで、新しいバージョンのサブプロセスに対する SCA ネイティブ・バインディングを備えた SCA エクスポートを生成します。

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、「技術情報 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。

プロセス・インスタンスが失敗状態にあるために、要求された sendMessage アクションを実行できません (メッセージ: CWWBE0126E)

プロセス・インスタンスにイベントを送信しようとすると、'EngineProcessWrongStateException' というメッセージを受け取ります。

理由

このエラーの共通の理由は、新規プロセス・インスタンスを作成するためにメッセージを receive または pick アクティビティーに送信しても、新規プロセス・インスタンスをインスタンス化できないというものです。この状況が発生するのは、このアクティビティーによって使用される相関セットの値 (このメッセージと一緒に渡される) が、既に失敗状態にある既存のプロセス・インスタンスに一致している場合です。

解決方法

この問題を訂正するには、既存のプロセス・インスタンスを削除するか、または既存のプロセス・インスタンスに一致しない相関セットの値を渡します。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、変数を定義するときに初期値を指定し、assign アクティビティーを指定することによって実行できます。例えば、変数を assign の to-spec で指定する必要があります。あるいは、Java 断片の中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。

障害ハンドラーが障害を catch しない

invoke アクティビティーには、呼び出されたサービスによってスローされる特定の障害を catch するために障害ハンドラーが関連付けられています。しかし、呼び出されたサービスが予期される障害を返したのに、障害ハンドラーが実行されません。

理由

この問題の一般的な理由は、障害に関連付けられたデータを catch するための障害変数が、障害ハンドラーに指定されていないことです。障害が障害データに関連付けられている場合、障害ハンドラーは、以下のいずれかの状況に該当したときに限り、その障害を catch します。

- 障害ハンドラーの名前がその障害名に一致し、傷害に関連付けられたデータのタイプに一致するデータ型の障害変数が指定されている。
- 障害ハンドラーには障害名が指定されていないが、傷害に関連付けられたデータのタイプに一致するデータ型の障害変数が指定されている。
- catchAll 障害ハンドラーが指定されている。

解決方法

障害ハンドラーに障害変数を追加してください。障害変数のデータ型が、障害に関連付けられたデータのタイプに一致していることを確認してください。

関連概念

42 ページの『ビジネス・プロセスの障害データの取得』
プロセスは、ランタイム障害および BPEL 標準の障害を処理できます。これらの障害を処理するために、障害に関する情報にアクセスする必要がある場合があります。

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 Derby、Oracle または Informix 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。これは、これらのデータベース・システムのデータベース・エンティティーのロックが、例えば DB2 データベースほど細分化されていないためです。ただし、並列分岐によって非同期的にトリガーされるサービスは、依然として並列で実行されます。つまり、これらのデータベース・システムに対して直列化されるのはプロセス・ナビゲーションのみです。

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

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。

このタスクについて

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。

手順

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。
      - *profile_root*/profiles/*profile_name*/logs/activity.log
   b. 「**開く (Open)**」をクリックします。

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。

このタスクについて

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。

手順

- 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安全に停止できます。この問題を回避する方法について詳しくは、「技術情報 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。

- 呼び出しタスクによって開始される長期実行ビジネス・プロセスは、開始に失敗します。 JSP スニペットは、呼び出しタスクをユーザーが使用できるようにします。以下の例では、同期呼び出しパターン createAndCallTask が使用されています。この場合、長期実行ビジネス・プロセスは開始に失敗します。

```
HumanTaskManager htm = ...
TaskTemplate taskTemplate = htm.getTaskTemplate( "start the process" );
Task task = htm.createAndCallTask( taskTemplate.getTKTID() );
while (task.getState() != TASK.TASK_STATE_FINISHED)
{
     Sleep(100);
}
```

長期実行プロセスはいくつかのトランザクションで構成されており、その呼び出しスタイルは非同期です。そのため、長期実行プロセスを開始するには、非同期呼び出しパターンである createAndStartTask を使用する必要があります。

```
HumanTaskManager htm = ...
TaskTemplate taskTemplate = htm.getTaskTemplate( "start the process" );
Task task = htm.createAndStartTask( taskTemplate.getTKTID() );
while (task.getState() != TASK.TASK_STATE_FINISHED)
{
     Sleep(100);
}
```

さらに、JSP デプロイメント記述子内の transaction 属性は `NotSupported` に設定する必要があります。これにより、トランザクションがなくてもコード・スニペットを実行し、createAndStartTask メソッドで新規トランザクションを開いて、プロセス・インスタンスを開始することができます。このトランザクションは createAndStartTask メソッドが戻るとコミットされ、メッセージが表示されます。

完了以外の状態に対しては「while」ループを含めてください。例えば、プロセスの実行中にアクティビティーが失敗した場合、終了状態は TASK.TASK_STATE_FAILED になることがあります。

エスカレーション E メールのトラブルシューティング

この情報を使用して、E メールのエスカレーションに関連した問題を解決します。

このタスクについて

エスカレーションは、ヒューマン・タスクが予期されたとおりに進行しないとトリガーされます。エスカレーションは作業項目を作成します。また、エスカレーションの影響を受けるユーザーに E メールを送信することもできます。エスカレーション E メールに問題がある場合、このトピックにある情報を使用して問題を解決してください。

手順

- `SystemOut.log` ファイルで、ユーザーの割り当てまたは E メール・アドレスに関するエラー・メッセージを調べます。
- `SystemOut.log` ファイルに関係のあるメッセージが含まれていない場合、メール・セッション・サーバーに対してデバッグ・モードを有効にします。

管理コンソールで「**リソース**」 → 「**メール**」 → 「**メール・セッション**」をクリックし、次に「***HTMMailSession_server***」をクリックして、「**デバッグ・モードを使用可能にする**」チェック・ボックスを選択します。 エスカレーション E メールが送信されると、デバッグ情報が SystemOut.log ファイルに書き込まれます。

- Virtual Member Manager を担当者ディレクトリー・プロバイダーとして使用しているときに、E メール・アドレスに問題がある場合、Staff.Diagnosis カスタム・プロパティーを使用可能にします。
  1. 管理コンソールで、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」をクリックします。
  2. 「**構成**」タブの「**追加プロパティー**」の下で、「**カスタム・プロパティー**」 → 「**Staff.Diagnosis**」をクリックし、「**値**」フィールドに on を入力します。

  エスカレーション E メールが送信されると、担当者の割り当てに関する追加情報が SystemOut.log ファイルに書き込まれます。
- Human Task Manager 保留キューにメッセージが含まれているかどうか確認します。
  1. 管理コンソールで、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」をクリックします。
  2. 「**ランタイム**」タブで、「**保留キューの再生**」をクリックします。 保留キューのメッセージが「**保留キュー・メッセージ**」フィールドに表示されます。

  保留キューにメッセージが含まれている場合、サーバーの First Failure Data Capture (FFDC) ディレクトリーで、エラーの詳細を調べてください。
- カスタム・プロパティーの値を調べて、E メールが再送される回数および E メールを送信するためのタイムアウトを確認します。
  1. 管理コンソールで、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」をクリックします。
  2. 「**構成**」タブの「**追加プロパティー**」セクションで、「**カスタム・プロパティー**」をクリックします。
  3. **EscalationEmail.RetryTimeout** および **EscalationEmail.MaxRetries** フィールドの値を確認します。

