WebSphere Process Server for Multiplatforms

バージョン 6.1.0

Business Process Choreographer

WebSphere Process Server for Multiplatforms

バージョン 6.1.0

Business Process Choreographer

お願い

本書および本書で紹介する製品をご使用になる前に、特記事項に記載されている情報をお読みください。

本書は、WebSphere Process Server for Multiplatforms (製品番号 5724-L01) バージョン 6、リリース 1、モディフィケーション 0、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

本書についてのご意見は、doc-comments@us.ibm.com へ E メールでお寄せください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

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 Multiplatforms
Version 6.1.0
Business Process Choreographer

発行： 日本アイ・ビー・エム株式会社

担当： ナショナル・ランゲージ・サポート

第1刷 2008.4

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・プロセスについて

ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、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・モデルのバージョン管理のための有効開始タイム・スタンプ。
- 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。

関連情報

Business Process Execution Language for Web Services バージョン 1.1

OASIS Web Services Business Process Execution Language バージョン 2.0

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、プロセス・テンプレートのプロパティーは PROCESS_TEMPLATE データベース・ビューに格納されます。

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 ビジネス・プロセス・アプリケーションをアンインストールするには、その前にプロセス・テンプレートを停止する必要があります。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。

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、中断不可能なビジネス・プロセスと呼ばれる場合もあります。Microflow は、それぞれ異なるトランザクション機能を持つことができます。Microflow は、グローバル・トランザクション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。
- 中断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

19 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。

プロセスのバージョン管理

ビジネス・プロセスの新規バージョンを作成することができるため、同じプロセス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ことができます。

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。プロセスのバージョンは、その有効開始日によって決まります。つまり、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。実行時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、開始時に関連付けられたプロセス・テンプレートを使って、継続して実行されます。これにより、プロセス・テンプレートは次のカテゴリーに分けられます。

- 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
- もはや有効ではないが、長期実行している既存のプロセス・インスタンスにはまだ使用されるプロセス・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

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。

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常に、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ものではありません。

関連概念

20 ページの『プロセスの実行時バインディング』
このシナリオでは、ビジネス・プロセスがクライアントまたは呼び出されるサービスとして使用されると想定しています。これは、validFrom プロセス・プロパティーによって決定される使用中プロセス・バージョンに関する実行時バインディングを適用します。

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・テンプレートをステートフルに具現化したものです。

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してそのインスタンスが完了するか、強制終了アクティビティーが実行されるか、またはそのプロセスで処理されない障害がそのインスタンスに発生します。

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。これらのプロ

パティーは、いずれも PROCESS_INSTANCE データベース・ビューに格納されます。プロセス・インスタンスでのアクティビティーのプロパティーは、ACTIVITY データベース・ビューに格納されます。

プロセス・ライフ・サイクル

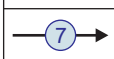
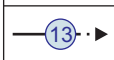
プロセスが開始されるとビジネス・プロセス・インスタンスの処理が開始され、そのビジネス・プロセス・インスタンスはその環境との対話を開始します。つまり、特定の対話は特定のプロセス状態でのみ可能であり、これらの対話が今度はプロセス・インスタンスの状態に影響を与えます。

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。

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。

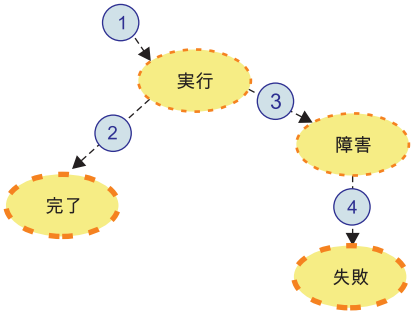
シンボル	説明
	過渡状態。このような状態は表示されません。
	永続状態。
	過渡の終了状態。
	永続の終了状態。
	Business Flow Manager によって自動的にトリガーされる状態遷移。
	API を使用した外部対話の結果である状態遷移。
	Business Flow Manager によって制御される状態遷移、または API を使用した外部対話の結果である状態遷移。

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。

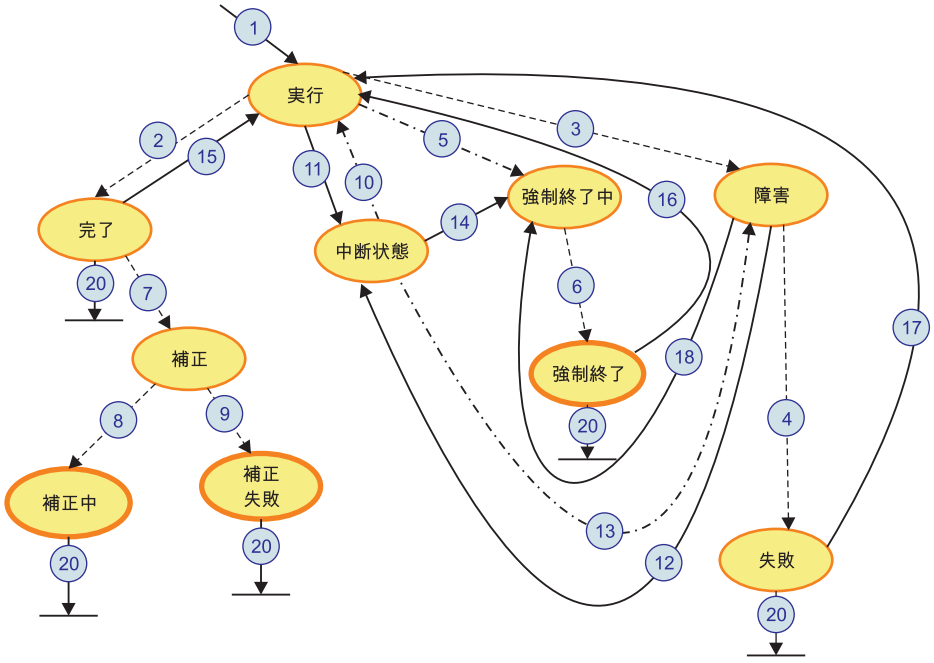


プロセス・インスタンスの正常な開始後、プロセス・インスタンスが到達する最初のプロセス状態は実行状態です (1)。プロセス・インスタンスが通常どおり最後まで実行されて完了すると、プロセス状態は実行から完了に変化します (2)。障害がプロセス境界に到達すると、プロセスは障害状態になります (3)。障害ハンドラーの実行中は、プロセスは障害状態のままです。このあと、プロセス・インスタンスは失敗状態になります (4)。

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、完了状態、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 (20)。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。

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開始されたプロセスをサブプロセス といいます。サブプロセスのライフ・サイクルの管理方法は、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。

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、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。

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。

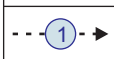
- short-lived アクティビティー (assign、empty、reply、rethrow、throw、および terminate アクティビティーなど)

- 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
- Java 断片アクティビティー
- invoke アクティビティー
- human task 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。

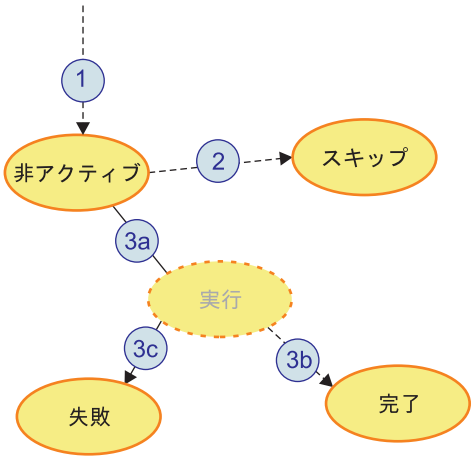
以下の図で使用される規則

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、および terminate アクティビティー) 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、および失敗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。