     EscalationEmail.RetryTimeout

     Human Task Manager が失敗した E メール通知を再送するまで待機する時間を指定します。このフィールドのデフォルト値は 3600 秒 (1

時間) です。再試行が失敗した場合、再試行タイムアウトは再試行の失敗ごとに動的に 2 倍にされます。デフォルトでは、最初の再試行が失敗すると、2 時間後にもう一度再試行されます。

EscalationEmail.MaxRetries

Human Task Manager が、失敗した E メール通知の再送を試みる回数を指定します。このフィールドのデフォルト値は 4 回の再試行です。このフィールドの値が 0 に設定される場合、失敗した E メール通知は再送されません。すべての再試行が失敗すると、メッセージは保留キューに書き込まれます。管理コンソールの保留キューにあるメッセージは、Human Task Manager の「**ランタイム**」タブで見ることができます。メッセージを再生する場合、これは E メールを初めて送信することと同じになります。

担当者割り当てのトラブルシューティング

以下の情報を使用して、許可のロールへの担当者の割り当てに関連した問題を解決します。

このタスクについて

この情報では、以下の問題を取り上げています。

- ユーザーがプロセス、スコープ、またはアクティビティーのインスタンスを管理できない
- 担当者ディレクトリー・プロバイダーのデプロイ中のエラー
- 担当者ディレク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
- 担当者ディレク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
- タスクまたはプロセス・インスタンスの予期しない担当者割り当て
- 停止したヒューマン・タスク
- 担当者割り当てに関連したエラー・メッセージと警告メッセージ
- 担当者割り当ての決定に関する追加メッセージを使用可能にする
- グループ作業項目および「グループ」担当者割り当て基準の問題
- 保管されている担当者割り当て結果のクリーンアップ
- 変更後の XSL 変換ファイルが適用されない

テクニカル・サポートの検索ページでも、追加情報を検索できます。

ユーザーがプロセス、スコープ、またはアクティビティーのインスタンスを管理またはモニターできず、管理タスクが作成されない

プロセス管理がシステム管理者に制限される場合、インスタンス・ベースの管理は使用不可になり、プロセス、スコープ、およびアクティビティーに対するすべての管理アクションは BPESystemAdministrator ロールのユーザーに制限されます。この管理モードについて詳しくは、58 ページの『代替プロセス管理許可モード』を参照してください。

Business Flow Manager が代替モードで稼働するように切り替えられた場合は、次のいずれかのアクションを実行しなければならない可能性があります。

- 管理アクションを実行するすべてのユーザーおよびプログラムが、適切なロールのユーザー ID を使用していることを確認します。例えば、BPESystemAdministrator や BPESystemmonitor などです。
- 代替プロセス管理許可モードをオフにすることにより、インスタンス・ベースの管理を復元します。

担当者ディレクトリー・プロバイダーのデプロイ中のエラー

Lightweight Directory Access Protocol (LDAP) 担当者ディレクトリー・プロバイダーを使用している場合は、プロバイダー構成パラメーターの値が正しくない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。

担当者ディレク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

担当者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 担当者ディレクトリー・プロバイダーで使用するサンプル XSL 変換ファイルは、`LDAPTransformation.xsl` です。 このファイルは、*`install-root`*`/ProcessChoreographer/Staff` ディレクトリーにあります。 デフォルトの Threshold 値は `1000000` であるため、デフォルトのままでは、しきい値に現実的な意味はありません。ただし、小さな値に変更する場合も、慎重に検討した上で変更してください。

1. 新規担当者ディレクトリー・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">1000000</xsl:variable>
```

担当者ディレク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・データベース内の担当者ディレクトリー (LDAP サーバーなど) に突き合わせて評価された担当者割り当ての結果をキャッシュに入れます。担当者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担当者照会結果の最新表示**。大幅な変更があり、ほとんどすべての担当者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担当者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場合や、担当者照会結果の全体またはサブセットをただちに最新表示する場合は、この方式を使用します。
- **更新デーモンを使用した担当者照会結果の最新表示**。期限切れのすべての担当者照会結果を定期的に自動で最新表示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 verb 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ることはありません。

タスクまたはプロセス・インスタンスの予期しない担当者割り当て

タスクの特定のロールに担当者割り当て基準を定義しない場合、または担当者割り当てが失敗するか、結果を戻さない場合、デフォルトの担当者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できます。 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 136. 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、担当者割り当てが失敗するか、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 137. 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、担当者割り当てが失敗するか、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
タスクの潜在的スターター (適用対象は呼び出しタスクのみ)	だれでも潜在的スターターになる	例外が発生し、プロセスは開始されない
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読者になります。
- 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読者になります。
- 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読者になります。
- エスカレーション受信側が、エスカレートしたタスクの読者になります。

表 138. 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、担当者割り当てが失敗するか、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	この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	例外がスローされ、タスクは作成されない
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
読者	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、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読者になります。
- 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読者になります。
- エスカレーション受信側が、エスカレートしたタスクの読者になります。

注: Business Flow Manager API を使用してメソッドが呼び出されると、BPESystemAdministrator ロールのメンバーは管理者権限を付与され、BPESystemMonitor ロールのメンバーは読者権限を付与されます。

注: Human Task Manager API を使用してメソッドが呼び出されると、TaskSystemAdministrator ロールのメンバーは管理者権限を付与され、TaskSystemMonitor ロールのメンバーは読者権限を付与されます。

停止したヒューマン・タスク

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...

これらの問題は、管理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. 管理セキュリティーが使用可能であ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、「**管理セキュリティーを使用可能にする**」 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。

担当者割り当てに関連したエラー・メッセージと警告メッセージ

担当者割り当て中に担当者ディレク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、使用する担当者ディレクトリー・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。担当者割り当て構成、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・モデルの担当者割り当て基準 (verb) 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・モデルの担当者割り当て基準 (verb) 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。担当者割り当て構成の XSLT ファイルにエラーがないかどうかも検査します。

担当者割り当ての決定に関する追加メッセージを使用可能にする

追加メッセージを SystemOut.log に記録するようにカスタム・プロパティーを設定できます。メッセージに記録されるイベントは以下のとおりです。

- 担当者解決でタスク・ロールのユーザーが見つからず、デフォルト・ユーザーが選択された。
- VMM を使用している場合に、指定されたエンティティーまたは特定の属性が VMM 担当者ディレクトリーで見つからないときに警告が出された。
- 代替を使用している場合に、ユーザーが置き換えられたかどうかの決定がログに記録された。

これらのメッセージは SystemOut.log のデータ量を著しく増加させる可能性があるので、これらの追加メッセージはテストまたはデバッグの目的でのみ使用可能に設定するようにしてください。

スタッフ診断機能を使用可能にするには、以下を実行します。

1. 管理コンソールを使用して、「**サーバー**」 → 「**クラスター**」 → 「**WebSphere Application Server クラスター**」 → 「***クラスター名***」または「**サーバー**」 → 「**サーバー・タイプ**」 → 「**WebSphere Application Server**」 → 「***サーバー名***」のいずれかをクリック後、「**構成**」タブの「**ビジネス・インテグレーション**」セクションで、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」をクリックします。
2. 「**構成**」タブで、カスタム・プロパティー `Staff.Diagnosis` の値を次のいずれかの値に設定します。

   off　追加の担当者割り当て情報は書き込まれません。

   on　常に追加の担当者割り当て情報が書き込まれます。

   development_mode
   サーバーが開発モードで稼働している場合のみ、追加の担当者割り当て情報が書き込まれます。これはデフォルト値です。デフォルトでは、WebSphere テスト環境は開発モードで稼働します。
3. サーバーを再始動します。

以下のメッセージが生成されます。

- `Core.StaffDiagMsgIsEnabled=CWTKE0057I: 担当者 (スタッフ) 解決診断メッセージの出力が使用可能になっています。` 診断機能が使用可能になっていることを示します。このメッセージは、Human Task Manager の開始時に生成されます。
- `Core.EverybodyIsPotInstanceCreator=CWTKE0047I: タスク {0} の潜在的インスタンス作成者が全員です。` 潜在的インスタンス作成者が定義されていないため、全員が潜在的インスタンス作成者になったことを示します。

- Core.OriginatorBecomesPotStarter=CWTKE0046I: オリジネーターがタスク {0} の潜在的開始者になります。 (スタンドアロン・タスクの場合のみ) 潜在的開始者が定義されていないため、オリジネーターが潜在的開始者になったことを示します。
- Core.EverybodyIsPotentialStarter=CWTKE0045I: タスク {0} の潜在的開始者が全員です。 (インライン・タスクの場合のみ) 潜在的開始者が定義されていないため、全員が潜在的開始者になったことを示します。
- Core.OriginatorBecomesAdministrator=CWTKE0044I: オリジネーターがタスク {0} の管理者になります。 管理者が定義されていないため、オリジネーターが管理者になったことを示します。
- Core.EscalationReceiverDoesNotExist=CWTKE0043W: 管理者は、エスカレーション {0} のエスカレーション受信側になります。 エスカレーション受信者のスタッフ解決は失敗したか、空のリストが返されたため、管理者がエスカレーション受信者になったことを示します。エスカレーション受信者が定義されていない場合、デフォルトは全員であり、トレース・メッセージが書き込まれます。
- Core.EverybodyIsPotentialOwner=CWTKE0014I: タスク {0} の潜在的所有者が Everybody です。 潜在的所有者が定義されていないため、全員が潜在的所有者になったことを示します。
- Core.PotentialOwnerDoesNotExist=CWTKE0015W: 管理者は、タスク {0} の潜在的な所有者となります。 潜在的所有者のスタッフ解決は失敗したか、空のリストが返されたため、管理者が潜在的所有者になったことを示します。潜在的所有者が定義されていない場合、デフォルトは全員であり、トレース・メッセージが書き込まれます。
- StaffPlugin.VMMEntityNotFound=CWWBS0457W: VMM エンティティーが見つかりませんでした。受け取った VMM メッセージは ''{0}'' です。 指定された VMM エンティティー (グループまたは個人) が担当者ディレクトリーで見つからなかったこと、およびその理由を示します。担当者ディレクトリーで見つからなかった担当者またはグループは、担当者解決結果には含まれません。
- StaffPlugin.VMMEntityAttributeNotFound=CWWBS0454W: VMM エンティティー ''{0}'' には、名前が ''{1}'' でタイプが ''{2}'' の属性はありません。 担当者ディレクトリーで VMM エンティティー (個人) を検索するときに、指定された属性が見つからなかったことを示します。ユーザーの E メール・アドレスが見つからない場合、ユーザーはエスカレーションの E メール通知を受信できません。ユーザーの preferredLanguage が見つからない場合は、デフォルトの言語設定が使用されます。読み取るときに代替属性 (isAbsent または substitutes) が見つからない場合は、属性の初期化が試行されます。書き込みまたは更新のときに代替属性が見つからない場合は、例外が生成されます。
- StaffPlugin.VMMResultIsEmpty=CWWBS0456W: VMM 起動によって、要求された結果エンティティーが戻されませんでした。 VMM の起動 (取得または検索) でエンティティーが戻されなかったことを示します。担当者解決結果にユーザーは含まれません。