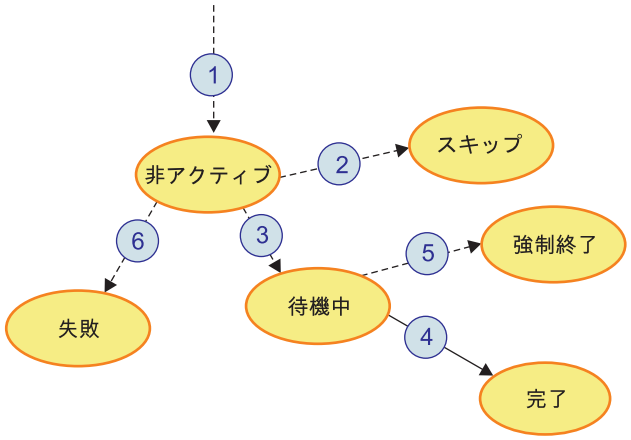
  アクティビティーの状態はスキップに変わり (2)、アクティビティーを離れるリンクはデッド・パスとしてナビゲートされます。
- 結合条件が `false` と評価され、**suppressJoinFailure** 属性が `false` に設定されます。

  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。アクティビティーは活動状態にされ、その状態は実行に変化します (3a)。アクティビティーの実装が実行され、アクティビティーの状態は終了に変化します (3b)。実装が失敗する場合 (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など)、アクティビティーの状態は失敗に変化します (3c)。すべての short-lived アクティビティーは中断不可能です。その結果、実行状態は表示されません。

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。



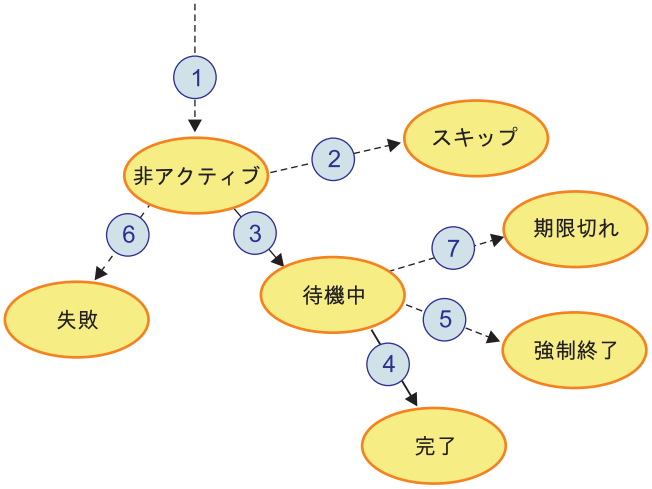
receive および wait アクティビティーの開始段階は、short-lived アクティビティーの開始段階と同じです。しかし、receive および wait アクティビティーがアクティブ化された後、状態は実行ではなく待機に変化します (3)。 receive または wait アクティビティーは外部要求を受け取るか、または指定されたタイムアウトを待機する準備ができました。その後で完了し、終了状態へと移行します (4)。 receive アクティビティーの場合、終了状態への遷移は、受信されたメッセージによってトリガーされます。 wait アクティビティーの場合、この遷移は指定された待機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。

wait または receive アクティビティーの実装は、単純な short-lived アクティビティー・タイプと比べて複雑です。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。この失敗により、アクティビティー状態が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)。

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。

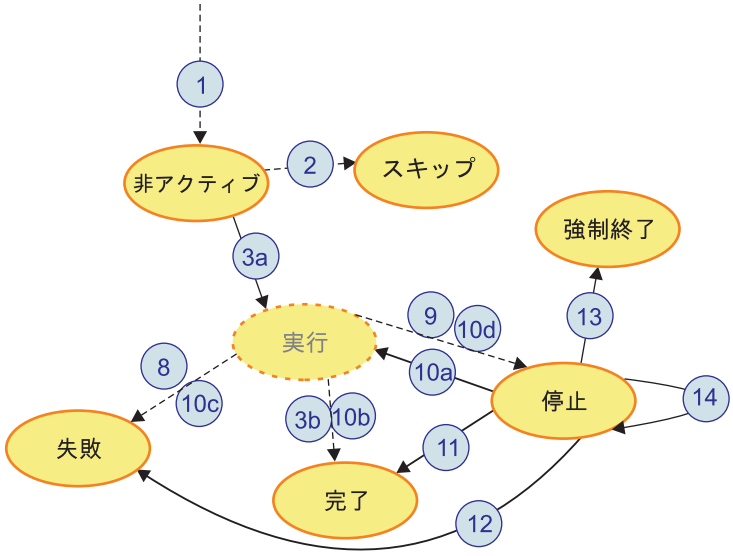


pick アクティビティーの状態および状態遷移 (1) から (6) は、receive アクティビティーの状態および状態遷移と同じです。

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。

Java 断片アクティビティーの状態遷移図

次の図は、Java 断片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。



Java 断片アクティビティーのランタイム動作は、short-lived アクティビティーのランタイム動作に似ています。Java 断片アクティビティーが活動状態にされると

(3a)、実装が実行されます。 Java 断片の実装は短いため、実行状態は表示されません。 Java 断片が正常に完了した場合、アクティビティーの状態はすぐに終了に変化します (3b)。

Java 断片が失敗した場合、アクティビティーは以下のいずれかの方法で動作できます。

- アクティビティー状態が停止に変化します (9)。アクティビティーの **continueOnError** 属性が `false` に設定され、Java 断片によってスローされた障害を Java 断片のエンクロージング・スコープで障害ハンドラーがキャッチできない場合、プロセスのナビゲーションが続行できるようにするため、管理対話が必要です。
- アクティビティー状態は失敗に変化し (8)、障害処理が開始されます。

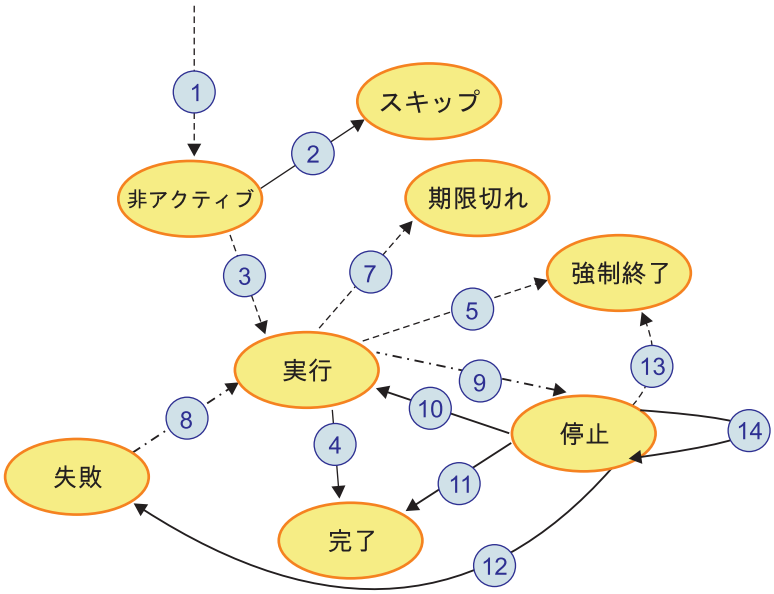
アクティビティーが停止状態の場合、API 要求を使用して強制再試行または強制完了できます。強制再試行する場合、アクティビティーは実行状態に置かれ、実装は再度実行されます (10a)。外部条件またはプロセス内部条件が変更されたため、Java 断片は正常に完了でき、アクティビティーの状態は終了に変化します (10b)。アクティビティーが再度失敗する場合、管理者は強制再試行 API の実装を使用してアクティビティーを停止するか (10d)、障害をプロセス障害処理 (10c) に任せます。

アクティビティーが停止状態中にエンクロージング・プロセスまたはスコープが終了される場合、アクティビティー状態は強制終了に変化します (13)。

アクティビティーが強制的に完了状態にされる場合、アクティビティーの実装は再度実行されません。アクティビティーは終了 (11)、失敗 (12)、または停止状態 (14) になります。

invoke アクティビティーの状態遷移図

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。プロセスおよびサービス・コンポーネントの SCA 修飾子が、サービスの呼び出しが同期か非同期かを判別します。



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。非同期サービスが子プロセスである場合、アクティビティーが実行状態の間は、アクティビティーの強制再試行または強制完了を実行できません。

Java 断片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。

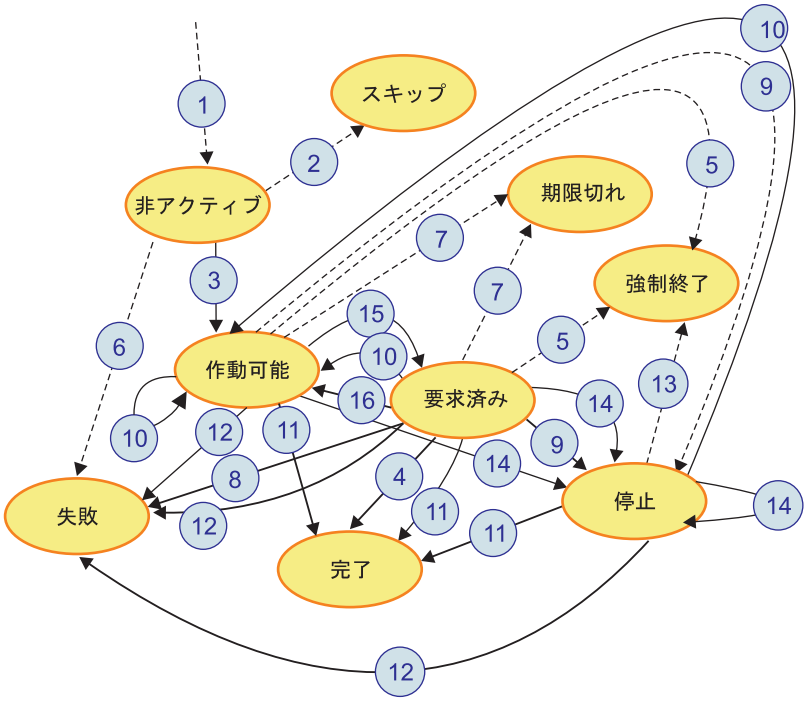
例えば、プロセス内の並列パスでの障害などのためにアクティビティーのエンクロージング・スコープが強制終了されたときに、そのアクティビティーが実行状態の場合は、そのアクティビティーも強制終了され、強制終了状態になります (5)。

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。

human task アクティビティーの状態遷移図

次の図は、human task 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。



human task 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、human task 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)。

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。

関連概念

28 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。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。

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、SCA (Service Component Architecture) コンポーネントの実装タイプです。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。 Business Process Choreographer クライアント・アプリケーションもこの API を使用して、ビジネス・プロセスをサービス・プロバイダーとして活用できます。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操作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、および JMS クライアントの設計に使用できる、EJB、Web サービス、および JMS メッセージの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 クライアントおよび他の SCA コンポーネ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

逆に、ビジネス・プロセスは SCA コンポーネントの実装であるため、ビジネス・プロセスで消費されるサービスは、他の SCA コンポーネントまたは SCA インポートからの提供を受けることができます。アウトバウンド対話で使用される BPEL パートナー・リンクは、SCA 参照で表されます。この参照は、他の SCA コンポーネントまたはインポートに結合できます。また、SCA 修飾子を使用してサービス品質属性を対話に関連付けることができます。

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

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。片方向操作は、完了が呼び出し側クライアントに知らされないサービス実行を意味します。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。要求応答操作は、完了が呼び出し側クライアントに知らされるサービス実行を意味します。サービス実行は、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において、呼び出し側のクライアントは、呼び出されるサービスに以下の方法で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・サイド、サービス・サイド、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・リンクのコンテキストに対して設定できます。エンドポイント参照は、呼び出す Web サービスの通信エンドポイントを一意的に決定します。一般に、どの Web サービスでも呼び出すこ

とができます。エンドポイント参照は、Web サービス・バインディングまたは SCA バインディングのいずれかに適用できます。

- 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 SCA 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・モデルでエンドポイント参照を設定でき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開発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。パートナー・リンクをサービス・エンドポイントと静的に関連付ける代わりに、エンドポイント参照をパートナー・リンクに割り当てることによって、関係を動的に設定できます。プロセス定義に含めた割り当てアクティビティーによってパートナー・リンクをエンドポイント参照で初期化した後に、パートナー・リンクをアウトバウンド対話に使用する場合は、パートナー・リンクに関連付けられた SCA 参照を別のサービスに静的に結合する必要はありません。

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を組み込んでください。クライアントが 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。サービス・インターフェースの優先対話スタイルは無視されます。

プロセスの実行時バインディング

このシナリオでは、ビジネス・プロセスがクライアントまたは呼び出されるサービスとして使用されると想定しています。これは、validFrom プロセス・プロパティーによって決定される使用中プロセス・バージョンに関する実行時バインディングを適用します。

ターゲット・プロセス・サービスのプロセス・テンプレート名は、以下の場合に使用されます。

- クライアント・プロセスでは、ターゲット・プロセス・サービスのプロセス・テンプレート名は、プロセス定義のパートナー・リンクで設定されます。

- ターゲット・プロセス・サービスのエンドポイント参照は getServiceRefForProcessTemplate Java 断片 API または getServiceRefForProcessTemplate XPath 関数によって評価され、パートナー・リンクに割り当てられます。

関連概念

5 ページの『プロセスのバージョン管理』
ビジネス・プロセスの新規バージョンを作成することができるため、同じプロセス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ことができます。

ビジネス・プロセスとサービスの間のパラメーターの引き渡し

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インプリメントされます。 SCA サービスに渡されるデータは、1 つ以上の BPEL 変数から検索されます。通常、データは値によって受け渡されます。これは、呼び出されたサービスがデータのコピーを処理することを意味します。

特定の環境では、データを参照によって受け渡すことができます。参照によるデータの受け渡しにより、ビジネス・プロセスのパフォーマンスを改善することができます。

以下の条件が**すべて**満たされる場合、データは参照によってビジネス・プロセスに受け渡されます。

- サービスの呼び出しが同期である。
- BPEL プロセスと呼び出されるサービスが同じモジュールにある。
- データの交換には、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、最良実例として、呼び出されたサービスはデータを更新すべきではありません。その理由は、データに対して行われる変更は永続的ではないからです。長期間のプロセスの場合、現行のトランザクションがコミットすると変更は破棄され、Microflow の場合、プロセスが終了すると変更が破棄されます。さらに、呼び出されたサービスによって変数が更新されると、イベントは生成されません。

ビジネス・プロセスがサービスによって消費される

他のサービスによって消費されるビジネス・プロセスでは、プロセス・モデル内に、receive アクティビティー、pick アクティビティー、またはイベント・ハンドラーが含まれています。プロセスに受け渡されるデータは 1 つ以上の BPEL 変数に書き込まれます。通常、データは値によって受け渡されます。

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

18 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、SCA (Service Component Architecture) コンポーネントの実装タイプです。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。

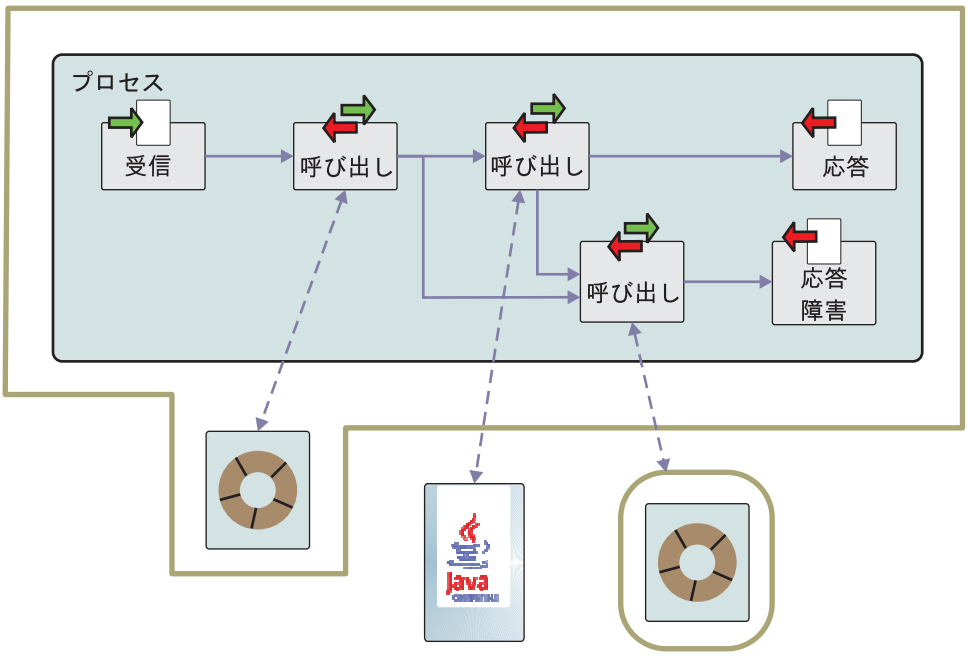
Microflow のトランザクションの振る舞い

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 human task 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内で実行されますが、Microflow の実行には複数のトランザクションを含めることができます。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 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 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。