グループ作業項目および「グループ」担当者割り当て基準の問題

グループ担当者割り当て基準を使用している場合は、以下の状況が発生する可能性があります。

- 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、「**許可検査で大/小文字を区別しない**」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 が提供している最新表示メソッドは、この担当者割り当て基準には適用されません。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。

保管されている担当者割り当て結果のクリーンアップ

担当者割り当て結果は、データベース内に保管されます。保管されているすべての担当者割り当て結果は、担当者割り当ての更新の対象になります。担当者割り当て結果を計算する元となるタスク・インスタンスを含むタスク・テンプレートが削除されると、保管されている担当者割り当て結果も削除されます。ただし、保管されている担当者割り当て結果を使用しているタスク・インスタンスのみが削除される場合は、保管されている担当者割り当て結果は削除されません。

データベースに保管されている不要な担当者割り当て結果が大量になるのを避けるには、タスク・テンプレートのコンテキストで以下のステップを実行します。

1. 担当者割り当て基準定義によって、共用または非共用の担当者割り当て結果が発生するかどうかを評価します。
2. 非共用の割り当て結果が発生した場合は、担当者割り当て結果にクリーンアップ手順を導入することを検討してください。クリーンアップ期間は、タスク・インスタンスの予想数、およびクリーンアップ期間ごとの非共用の担当者割り当て結果をベースにします。スクリプト・ベースのクリーンアップ手順を適用する方法については、『管理コマンドを使用した未使用の担当者割り当て照会結果の除去』を参照してください。

変更後の XSL 変換ファイルが適用されない

XSL 変換ファイルを変更した場合は、その変更を有効にするためにサーバーを再始動する必要があります。さらに、変更後の XSL ファイルが適用さ

れるのは、新しくデプロイしたプロセスとタスクに限られます。XSL ファイルの変更前にデプロイされていたプロセスやタスクには、その変更は適用されません。

Business Process Choreographer Explorer のトラブルシューティング

この情報を使用して、Business Process Choreographer Explorer に関連した問題を解決します。

このタスクについて

以下の情報を使用して、Business Process Choreographer Explorer へのアクセスまたは使用に関連した問題を解決します。

ブラウザーから Business Process Choreographer Explorer にアクセスしようとした際に発生したエラー

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、ログイン・ページの代わりにエラー・メッセージ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 の使用時のエラー・メッセージ

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**」をクリックします。

標準障害 `missingReply` (メッセージ CWWBE0071E) を伴うエラー・メッセージ `StandardFaultException`

`StandardFaultException` エラーを標準障害 `missingReply` (メッセージ CWWBE0071E) とともに受け取った場合、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、791 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 にログオンした場合に一部の項目が表示されない

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、または特定のアクショ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、WebSphere 管理セキュリティーが使用可能であることを確認する。
- 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。ユーザー ID に付与された許可に従って、管理ビューおよびオプションが表示されていないか、使用できません。
- 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。

「レポート」タブが表示されない場合は、システム管理者に連絡して、レポート作成機能を含む Business Process Choreographer Explorer が構成済みであることを確認してください。

エラー・メッセージ CWWBU0001E または HTMConnection 関数での通信エラー

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」を受け取った場合は、以下の情報を使用して問題の解決に役立てます。

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「例外ネーミング」でエラー・メッセージ WWBU0024E:「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」を受け取った場合は、以下の情報を使用して問題の解決に役立てます。

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。

Business Process Choreographer Explorer レポートのトラブルシューティング

Business Process Choreographer Explorer レポートで問題が発生した場合は、このトピックの情報を参照してください。

症状: テーブル作成オプションを使用したレポート・データベースのセットアップが失敗し、エラー・メッセージ CWWBO4013E が表示される

System.out に以下のメッセージが表示されます。

- CWWBO4015W: Business Process Choreographer Explorer レポート・データベース・スキーマが不完全です。$WAS_HOME/ProcessChoreographer/config/setupEventCollector のメニュー・オプション 6 を使用して、 JAR ファイルをインストールしてください。
- CWWBO4013E: bpcodbutil.jar ファイルが Derby ネットワーク・サーバーで見つかりませんでした。

理由

レポート・データベースのセットアップでは、 Derby 作業ディレクトリーを使用して UDF JAR ファイルをサーバーにインストールします。Derby ネットワーク・サーバーの作業ディレクトリーを正しく指定しないと、 JAR ファイルが見つかりません。

解決方法

以下の手順で、 networkServer サブディレクトリーから Derby ネットワーク・サーバーを始動します。

1. Derby ネットワーク・サーバーが稼働している場合は停止します。
2. コマンド行で、ディレクトリー $WAS_HOME/derby/bin/networkServer に移動します。
3. Derby ネットワーク・サーバーを再始動します (例えば startNetworkServer.bat を実行)。
4. Business Process Choreographer Explorer アプリケーションを再始動します。これで、テーブル作成が再開されます。

症状: Business Process Choreographer Explorer の「レポート」タブにイベントが表示されない

Business Process Choreographer Explorer のレポート・データベースにイベントが入っていません。あるいは、イベントがまだ変換されていません。この現象についてはさまざまな理由が考えられるので、以下のセクションでその理由と解決方法について述べます。

理由

イベントは正常に変換されているが、Business Process Choreographer Explorer のレポート・データベースに表示されない。

解決方法

イベントを受信したというメッセージのトレース項目と startTransform メッセージがトレース・ログに記録されているが、 Business Process Choreographer Explorer にイベントがまったく表示されない場合は、Business Process Choreographer Explorer と Event Collector が同じデータ・ソースを使用していることを確認してください。

1. 管理コンソールを使用して、「**アプリケーション**」 → 「**アプリケーション・タイプ**」 → 「**WebSphere エンタープライズ・アプリケーション**」をクリックした後、BPCExplorer アプリケーションを選択し、「**リソース参照**」をクリックします。

2. モジュールの「**ターゲット・リソース JNDI 名**」の値をメモします。通常、この値は jdbc/BPEDB です。
3. この操作を繰り返して、Event Collector アプリケーションの値を比較します。
4. 同一でない場合は、同一にします。

理由

サーバーで CEI サービスが使用可能になっていないため、イベントが受信されない。

解決方法

管理コンソールで「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「**サーバー名**」をクリックし、次に「**ビジネス・インテグレーション**」セクションで「**Common Event Infrastructure**」を展開して、「**Common Event Infrastructure の宛先**」をクリックし、「**サーバー起動時にサービスを使用可能にする**」チェック・ボックスが選択されていることを確認してください。

理由

ビジネス・プロセス・コンテナーで CEI ロギングが使用可能になっていない。

解決方法

ビジネス・プロセス・コンテナーで CEI ロギングが使用可能になっていることを確認します。CEI ロギングを使用可能にする方法については、301 ページの『Business Process Choreographer のロギング可能化』を参照してください。

理由

Common Event Infrastructure イベント・サーバーまたは Business Process Choreographer Event Collector が稼働していない。

解決方法

管理コンソールを使用して、Common Event Infrastructure イベント・サーバーと Business Process Choreographer Event Collector が稼働していることを確認してください。

理由

ビジネス・プロセスのイベント・モニターが使用不可になっている。

解決方法

WebSphere Integration Developer のプロセス・モデルの定義で、イベント・モニターが使用可能になっていることを確認してください。ビジネス・プロセスのイベント・モニターを使用可能にするための推奨方法については、WebSphere Integration Developer インフォメーション・センターを参照してください。

理由

イベント変換プログラムが起動されていない。

解決方法

Business Process Choreographer Explorer レポート作成機能の構成パラメーター変更に関する文書の説明に従って、Event Collector のしきい値の設定を低くします。その後、新規イベントを作成すると、Event Collector が起動します。

理由

イベントが生成され、CBE ブラウザーに表示されるが、イベント・サーバーでイベント配布が使用不可になっているため、Business Process Choreographer Explorer のレポート・データベースにはイベントが表示されない。

解決方法