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 **asychronousInvocation** 修飾子を `commit` に設定する必要があります。

関連概念

32 ページの『ビジネス・プロセスの補正』
補正は、正常に完了したプロセス内の操作を元に戻すための手段です。

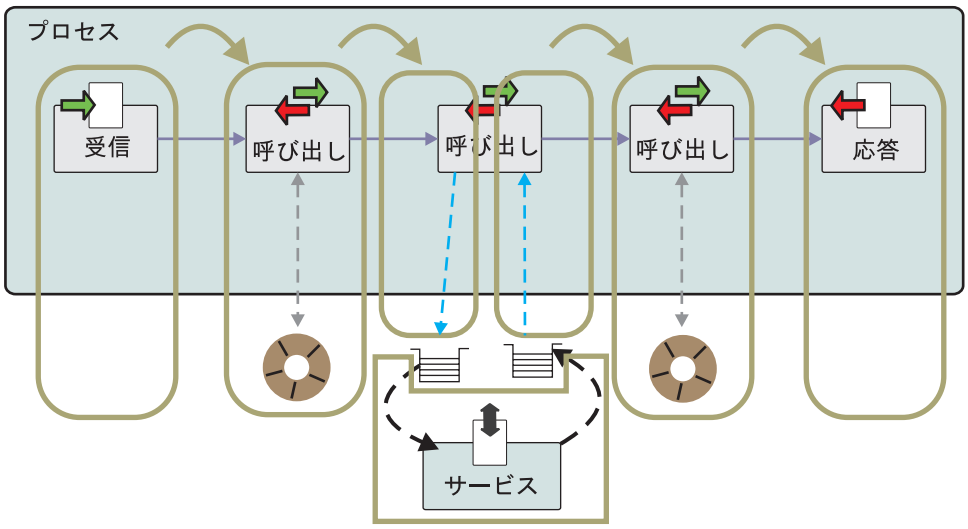
長期実行プロセスのトランザクションの振る舞い

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) メッセージまたは外部イベントによってトリガーされます。メッセージング・キューに新規メッセージを入れることにより、トランザクションは後続トランザクションをトリガーできます。

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。
- human task 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。

- プロセスを開始する receive アクティビティーまたは pick アクティビティーの要求を受け取る場合。
- 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、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、つまりサブプロセス と解釈することをお勧めします。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 human task 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。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 human task アクティビティーでは、human task 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 human task 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 human task 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 human task 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 human task 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。

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。

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。

注: Informix®、 Oracle、および Derby データベース・システムの場合、プロセス・インスタンス内の並列分岐のナビゲーション・トランザクションは直列化されているため、並列で実行することはできません。これは、データベース・エンティティーのロックが 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、このような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。

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。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、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 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トランザクションに参加させるには、長期実行プロセスの **asychronousInvocation** 修飾子を `commit` に設定する必要があります。

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。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

10 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。

障害の処理

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に移動します。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。

障害ハンドラーでは、特定の障害名、障害タイプ、またはその両方を catch できます。障害が発生すると、Business Flow Manager は障害ハンドラーを使用して障害を突き合わせます。Business Flow Manager は、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に対する障害ハンドラーを探します。障害ハンドラーの選択には、以下のルールが使用されます。

- その障害に関連する障害データがない場合、Business Flow Manager は一致する障害名を持つ障害ハンドラーを使用します。該当する障害ハンドラーがない場合は、catch-all 障害ハンドラーを使用します (使用可能な場合)。障害変数が定義された障害ハンドラーでは、データを何も持たない障害を 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が見つからない場合、Business Flow Manager は、障害名と、障害データのタイプに一致するタイプの障害変数を持たない障害ハンドラーを使用します。適切な障害ハンドラーが見つからない場合は、catch-all 障害ハンドラーを使用します (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では、データを持つ障害を catch することはできません。

これらの障害ハンドラー定義に一致しない障害が発生した場合、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、直接含まれる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補正と逆の順序で実行してから、障害を次のレベル、つまり含まれる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に対して障害の突き合せを再試行します。

特定の障害ハンドラーも、障害ハンドラー・チェーン内のどの catch-all 障害ハンドラーも、障害を catch しない場合、障害はプロセス・スコープに到達し、プロセスは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。

障害ハンドラーの設計

障害ハンドラーを設計するときは、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。
  - これが要求応答操作である場合に、障害を応答する。
  - ヒューマン・タスクを起動して問題を修正する。障害ハンドラーで問題を解決できない場合は、プロセスをロールバックし、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 **continueOnError** パラメーターを使用して、障害を管理的に処理することも考慮してください。

障害の発生

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。

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。

throw および rethrow アクティビティーを使用した障害の発生

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例外は、ビジネス・プロセス内で catch および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、ビジネス障害または標準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。

障害のプログラマチックな発生

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/ ネーム・スペースに 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、障害のネーム・スペースとしてプロセスのターゲット・ネーム・スペースを指定します。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。

reply アクティビティーの使用による、呼び出し元への障害の伝搬

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。このアクティビティーが有用なのは、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せず、即時に呼び出し元に戻ります。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。

障害に関する情報の取得

プロセスでは、システム障害を処理する必要があります。システム障害を catch するには、runtimeFailure 標準障害を catch するよう定義された障害ハンドラー、または catch-all 障害ハンドラーのいずれかを使用できます。場合によっては、障害に付随する情報が必要になることがあります。

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。

- 標準障害またはシステム障害時にデータが保管される障害変数。型付き変数を使用して障害を処理するには、StandardFaultType 複合タイプを手動で作成する必要があります。
- catch-all 障害ハンドラー。catch-all 障害ハンドラーには関連付けられた変数がありません。catch-all 障害ハンドラーから障害データを取得するには、Java 断片アクティビティーに getCurrentFaultAsException メソッドを使用します。この Java 断片アクティビティーは、catch-all 障害ハンドラーに組み込む必要があります。getCurrentFaultAsException メソッドを使用すると、システム障害だけではなく、あらゆるタイプの障害のデータを取得できます。

getCurrentFaultAsException 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、ユーザーは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()
```

エラー動作の継続

このパラメーターは、WebSphere Integration Developer のプロセスを定義するときに、一部のアクティビティーに対して使用可能です。このパラメーターでは、それらのアクティビティーによって障害が発生した場合にプロセスに対して行う処理を決定します。その障害に対して障害ハンドラーは定義されません。

アクティビティーに障害が検出されると、プロセスの障害処理が開始します。障害が、直接エンクロージング・スコープによって処理されず、**continueOnError** パラメーターが設定されていない場合、そのアクティビティーは停止し、プロセスを継続するために管理アクションが使用可能になります。停止したアクティビティーを持つ実行中のプロセス・インスタンスを検索するには、Business Process Choreographer API または Business Process Choreographer Explorer を使用できます。アクティビティーは、再始動することも完了を強制することもできます。**continueOnError** パラメーターが設定されている場合、標準の障害処理が適用されます。

例えば、forceRetry メソッドを使用して、異なる入力データでアクティビティーを再始動することも、forceComplete メソッドを使用して、出力データまたは障害データとともにアクティビティーを完了することもできます。

```
public interface BusinessFlowManagerService {
   public void forceRetry(String aiid, ClientObjectWrapper inputMessage,
          boolean continueOnError);
   ...
}
```

エラーが再発生した場合にプロセスが停止するよう設定する場合は、**continueOnError** パラメーターを使用不可にする必要があります。

forceComplete メソッドを使用してアクティビティーの完了が強制されると、再実行されることはありません。その出力メッセージを使用して、プロセス・ナビゲーションが続行されます。

ビジネス・プロセスの補正

補正は、正常に完了したプロセス内の操作を元に戻すための手段です。

補正処理は、プロセス・モデルで補正が定義された実行中のプロセス・インスタンスにおける、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。

長期実行プロセスの補正

長期実行プロセスに対する補正は、*ビジネス・レベル補正* とも呼ばれます。このタイプの補正は、スコープまたはプロセス・レベルで定義できます。これは、プロセスの一部、またはプロセス全体が補正可能ということです。

補正は、障害ハンドラー、スコープまたはプロセスの補正ハンドラーによって起動されます。補正は、プロセスのもう一つのナビゲーション・パスです。

長期実行プロセスは、子プロセスを囲む親スコープが補正されたときに、正常終了した子プロセスを自動的に補正します。プロセス内では、正常に完了した起動およびスコープ・アクティビティーのみ補正されます。

Microflow の補正

Microflow の補正は、*テクニカル補正* とも呼ばれます。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。したがって、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することをお勧めします。

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。

関連概念

22 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。

関連情報

Using compensation in processes with Business Process Choreographer

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 プロパティーは管理コンソールで指定されます。「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスター上で構成されている場合は、「**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」の下で、「**Business Process Choreographer**」 → 「**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。

このプロパティーは管理コンソールで指定されます。「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスター上で構成されている場合は、「**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」の下で、「**Business Process Choreographer**」 → 「**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、管理コンソールか管理スクリプトを使用して実行できます。

関連タスク

331 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。

351 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』
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。

332 ページの『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。

プロセスの許可および担当者割り当て

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。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。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。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。

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、J2EE のロールまたはインスタンス・ベースのロールとすることができます。

関連概念

91 ページの『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。

75 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 J2EE のロールまたはインスタンス・ベースのロールとすることができます。

ビジネス・プロセスの J2EE ロール

Business Process Choreographer の構成時に、J2EE ロールが設定されます。 J2EE 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、グローバル・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 (J2EE) 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。

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・ベースのロール

プロセスとアクティビティーには、事前定義の許可ロール・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、これらのロールをプロセスとアクティビティーに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) のリーダーになります。

ロール	許可されたアクション
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	プロセスのアクティビティーを管理します。

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が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。

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リーダー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。

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。

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 Manger 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 Manger API を使用することによって開始することもできます。どちらの方法でも、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結合された SCA クライアントを使用してプロセスを開始する場合、検査は省略される可能性があります。
  – スタンドアロン呼び出しタスクは 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が、インライン・タスクは内部インターフェースを使用します。したがって、インライン呼び出しタスクはスタンドアロン・タスクよりも動作が優れています。
  – 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。
- 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、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。

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。

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。

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リーダーのロールは、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 WebSphere Integration Developer で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リーダーも割り当てません。

アクティビティー管理

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。また、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・レベルでモデル化して、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することもできます。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。

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。同期 invoke アクティビティーの場合、このタスクは、呼び出しの失敗後にアクティビティーが停止した場合のみ作成されます。停止したら、管理タスクは修復要求の処理 (強制完了や強制再試行) に使用されます。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。

第 2 章 ヒューマン・タスクについて

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。

関連タスク

524 ページの『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。

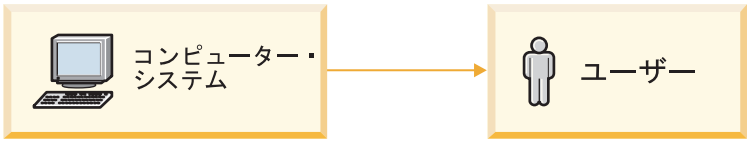
ヒューマン・タスクの種類

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。

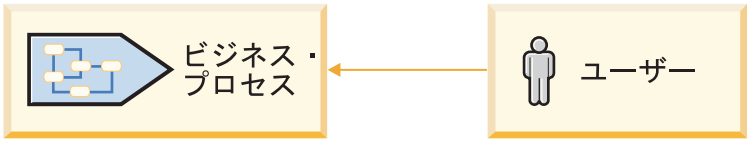
予定 (参加) タスク

サービスと人との対話をサポートします。これにより、人がサービスをインプリメントできます。例えば、ビジネス・プロセス内のヒューマン・タスク・アクティビティーを予定タスクにすることができます。



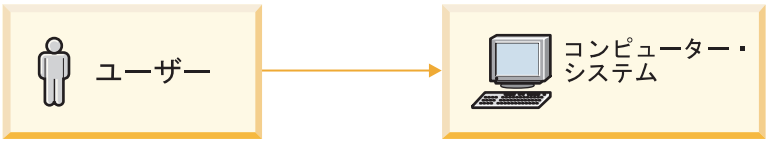
管理タスク

管理タスクは、プロセスで発生する技術的な問題を解決するために管理者が使用します。現在、ビジネス・プロセスにのみ管理タスクを使用できます。



呼び出し (親) タスク

人とサービスとの対話をサポートします。これにより、ユーザーがサービスを作成して、それを開始することができます。例えば、ユーザーがビジネス・プロセスを開始し、そのプロセスに呼び出しタスクを使用してイベントを送信することができます。



コラボレーション (人のみ) タスク

人と人との対話をサポートします。これにより、構造化され制御された方法で、個人が他の個人と作業を共用できます。コラボレーション・タスクは、ビジネス・プロセスまたは他のサービスとは直接対話しません。

ヒューマン・タスクのバージョン管理

同一のヒューマン・タスクの代替インスタンスをランタイム環境に共存させる場合には、バージョン管理を使用します。

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・テンプレートのバージョンは、その有効開始日によって決まります。つまり、あるタスク・テンプレートの異なるバージョンに同じタスク・テンプレート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・テンプレートのバージョンは、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・テンプレートのバージョンは、モデリング中かまたはタスク・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・テンプレートを呼び出します。有効開始日に照らして有効な、タスク・テンプレート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・テンプレートが使用され、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・テンプレート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・テンプレートは、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、現行で有効なタスク・テンプレートのバージョンが使用されます。タスク・テンプレート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・テンプレートを使って、継続して実行されます。これにより、タスク・テンプレートは次の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・テンプレート
- タスク・テンプレートが有効でなくなっても、このテンプレートは、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場合があります。
- 有効開始日に従って将来的に有効になるタスク・テンプレート

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いつでも、有効開始日付が将来ではないタスク・テンプレート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・インスタンスには必要ありません
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・テンプレートは非随時タスク・インスタンスを作成します • 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・テンプレートのマルチリンガル記述に置換式 (%htm:input.part% など) が含まれている場合、タスク・インスタンス (TASK_DESC ビュー) はこれらの行のみを継承します。つまり、タスク・インスタンスの説明がタスク・テンプレートの説明からコピーされた場合は、マルチリンガル表示名のみがタスク・インスタンスにコピーされます。

関連資料

484 ページの『TASK_TEMPL ビュー』
この定義済みデータベース・ビューは、タスクのインスタンスを生成するために使用できるデータを保持します。

480 ページの『TASK ビュー』
この定義済みデータベース・ビューは、タスク・オブジェクトの照会に使用します。

486 ページの『TASK_TEMPL_CPROP ビュー』
この定義済みデータベース・ビューは、タスク・テンプレートのカスタム・プロパティーを照会するために使用します。

483 ページの『TASK_CPROP ビュー』
この定義済みデータベース・ビューは、タスク・オブジェクトのカスタム・プロパティーを照会するために使用します。

486 ページの『TASK_TEMPL_DESC ビュー』
この定義済みデータベース・ビューは、タスク・テンプレート・オブジェクトのマルチリンガル記述データを照会するために使用します。

484 ページの『TASK_DESC ビュー』
この定義済みデータベース・ビューは、タスク・オブジェクトのマルチリンガル記述データを照会するために使用します。

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 3.