管理コンソールで「**サービス統合**」 → 「**Common Event Infrastructure**」 → 「**イベント・サービス**」 → 「**イベント・サービス**」 → 「**デフォルトの Common Event Infrastructure イベント・サーバー (Default Common Event Infrastructure event server)**」をクリックし、「**イベント配布の使用可能化**」が選択されていることを確認してください。

理由

Business Process Choreographer Event Collector の構成設定が不適切なため、Business Process Choreographer Explorer の レポート・データベースでデータを表示できない。

解決方法

setupEventCollector 構成スクリプトを呼び出し、Business Process Choreographer Event Collector の構成設定 BPCEventTransformerEventCount、BPCEventTransformerMaxWaitTime、および BPCEventTransformerToleranceTime を変更してください。 Business Process Choreographer Event Collector の構成設定の変更について詳しくは、Business Process Choreographer Explorer レポート作成機能の、構成パラメーターの変更に関する文書を参照してください。

理由

BFMEvents イベント・グループを定義する必要がある。

解決方法

管理コンソールで「**サービス統合**」 → 「**Common Event Infrastructure**」 → 「**イベント・サービス**」 → 「**イベント・サービス**」 → 「**デフォルトの Common Event Infrastructure イベント・サーバー (Default Common Event Infrastructure event server)**」 → 「**イベント・グループ**」をクリックした後、グループ BFMEvents が存在するかどうかを確認してください。

- グループが存在しない場合は、Event Collector アプリケーションを再インストールします。
- イベント・グループが存在する場合は、セレクター・ストリングを確認します。通常は、 `CommonBaseEvent[starts-with(@extensionName,'BPC.BFM.')]` というストリングに設定されています。

症状: 表示イベントの数が予想より少ない

Business Process Choreographer Explorer のレポート・データベースにイベントが入っていません。あるいは、イベントがまだ変換されていません。この現象についてはさまざまな理由が考えられるので、以下のセクションでその理由と解決方法について述べます。

理由

発行イベントがサポートされていない。これは、トレース機能で検証できます。`com.ibm.bpe.observer.*` のトレースを使用可能にしてください。トレースで、次のようなメッセージを検索します。「イベント・コード *eventCode* は Observer ではサポートされていません。(Event Code *eventCode* is not relevant for Observer.)イベントを破棄しています。」 このようなメッセージが見つかった場合、当該イベントは Event Collector から無視されています。

解決方法

発行されるイベントがサポートされていることを確認してください。サポートされていないイベントは認識されません。

理由

関連付けができないため、イベントが消去されている。プロセス開始イベントは、どのような場合も必ず発行してください。発行しないと、アクティビティーが起動するイベントは消去されます。

前のイベントがないためにイベントが消去されているかどうかを確認するには、メッセージ「CWWBO0014I: PIID が '*nnnnnn*' であるプロセス・インスタンスのプロセス開始イベントが見つかりませんでした。イベントを破棄しています。」があるか調べます。 *nnnnnn* はプロセス・インスタンスの ID です。

問題が解決しない場合

- サーバーのシステム・ログ・ファイル SystemOut.log でエラー・メッセージがないかどうかを確認してください。
- Business Process Choreographer Event Collector および Business Process Choreographer Explorer のデプロイメントと構成を確認してください。構成設定を確認するには、管理コンソールか clientconfig.jacl 構成スクリプトを使用します。Business Process Choreographer Event Collector の構成設定の変更方法について詳しくは、Business Process Choreographer Explorer レポート作成機能の構成パラメーターの変更に関する文書を参照してください。
- 管理コンソールで「**トラブルシューティング**」 → 「**ログおよびトレース**」 → 「**サーバー名**」 → 「**診断トレース・サービス**」 → 「**ログ詳細レベルの変更**」をクリックして、レポート作成のトレース機能を使用可能にします。`com.ibm.bpe.observer.*` の詳細レベルを `all` に設定し、BPCECollector アプリケーションと BPCExplorer アプリケーションを再始動します。

第 8 部 付録

付録. Business Process Choreographer のデータベース・ビュー

この参照情報では、事前定義データベース・ビューの列について説明します。

ACTIVITY ビュー

この定義済み Business Process Choreographer 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 139. 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。
SIID	ID	スコープ・インスタンス ID。
STID	ID	テンプレートのスコープの ID。
EHIID	ID	このアクティビティーがイベント・ハンドラーの一部である場合はイベント・ハンドラー・インスタンスの ID。
ENCLOSING_FEIID	ID	このアクティビティーが別の forEach アクティビティーにネストされている場合、囲む側の forEach アクティビティーの ID。

表 139. 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) KIND_REPEAT_UNTIL (52)
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アクティブ化された時刻。
FIRST_ACTIVATED	タイム・スタンプ	そのアクティビティーが初めてアクティブ化された時刻。
STARTED	タイム・スタンプ	アクティビティーの開始時刻。
PREVIOUS_EXPIRATION_TIME	タイム・スタンプ	前のアクティビティー有効期限時刻。

表 139. 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_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) STATE_PROCESSING_UNDO (14)
SUBSTATE	整数	プロセス・インスタンスがマイグレーションされたときのアクティビティーの副状態。指定可能な値は、以下のとおりです。 SUB_STATE_NONE (0) SUB_STATE_EXPIRING (1) SUB_STATE_SKIPPING (2) SUB_STATE_RESTARTING (3) SUB_STATE_FINISHING (4) SUB_STATE_FAILING (5)
STOP_REASON	整数	アクティビティーが停止した理由。指定可能な値は、以下のとおりです。 STOP_REASON_UNSPECIFIED (1) STOP_REASON_ACTIVATION _FAILED (2) STOP_REASON_IMPLEMENTATION _FAILED (3) STOP_REASON_FOLLOW_ON _NAVIGATION_FAILED (4) STOP_REASON_EXIT_CONDITION _FALSE (5)
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。

表 139. ACTIVITY ビュー内の列 (続き)

列名	タイプ	コメント
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。
INVOKED_INST_ID	整数	呼び出されたプロセスまたはタスクのインスタンス ID。インスタンスのタイプは、INVOKED_INSTANCE_TYPE 列の値で確認できます。
INVOKED_INST_TYPE	整数	INVOKED_INST_ID 列にあるインスタンス ID のタイプ。指定可能な値は、以下のとおりです。 INVOKED_INSTANCE_TYPE_ NOT_SET (0) INVOKED_INSTANCE_TYPE_ INLINE_TASK (1) INVOKED_INSTANCE_TYPE _CHILD_TASK (2) INVOKED_INSTANCE_TYPE _CHILD_PROCESS (3)
SKIP_REQUESTED	ブール	アクティビティーにスキップのマークを付けるかどうかを指定します。
CONTINUE_ON_ERROR	ブール	予期しない障害が発生し、その障害に対して障害ハンドラーが定義されていない場合に、プロセスに対して行う処理を指定します。この列は、アクティビティー・テンプレートの対応する値で初期化されますが、forceComplete および forceRetry という API で上書きできます。 指定可能な値は、以下のとおりです。 **True** 標準の障害処理が適用されます。 **False** プロセスのナビゲーションが停止し、プロセスの修復が可能になります。

ACTIVITY_ATTRIBUTE ビュー

この定義済み Business Process Choreographer 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 140. 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。
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・タイプ。

ACTIVITY_SERVICE ビュー

この定義済み Business Process Choreographer 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 141. 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・インスタンスの 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	名前空間の 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 Business Process Choreographer 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 142. APPLICATION_COMP ビュー内の列

列名	タイプ	コメント
ACOID	ID	アプリケーション・コンポーネントの ID。

表 142. APPLICATION_COMP ビュー内の列 (続き)

列名	タイプ	コメント
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。
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。

AUDIT_LOG_B ビュー

この定義済み Business Process Choreographer データベース・ビューは、ビジネス・プロセスの監査ログ情報の照会に使用します。

インライン・タスクは、AUDIT_LOG_B ビューと TASK_AUDIT_LOG ビューの両方に記録されます。例えば、インライン参加タスクを要求すると TASK_CLAIMED イベントおよび ACTIVITY_CLAIMED イベントが生成されます。他のすべてのタス