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 SOA パターンに従っており、スタンドアロン・タスク (予定タスク) を呼び出すコンポーネント、またはスタンドアロン・タスク (呼び出しタスク) によって呼び出されるコンポーネント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。

これらのタスクは、SCA 手段により、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、常にその Common Event Infrastructure (CEI) と監査ログ・イベントを Human Task Manager イベントとして発行します。

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・タスクですが、2 つのヒューマン・インターフェースを持っているため、サービス・コンポーネントに結合することはできません。

管理タスクは、スタンドアロン・タスクとしても、SCA コンポーネントとしても使用することはできません。

以下の規則は、ビジネス・プロセスで使用されるスタンドアロン・タスクに適用されます。

- スタンドアロン・タスクのライフ・サイクルは、プロセスから独立している。
  - デフォルトでは、予定タスクはプロセスによって作成されます。プロセスが削除されると、予定タスクは削除されます。
  - 呼び出しタスクはプロセスを作成できます。ただし、呼び出しタスクはプロセスが削除されても削除されないため、タスクの結果を確認することができます。
- 予定タスクは、ビジネス・プロセス内の invoke アクティビティーとして表示可能である。
- 呼び出しタスクは、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)、または on-event イベント・ハンドラーに結合される。
- タスクの説明、表示名、および資料は、複数の言語を同時にサポートする。
- スタンドアロン・タスクには、プロセス・コンテキストへのアクセス権限がない。スタンドアロン・タスクは、プロセス変数、カスタム・プロパティー、または他のプロセス・アクティビティーからのデータにアクセスすることはできません。

コラボレーション・タスクが最上位タスクの場合、そのライフ・サイクルは独立して管理され、手動で削除されるか、または指定された時間が経過すると自動的に削除されます。 コラボレーション・タスクがサブタスクか後続タスクの場合、そのラ

イフ・サイクルはその親タスクまたは最上位タスクによって管理されます。

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、および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・コンテキストおよびプロセス・データ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 Business Flow Manager のヒューマン・タスク・アクティビティー・イベントとして発行します。それぞれのサブタスクおよび後続タスクは、イベントを Human Task Manager の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。予定タスク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。
  - 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・サイクルはプロセスには依存せず、その結果はプロセスが削除された後でも表示できます。
- 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・タスクには、有効期限までの期間はない。ただし、予定タスクに対応するヒューマン・タスク・アクティビティーには有効期限を定義することができます。
- インラインの呼び出しタスクのみ、削除までの期間があるが、その期間は、呼び出しタスクが Human Task Manager API を使用して開始される場合にのみ適用される。
- 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 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。

- 予定タスクのリーダー、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リーダー・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。

ヒューマン・タスクとビジネス・プロセスの関係

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 アクティビティーを呼び出すための許可に影響を与えることができます。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 アクティビティーを開始する場合のビジネス・プロセスの呼び出しが含まれています。

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。プロセスに対して管理タスクが WebSphere Integration Developer でモデル化されていない場合は、実行時にデフォルトの臨時タスクが作成されます。このデフォルト・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、そのプロセスにはリーダーを割り当てません。

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。

関連概念

60 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。

37 ページの『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・ベースのロール』
プロセスとアクティビティーには、事前定義の許可ロール・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、これらのロールをプロセスとアクティビティーに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。

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

サブタスクに対して指定されるもの以外に、サブタスクは、その親タスクから以下の許可のロールも継承します。

- 親タスクのリーダー、エディター、オリジネーター、および所有者は、サブタスクとそのエスカレーションのリーダーになります。
- 親タスクの管理者はサブタスクの管理者になります。
- 親タスクのエスカレーション受信側は、サブタスクのリーダーになります。

ライフ・サイクルの考慮事項

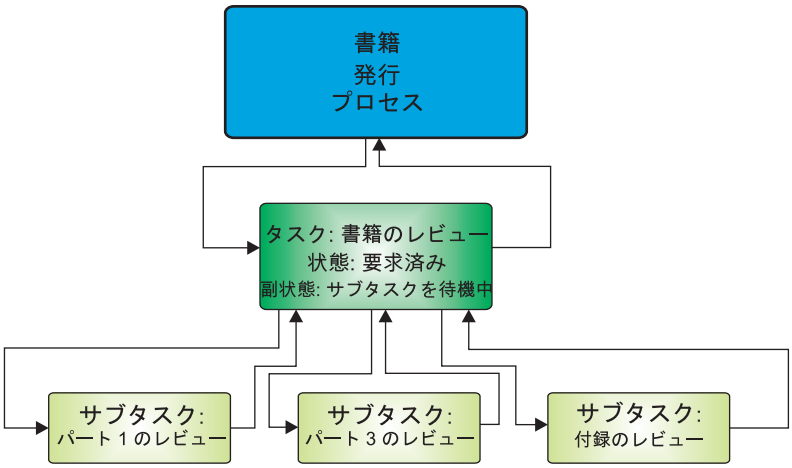
最初のサブタスクが開始すると、その親タスクは、サブタスク待機副状態になります。親タスクは、最後のサブタスクが、完了、失敗、期限切れ、または強制終了のいずれかの最終状態に到達するまでこの副状態に留まります。親タスクのいくつかのライフ・サイクル操作 (状態変更) は、そのサブタスクに伝搬されます。そのため、親タスクが中断、再開、強制終了、または削除されるか、期限切れになると、そのサブタスクもすべて、中断、再開、強制終了、削除されるか、期限切れになります。親タスクのエスカレート済み副状態は伝搬されません。サブタスクは、親タスクがエスカレートされてもエスカレートされません。サブタスクは、それぞれエスカ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ライフ・サイクル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。サブタスクは、その親タスクが削除されると削除されます。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。すべての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を出て、完了することができます。

関連概念

75 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 J2EE のロールまたはインスタンス・ベースのロールとすることができます。

60 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、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。

後続タスクの対象は、コラボレーション・タスクのみです。コラボレーション・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、チェーン内の最上位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、後続タスク内で変更できます。

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。

- 先行タスクのリーダー、エディター、オリジネーター、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リーダーになります。
- 先行タスクの管理者は、後続タスクの管理者になります。
- 先行タスクのエスカレーション受信側は、後続タスクのリーダーになります。

ライフ・サイクルの考慮事項

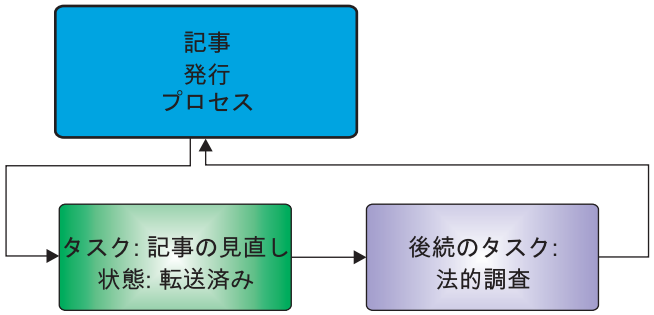
後続タスクが開始されると、先行タスクは転送済み状態になります。後続タスクは、その先行タスクの子であるため、先行タスクの一部のライフ・サイクル操作 (状態変更) は後続タスクに伝搬されます。先行タスクが中断、再開、エスカレート、終了、削除、または有効期限切れの状態である場合、その後続タスクのすべても中断、再開、エスカレート、終了、削除、または有効期限切れの状態になります。

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、チェーン内の最上位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。
- 後続タスクでの作業は、親タスクでの作業が続行していても停止する必要がある場合があるため、後続タスクは中断および再開できます。
- 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。
- 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という出力を続行しますが、これは実際には「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。

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。

タスクのエスカレーションを定義するには、タスク・テンプレートのエスカレーション・テンプレートを定義するか、または実行時に臨時のタスクを付加してエスカレーション・テンプレートを定義します。

エスカレーションは特定のタスク状態で活動状態にされ、エスカレーションの制限時間が満了するときに、まだ予期されたタスク状態 (監視状態) に達していない場合にのみエスカレートします。エスカレーション・タイムアウトの制限時間はストリングとして指定され、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。

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　要求済み状態の予定タスクまたはコラボレーション・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「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。
- このタスクのサブタスクが、サブタスクの予期されたタスク状態である「完了」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。あるいは、waiting-for-subtasks アクティブ化状態を使用して、サブタスクの進行状況を追跡できるようにすることもできます。

サブタスクの待機中
予定タスクまたはコラボレーション・タスクの「サブタスクの待機中」副状態では、このタスクのサブタスクが時間内に完了されない場合に、予期されたタスク状態である「サブタスク完了」を使用してエスカレートできます。

実行中　呼び出しタスクの実行状態では、呼び出されたサービスが、予期されたタスク状態である「終了」を使用して時間内に戻らない場合にエスカレートできます。

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、作業項目作成用のユーザー ID のセットに解決する必要があります。

increasePriority プロパティーを使用して、エスカレートされるタスクの優先順位を大きくするようにエスカレーションを定義できます。優先順位は、最初の反復のみ、またはエスカレーションの反復ごとに自動的に大きくすることができます。

エスカレーション・ライフ・サイクル

エスカレーションは、そのライフ・サイクル中に以下の状態になります。

- 作成後、タスクがアクティブ化状態になるまでエスカレーションは非アクティブ状態のままになります。
- 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ます。タイマーが始動し、タイムアウトになるまでエスカレーションは待機します。
- タイムアウトが発生すると、監視下にあるタスクの atLeastExpectedState プロパティーが検査されます。タスクがこの状態まで達するか、またはこの状態を通過した場合、エスカレーション状態は過剰状態に置かれます。予期された状態にまだ達していない場合、エスカレーションはエスカレート済み状態に置かれ、モデル・エスカレーション・アクションが呼び出されます。

エスカレーションが作成された後は、変更できません。エスカレーション・アクションは繰り返し実行できます。繰り返し間隔は、エスカレーションの autoRepeatDuration プロパティーで定義されます。

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は、タスクがチェーン内の最初のエスカレーションについて開始状態になると活動状態にされます。チェーン内のすべてのエスカレーションは同じアクティブ化状態である必要があります。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と比べて計算されます。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と比較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。
- 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。

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	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

92 ページの『担当者ディレクトリー』
担当者ディレクトリーには、担当者解決に使用するユーザー情報が保管されます。

関連タスク

624 ページの『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。

エスカレーションを通知する E メール

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は作業項目を受け取ります。また、タスクがエスカレートされたことを示す E メールを送信できます。特定の規則が E メールに適用されます。

エスカレーションはそれぞれ異なる E メールを扱うことができます。例えば、組織の標準に適合するように、標準エスカレーションの E メールを個別設定できます。E メールを個別設定するには、ヒューマン・タスク・エディターでエスカレーションの詳細を編集します。

E メールの件名行と本文テキストの両方に置換式を含めて、E メールをそれが参照するタスク (例えばタスク名を含む) に関連付けることができます。これらの式は E メールが送信される前に設定する必要があります。そうしないと、E メールの受信側では、E メールに *%variable name%* と表示されます。すべてのタスクおよびエスカレーション式を使用できます。

以下の HTML 断片は、置換式を含むサンプル E メールを示しています。

```
<html>
<head>
</head>
<body lang="EN-US"><div>
<p>The task '<b><span style="font-size:14.0pt">%htm:task.displayName%</span></b>'
with id '<b><span style="font-size:14.0pt">%htm:task.instanceID%</span></b>
' has been escalated because the </p>
<p>expected state '<b><span style="font-size:14.0pt">%htm:escalation.expectedTaskState%</span>
</b>'
 has not been reached in time.
</p>
<p> </p>
<p>The task has the following description: </p>
<p><span style="font-size:14.0pt;color:red">%htm:task.description%</span></p>
<p> </p>
<p><span style="font-size:14.0pt;color:green">The name of the Escalation is: %htm:escalation.displayName%
 and the escalation description reads: %htm:escalation.description%</span></p>
<p> </p>
<p><a href="%htm:task.URLPrefix%?id=%htm:task.instanceID%">Task details</a></p>
<p><a href="%htm:escalation.URLPrefix%?id=%htm:escalation.instanceID%">Escalation details</a></p>
</div>
</body>
</html>
```

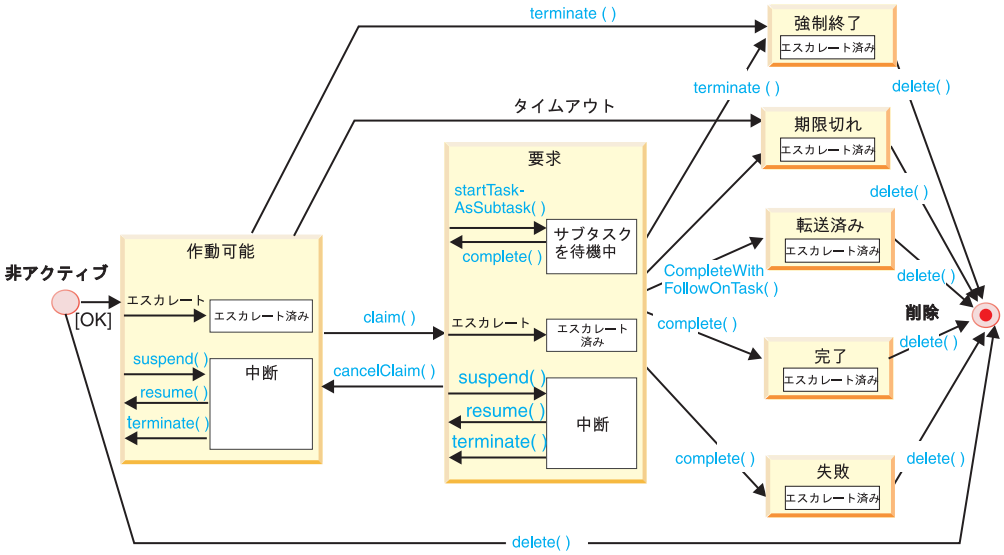
ヒューマン・タスクのライフ・サイクル

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。

予定タスクおよびコラボレーション・タスク

予定タスクおよびコラボレーション・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)、または公開されて使用可能になっている 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。予定タスクとコラボレーション・タスクは、その開始方法が異なっています。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。コラボレーション・タスクは、ユーザーによって作成され、開始されます。

以下の図は、予定タスクおよびコラボレーション・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。



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、タスクの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えますが、タスクを要求することはできません。予定タスクまたはコラボレーション・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。

- 時間が経過してもタスクが要求されないために、タスクがエスカレートされる。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。
- タスク所有者が作業の完了を誰か他の人に代行してもらいたい場合は、コラボレーション・タスクを作成して、後続タスクで当該タスクを完了することができます。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。

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ていないために、あるいはタスクがサブタスクを待っている時間が長すぎる場合に、その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。

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、そのサブタスクも削除されます。タスクが転送済み終了状態にある場合は、自動削除は適用されません。この場合は、親タスクは後続タスクと一緒に削除されます。後続タスクが終了状態に達すると、削除タイマーが開始します。

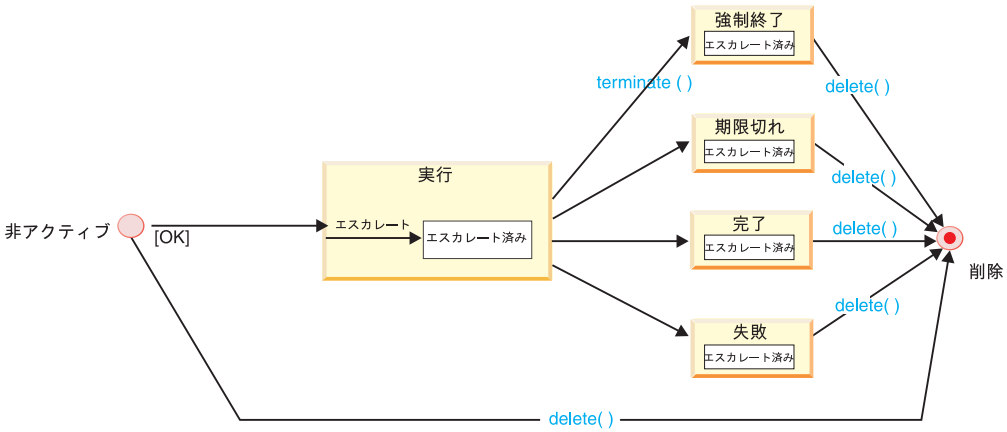
インライン予定タスクに適用される追加規則もいくつかあります。インライン・タスクは、ビジネス・プロセスに不可欠の部分であり、そのライフ・サイクルはプロセスのライフ・サイクルによって制御されます。

- 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。

呼び出し (親) タスク

呼び出しタスクは、ユーザーが Web サービスを呼び出すのをサポートします。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。

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

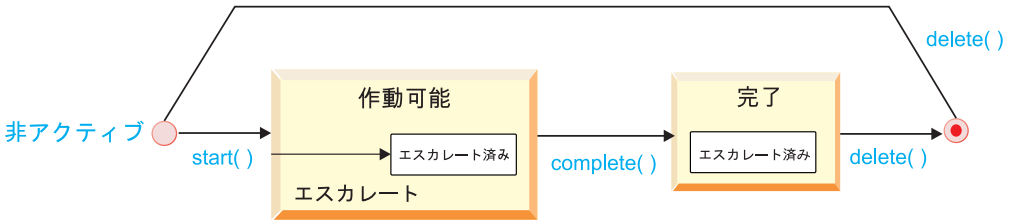
アクティビティー、pick (receive choice) アクティビティー、または on-event イベント・ハンドラーに関連付けられます。

- 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。
- インライン呼び出しタスクには期限切れまでの期間がないので、手動で終了させることはできません。
- 呼び出しタスクがビジネス・プロセスによって暗黙で開始される場合は、そのタスクの削除もビジネス・プロセスと一緒に暗黙で実行されます。
- タスクが Human Task Manager API によって開始される場合は、そのタスクはビジネス・プロセスと一緒には削除されません。タスクが自動削除を備えてモデル化されている場合は、そのタスクは、削除タイマーが満了になったときに削除されます。また、このタスクは手動で削除することもできます。

管理タスク

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・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。

関連概念

51 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。

47 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。

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) の実装および Web サービスの実装として提供されます。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。

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。

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、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。

呼び出しタスクを介した 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。次に、呼び出しタスク (SCA クライアント) は、同期的に呼び出された場合は同期的に、非同期的に呼び出された場合は非同期的に、関連する 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。

ビジネス・プロセスおよびアクティビティーを管理するために作成された管理タスク

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。Business Flow Manager は、管理を許可するすべてのビジネス・プロセスまたはアクティビティー・タイプに対する管理タスクを作成し、開始します。管理タスク・テンプレートがプロセスまたはアクティビティーに指定された場合は、このテンプレートが使用されます。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作成されます。

関連概念

68 ページの『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。