ク・タイプは TASK_AUDIT_LOG ビューにのみ記録されます。WebSphere Integration Developer での選択内容に応じて、インライン・ヒューマン・タスクは、Business Flow Manager スタッフ・アクティビティー・イベントおよび Human Task Manager タスク・イベント用に Common Event Infrastructure (CEI) イベントが発行されるようにトリガーすることもできます。

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)

以下の表で、AUDIT_LOG_B ビューについて説明します。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。

表 143. 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)。
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。

表 143. 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_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 KIND_REPEAT_UNTIL 52 これらは、ActivityInstanceData.KIND_* で定義される定数です。
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。

表 143. 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
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。
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

表 143. 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
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・タイプ名前空間 '}' 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。 **ACTIVITY_CHILD_ PROCESS_TERMINATING** プロセス・インスタンスがマイグレーションされたときのアクティビティーの副状態。指定可能な値は、以下のとおりです。 SUB_STATE_NONE (0) SUB_STATE_EXPIRING (1) SUB_STATE_SKIPPING (2) SUB_STATE_RESTARTING (3) SUB_STATE_FINISHING (4) SUB_STATE_FAILING (5)

ESCALATION ビュー

この定義済み Business Process Choreographer 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 144. ESCALATION ビュー内の列

列名	タイプ	コメント
ESIID	ID	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。
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アクティブ化される時刻。

表 144. ESCALATION ビュー内の列 (続き)

列名	タイプ	コメント
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。
ESCALATION_TIME	タイム・スタンプ	エスカレーションが発生する時刻。
ESTID	ID	対応するエスカレーション・テンプレートの ID。
FIRST_ESIID	ID	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)** タスクの優先度は一度に 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。
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	ID	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 Business Process Choreographer 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 145. ESCALATION_CPROP ビュー内の列

列名	タイプ	コメント
ESIID	ID	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 Business Process Choreographer 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 146. ESCALATION_DESC ビュー内の列

列名	タイプ	コメント
ESIID	ID	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。

ESC_TEMPL 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・テンプレートのデータを照会するために使用します。

表 147. ESC_TEMPL ビュー内の列

列名	タイプ	コメント
ESTID	ID	エスカレーション・テンプレートの 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 147. ESC_TEMPL 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。
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。
CONTAINMENT_ CTX_ID	ストリング	エスカレーション・テンプレートがインライン・タスク・テンプレートに属する場合、包含コンテキストはプロセス・テンプレートです。エスカレーション・テンプレート・コンテキストがスタンドアロン・タスク・テンプレートに属する場合、包含コンテキストはタスク・テンプレートです。
FIRST_ESTID	ID	エスカレーション・テンプレート・チェーンの 1 番目のエスカレーション・テンプレートの ID です。

表 147. ESC_TEMPL ビュー内の列 (続き)

列名	タイプ	コメント
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)** タスクの優先度は一度に 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・テンプレートの名前。
PREVIOUS_ESTID	ID	エスカレーション・テンプレート・チェーンの直前のエスカレーション・テンプレートの ID です。
TKTID	ID	エスカレーション・テンプレートが所属するタスク・テンプレートの ID。

ESC_TEMPL_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・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 148. ESC_TEMPL_CPROP ビュー内の列

列名	タイプ	コメント
ESTID	ID	エスカレーション・テンプレートの ID。
NAME	ストリング	プロパティーの名前。
TKTID	ID	エスカレーション・テンプレートが所属するタスク・テンプレートの ID。
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。
VALUE	ストリング	String 型のカスタム・プロパティーの値。

ESC_TEMPL_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・テンプレー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 149. ESC_TEMPL_DESC ビュー内の列

列名	タイプ	コメント
ESTID	ID	エスカレーション・テンプレートの 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。
TKTID	ID	エスカレーション・テンプレートが所属するタスク・テンプレートの ID。
DESCRIPTION	ストリング	タスク・テンプレートの説明。

表 149. ESC_TEMPL_DESC ビュー内の列 (続き)

列名	タイプ	コメント
DISPLAY_NAME	ストリング	エスカレーションの記述名。

MIGRATION_FRONT ビュー

この定義済み Business Process Choreographer データベース・ビューは、プロセス・インスタンスがプロセス・テンプレートの新規バージョンにマイグレーションされたときに、プロセス・インスタンスがそのナビゲーション内のどの場所にあったかを照会するために使用します。マイグレーション・フロントは、マイグレーションが実行されたときにアクティブであるか、あるいは完了したばかりのアクティビティーを表します。

表 150. MIGRATION_FRONT 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。
SOURCE_PTID	ID	マイグレーション前にプロセス・インスタンスに関連付けられているプロセス・テンプレートの ID。
TARGET_PTID	ID	マイグレーション後にプロセス・インスタンスに関連付けられているプロセス・テンプレートの ID。
AIID	ID	プロセスがマイグレーションされたときにプロセス・ナビゲーション・フロントの一部であったアクティビティーの ID。マイグレーション・フロントは、プロセスがマイグレーションされたときに各並列分岐でプロセス・ナビゲーションが到達した最後のアクティビティーから成っています。
SOURCE_ATID	ID	マイグレーション前のアクティビティー・テンプレートの ID。
TARGET_ATID	ID	マイグレーション後のアクティビティー・テンプレートの ID。プロセス・インスタンスがマイグレーションされる前にアクティビティーが完了した場合、アクティビティー・テンプレート ID は変更されません。
MIGRATION_TIME	タイム・スタンプ	プロセス・インスタンスがマイグレーションされた時刻。

表 150. MIGRATION_FRONT ビュー内の列 (続き)

列名	タイプ	コメント
STATE	整数	プロセス・インスタンスがマイグレーションされたときの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_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)
SUBSTATE	整数	プロセス・インスタンスがマイグレーションされたときのアクティビティーの副状態。指定可能な値は、以下のとおりです。 SUB_STATE_NONE (0) SUB_STATE_EXPIRING (1) SUB_STATE_SKIPPING (2) SUB_STATE_RESTARTING (3) SUB_STATE_FINISHING (4) SUB_STATE_FAILING (5)
STOP_REASON	整数	アクティビティーが停止した理由。指定可能な値は、以下のとおりです。 STOP_REASON_UNSPECIFIED (1) STOP_REASON_ACTIVATION _FAILED (2) STOP_REASON_IMPLEMENTATION _FAILED (3) STOP_REASON_FOLLOW_ON _NAVIGATION_FAILED (4) STOP_REASON_EXIT_CONDITION _FALSE (5)

PROCESS_ATTRIBUTE ビュー

この定義済み Business Process Choreographer 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 151. 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。
NAME	ストリング	カスタム・プロパティーの名前。

表 151. PROCESS_ATTRIBUTE ビュー内の列 (続き)

列名	タイプ	コメント
VALUE	ストリング	カスタム・プロパティーの値。
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・タイプ。

PROCESS_INSTANCE ビュー

この定義済み Business Process Choreographer 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 152. 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_PIID	ID	親プロセス・インスタンスの ID。
PARENT_NAME	ストリング	親プロセス・インスタンスの名前。
TOP_LEVEL_PIID	ID	トップレベル・プロセス・インスタンスのプロセス・インスタンス ID。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・インスタンスのプロセス・インスタンス ID になります。

表 152. PROCESS_INSTANCE ビュー内の列 (続き)

列名	タイプ	コメント
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。
TEMPLATE_DESCR	ストリング	関連するプロセス・テンプレートの説明。
RESUMES	タイム・スタンプ	プロセス・インスタンスが自動的に再開される時刻。
CONTINUE_ON_ERROR	ブール	予期しない障害が発生し、その障害に対して障害ハンドラーが定義されていない場合に、プロセスに対して行う処理を指定します。指定可能な値は、以下のとおりです。 **True** 標準の障害処理が適用されます。 **False** プロセスのナビゲーションが停止し、プロセスの修復が可能になります。
IS_MIGRATED	ブール	プロセス・インスタンスがプロセスの旧バージョンからマイグレーションされたかどうかを指定します。この属性が TRUE に設定されている場合、プロセス・インスタンスはマイグレーションされています。

PROCESS_TEMPLATE ビュー

この定義済み Business Process Choreographer 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 153. 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名前空間。
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。

表 153. PROCESS_TEMPLATE ビュー内の列 (続き)

列名	タイプ	コメント
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_SUPPORTS (4)
DISPLAY_NAME	ストリング	プロセスの記述名。
CAN_RUN_SYNC	ブール	call メソッドでプロセスを呼び出すことを可能にするかどうかを指定します。
CAN_RUN_INTERRUP	ブール	initiate メソッドまたは sendMessage メソッドでプロセスを呼び出すことを可能にするかどうかを指定します。
CONTINUE_ON_ERROR	ブール	予期しない障害が発生し、その障害に対して障害ハンドラーが定義されていない場合に、プロセスに対して行う処理を指定します。指定可能な値は、以下のとおりです。 **True** 標準の障害処理が適用されます。 **False** プロセスのナビゲーションが停止し、プロセスの修復が可能になります。

PROCESS_TEMPL_ATTR ビュー

この定義済み Business Process Choreographer データベース・ビューは、プロセス・テンプレートのカスタム・プロパティーの照会に使用します。

表 154. PROCESS_TEMPL_ATTR ビュー内の列

列名	タイプ	コメント
PTID	ID	カスタム・プロパティーを所有するプロセス・テンプレートの ID。
NAME	ストリング	カスタム・プロパティーの名前。

表 154. PROCESS_TEMPL_ATTR ビュー内の列 (続き)

列名	タイプ	コメント
VALUE	ストリング	カスタム・プロパティーの値。

QUERY_PROPERTY ビュー

この定義済み Business Process Choreographer 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 155. 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名前空間。
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。 プロパティー・タイプがブールの場合、値は `0` (false)、または `1` (true) にマップされます。
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・タイプが日付、時刻、またはタイム・スタンプの場合、値はタイム・スタンプ型にマップされ、この列にはタイム・スタンプの値が表示されます。

TASK_AUDIT_LOG ビュー

この定義済み Business Process Choreographer データベース・ビューは、ヒューマン・タスクの監査ログ情報の照会に使用します。

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。WebSphere Integration Developer での選択内容に応じて、インライン・ヒューマン・タスクは、Business Flow Manager スタッフ・アクティビティー・イベントおよび Human Task Manager タスク・イベント用に Common Event Infrastructure (CEI) イベントが発行されるようにトリガーすることもできます。

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)

以下の表で、TASK_AUDIT_LOG ビューについて説明します。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。

表 156. TASK_AUDIT_LOG 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。
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)。
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・タイプの名前空間。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

表 156. TASK_AUDIT_LOG ビューの構造 (続き)

名前	TIE	TTE	EIE	説明
FAULT_TYPE_NAME	x			障害メッセージ・タイプのローカル名。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・インスタンスの名前空間。
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名前空間。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。
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。

表 156. TASK_AUDIT_LOG ビューの構造 (続き)

名前	TIE	TTE	EIE	説明
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。 STATE_INACTIVE 1 STATE_READY 2 STATE_RUNNING 3 STATE_FINISHED 5 STATE_FAILED 6 STATE_TERMINATED 7 STATE_CLAIMED 8 STATE_EXPIRED 12 FORWARDED 101
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 156. TASK_AUDIT_LOG 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。

TASK ビュー

この定義済み Business Process Choreographer 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 157. 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	タスクがアクティブ化された時刻。
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。
ASSIGNMENT_TYPE	整数	タスクの作業の割り当て方法を指定します。指定可能な値は、以下のとおりです。 **ASSIGNMENT_TYPE_SINGLE** タスクは 1 人のユーザーにのみ割り当てられます。 **ASSIGNMENT_TYPE_PARALLEL** タスクは、そのタスクで同時に作業する複数のユーザーに割り当てられます。

表 157. 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読者権限が必要です。
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	タスクが初めてアクティブ化された時刻。
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。

表 157. TASK ビュー内の列 (続き)

列名	タイプ	コメント
INHERITED_AUTH	整数	サブタスクが親タスクから継承する許可の種類。指定可能な値は、以下のとおりです。 **INHERITED_AUTH_NONE (0)** サブタスクは親タスクの許可のロールを継承しません。 **INHERITED_AUTH_ADMINISTRATOR (1)** サブタスクは親タスクの管理者を継承します。 **INHERITED_AUTH_ALL (3)** サブタスクは、親タスクに定義されているすべての許可のロールを継承します。
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_CHILD	ブール	このタスクがビジネス・プロセスの子であるかどうかを示します。
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。
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_READ	ブール	いずれかのユーザーがタスクを読み取ったことを示します。
IS_TRANSFERRED_TO_WORK_BASKET	ブール	WORK_BASKET で指定された名前のワーク・バスケットにタスクが転送されたことを示します。
INVOKED_INSTANCE_ID	ID	呼び出されたサービス (通常はタスク、プロセス、またはアクティビティー) のインスタンス ID。

表 157. TASK ビュー内の列 (続き)

列名	タイプ	コメント
INVOKED_ INSTANCE_TYPE	整数	指定可能な値は、以下のとおりです。 **INVOKED_INSTANCE_TYPE_NOT_SET (0)** タスクがサービスを呼び出さなかったことを示します。 **INVOKED_INSTANCE_TYPE_PROCESS (1)** タスクがビジネス・プロセスを呼び出したことを示します。 **INVOKED_INSTANCE_TYPE_ACTIVITY (2)** タスクがビジネス・プロセス内のアクティビティーを呼び出したことを示します。 **INVOKED_INSTANCE_TYPE_TASK (3)** タスクがスタンドアロン・タスクを呼び出したことを示します。 **INVOKED_INSTANCE_TYPE_EVENT (4)** タスクがイベント・ハンドラーまたはアクティビティー内のアクティビティーを呼び出したことを示します。アクティビティーはまだイベントを受信する準備ができていません。
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、処理されるコラボレーション・タスク であることを示します。 **KIND_ORIGINATING (103)** タスクが人からコンピューターへの対話をサポートし、人がサービスを作成、開始、および始動できる*呼び出し*タスク であることを示します。 **KIND_PARTICIPATING (105)** タスクがコンピューターから人への対話をサポートし、人がサービスを実装できる予定タスク であることを示します。 **KIND_ADMINISTRATIVE (106)** タスクが管理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。

表 157. TASK ビュー内の列 (続き)

列名	タイプ	コメント
NAME_SPACE	ストリング	タスクのカテゴリー化に使用される名前空間。
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。
RESUMES	タイム・スタンプ	タスクが自動的に再開される時刻。
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。
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。

表 157. TASK ビュー内の列 (続き)

列名	タイプ	コメント
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。
WORK_BASKET	ストリング	このタスクが属するワーク・バスケットの名前。

TASK_CPROP ビュー

この定義済み Business Process Choreographer 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 158. TASK_CPROP 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 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_DESC ビュー

この定義済み Business Process Choreographer 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 159. TASK_DESC 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 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_HISTORY ビュー

この定義済み Business Process Choreographer データベース・ビューは、タスクのイベント・ログの照会に使用します。

表 160. TASK_HISTORY ビュー内の列

列名	タイプ	コメント
EVENT	整数	イベント・タイプ。
TKIID	ID	タスク・インスタンスの ID。
EVENT_TIME	タイム・スタンプ	ログに記録されたイベントが発生した時刻。
PRINCIPAL	ストリング	イベントをトリガーしたプリンシパルの名前。
FROM_ID	ストリング	TO_ID に転送された作業項目を持っていたユーザー、または作業項目が削除されたユーザーの名前。すべてのイベントが FROM_ID 値を持つわけではありません。
TO_ID	ストリング	作業項目が作成されたユーザーまたは作業項目の転送先ユーザーの名前。すべてのイベントが TO_ID 値を持つわけではありません。
WORK_ITEM_KIND	整数	許可タイプ。指定可能な値は、以下のとおりです。 WORK_ITEM_KIND_EVERYBODY (1) WORK_ITEM_KIND_USER (2) WORK_ITEM_KIND_GROUP (3)
REASON	整数	作業項目の割り当て理由。すべてのイベントが REASON 値を持つわけではありません。指定可能な値は、以下のとおりです。 REASON_NONE (0) REASON_POTENTIAL_OWNER (1) REASON_EDITOR (2) REASON_READER (3) REASON_OWNER (4) REASON_POTENTIAL_STARTER (5) REASON_STARTER (6) REASON_ADMINISTRATOR (7) REASON_ORIGINATOR (9) REASON_ESCALATION_RECEIVER (10) REASON_POTENTIAL_INSTANCE_ CREATOR (11)
PARENT_TKIID	ID	関連タスク・インスタンスの ID。タスク・インスタンスが親タスクのサブタスクまたは後続タスクである場合、この列には、親タスクの ID が入ります。それ以外の場合、この列はヌルです。
ESIID	ID	イベントがエスカレーション・インスタンスに関連付けられている場合、この列には、エスカレーション・インスタンス ID が入ります。それ以外の場合、この列はヌルです。

表 160. TASK_HISTORY ビュー内の列 (続き)

列名	タイプ	コメント
NEXT_TIME	タイム・スタンプ	イベントが発生する時刻。時刻はイベント・タイプによって異なります。 **TASK_STARTED** タスク・インスタンスの有効期限時刻。 **TASK_SUSPENDED** タスク・インスタンスの再開時刻。 **TASK_COMPLETED、 TASK_TERMINATED、 TASK_EXIRED、および TASK_FAILED** タスク・インスタンスが自動的に削除される時刻。 **ESCALATION_STARTED** エスカレーションが起動される時刻。 **ESCALATION_FIRED** エスカレーションが再度起動される時刻。

TASK_TEMPL ビュー

この定義済み Business Process Choreographer 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 161. TASK_TEMPL ビュー内の列

列名	タイプ	コメント
TKTID	ID	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。
ASSIGNMENT_TYPE	整数	タスクの作業の割り当て方法を指定します。指定可能な値は、以下のとおりです。 **ASSIGNMENT_TYPE_SINGLE** タスクは 1 人のユーザーにのみ割り当てられます。 **ASSIGNMENT_TYPE_PARALLEL** タスクは、そのタスクで同時に作業する複数のユーザーに割り当てられます。

表 161. TASK_TEMPL ビュー内の列 (続き)

列名	タイプ	コメント
AUTONOMY	整数	タスク・インスタンスと親プロセスのリレーションシップを指定します。指定可能な値は、以下のとおりです。 **AUTONOMY_PEER (1)** タスク・インスタンスはその親プロセスからは独立して実行されます。 **AUTONOMY_CHILD (2)** タスク・インスタンスの実行は、親プロセスに依存します。 **AUTONOMY_NOT_APPLICABLE (3)** タスク・インスタンスはインライン・タスクであるため、autonomy 属性は適用されません。
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。
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読者権限が必要です。
DEFINITION_NAME	ストリング	Task Execution Language (TEL) ファイルのタスク・テンプレート定義の名前。
DEFINITION_NS	ストリング	TEL ファイルのタスク・テンプレート定義の名前空間。

表 161. TASK_TEMPL ビュー内の列 (続き)

列名	タイプ	コメント
INHERITED_AUTH	整数	サブタスクが親タスクから継承する許可の種類。指定可能な値は、以下のとおりです。 **INHERITED_AUTH_NONE (0)** サブタスクは親タスクの許可のロールを継承しません。 **INHERITED_AUTH_ADMINISTRATOR (1)** サブタスクは親タスクの管理者を継承します。 **INHERITED_AUTH_ALL (3)** サブタスクは、親タスクに定義されているすべての許可のロールを継承します。
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、処理されるコラボレーション・タスク であることを示します。 **KIND_ORIGINATING (103)** タスクが人からコンピューターへの対話をサポートし、人がサービスを作成、開始、および始動できる*呼び出しタスク* であることを示します。 **KIND_PARTICIPATING (105)** タスクがコンピューターから人への対話をサポートし、人がサービスを実装できる予定タスク であることを示します。 **KIND_ADMINISTRATIVE (106)** タスクが管理タスクであることを示します。
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名前空間。
PRIORITY	整数	タスク・テンプレートの優先順位。

表 161. 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。
WORK_BASKET	ストリング	このテンプレートのインスタンスを含むワーク・バスケットの名前。

TASK_TEMPL_CPROP ビュー

この定義済み Business Process Choreographer 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 162. TASK_TEMPL_CPROP ビュー内の列

列名	タイプ	コメント
TKTID	ID	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 Business Process Choreographer 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 163. TASK_TEMPL_DESC ビュー内の列

列名	タイプ	コメント
TKTID	ID	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 Business Process Choreographer 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 164. 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 164. 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_TEMPLATE (2)** その作業項目がプロセス・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_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_ESCALATION_TEMPLATE (8)** その作業項目がエスカレーション・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_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。

表 164. WORK_ITEM ビュー内の列 (続き)

列名	タイプ	コメント
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。
REASON	整数	作業項目の割り当て理由。この整数値は、以下のいずれかを意味します。 REASON_NONE (0) REASON_POTENTIAL_OWNER (1) REASON_EDITOR (2) REASON_READER (3) REASON_OWNER (4) REASON_POTENTIAL_STARTER (5) REASON_STARTER (6) REASON_ADMINISTRATOR (7) REASON_ORIGINATOR (9) REASON_ESCALATION_RECEIVER (10) REASON_POTENTIAL_INSTANCE_ CREATOR (11)
CREATION_TIME	タイム・スタンプ	作業項目が作成された日時。

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21