69 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。

72 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。

関連タスク

621 ページの『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。

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

65 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)
参照修飾子: RequestExpiration	`any`
参照修飾子: ResponseExpiration	`any`

修飾子のタイプ: 属性タイプ	値
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

65 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

修飾子のタイプ: 属性タイプ	値
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

65 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。

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。J2EE 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 J2EE のロールまたはインスタンス・ベースのロールとすることができます。

システム・レベルの Java 2 Platform, Enterprise Edition (J2EE) のロールは、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。インスタンス・ベースのロールは、個々のタスクおよびエスカレーション・インスタンス、またはタスクあるいはエスカレーション・インスタンスの作成に使用されるテンプレートに有効です。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。

J2EE のロール

以下の J2EE のロールがサポートされています。

- 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。

インスタンス・ベースのロール

タスク・インスタンスまたはエスカレーション・インスタンスは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。
可能なスターター	このロールのメンバーは、既存のタスク・インスタンスを開始できます。可能なスターターが指定されない場合、オリジネーターが可能なスターターになります。可能なスターターがない場合のインライン・タスクについては、デフォルトは全員となります。
スターター	このロール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。
可能な所有者	このロールのメンバーはタスクを要求できます。タスク・テンプレートに対して可能な所有者が定義されていない場合、すべてのユーザーがこのロールのメンバーと見なされます。このロールのスタッフの解決が失敗する場合、管理者は潜在的な所有者として割り当てられます。
所有者	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。

関連概念

36 ページの『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、J2EE のロールまたはインスタンス・ベースのロールとすることができます。

51 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。

100 ページの『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。

タスクの種類に対するインスタンス・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使用可能かどうかが判別されます。

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

注:

1. このロールには、管理対象プロセスまたはアクティビティーで管理アクションを行う権限もあります。
2. このロールには、管理対象プロセスまたはアクティビティーで読み取り操作を行う権限もあります。
3. このロールには、これらのタスク・タイプから生成されたエスカレーションでアクションを実行する権限がありますが、タスク自体で実行する権限はありません。

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できるタスクの 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 (例えば、電子メールを作成するための情報) を取り出します。担当者割り当て基準は、実行時に、タスク・モデルまたはプログラマチックに作成する場合にも使用されます。

担当者割り当て基準定義 (以前はスタッフ動詞と呼ばれていました) 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定義に関連付けます。

   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。

事前に定義されている担当者割り当て基準

担当者ディレクトリーから一連のユーザーを検出するための担当者割り当て基準が事前に定義されています。

WebSphere Integration Developer で担当者割り当て基準 (従来はスタッフ動詞と呼ばれていたもの) を使用して、ヒューマン・タスクに担当者割り当てを指定できます。これらの基準は、モデリングおよびデプロイメント中に担当者ディレクトリーについて実行できる一連の照会に変換されます。このセクションでは、次の定義済み担当者割り当て基準のパラメーターを示します。

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

- Users by user ID
- Users by user ID without Named Users

担当者割り当て基準を割り当てるときには、次の点を考慮してください。

- 大きな担当者グループを扱う場合は、「Group」担当者割り当て基準が最適です。この基準では、グループに属するメンバーがひとまとまりで扱われるためです。これにより、グループ間でヒューマン・タスクを容易に移動できます。担当者のグループ・メンバーシップは、担当者がログインし、ヒューマン・タスクにアクセスした時点で解決されます。
- 1 つのグループに属する担当員を個別にヒューマン・タスクに割り当てるには、Group 割り当ての代わりとして Group Members 担当者割り当て基準を使用できます。Group Members 担当者割り当て基準を使用すると、各担当員の割り当てが個別に作成されます。この割り当てを、別の担当者に転送できます。代替、つまり不在の担当者が別の担当者に置き換えられることがあります。この担当者割り当て基準に類似した基準として、Group Members without Named Users があります。この担当者割り当て基準では、職務分離割り当てパターンがサポートされています。

  注: グループに担当者を個別に割り当てる場合、特に多数の担当者をグループに個別に割り当てる場合は、実行時のパフォーマンスに影響を与えます。
- 同一グループに属していない複数の担当者を 1 つのヒューマン・タスクに割り当てる場合は、User Records by user ID 担当者割り当て基準定義を使用することを検討してください。また、モデリング中に静的に定義されない担当者割り当てであり、この割り当てに置換式が含まれている場合にも、この定義を使用できます。置換式は、カスタム・プロパティーまたはヒューマン・タスクの入力メッセージを参照できます。Users by user ID 担当者割り当て基準定義は、User Records by user ID 定義に似ています。Users by user ID 定義は、実行時には User Records by user ID 定義よりもパフォーマンスが優れていますが、機能性は劣ります。
  - ユーザー ID が正しく入力されているかどうかを検査しません。
  - 指定されているユーザー ID の E メール・アドレスなどを取得しません。このため、E メール・エスカレーションへの担当者の割り当てには適していません。
- Everybody 担当者割り当て基準定義も検討する価値があります。この割り当て基準は、1 つのヒューマン・タスクにすべての認証ユーザーが割り当てられることを示します。組織内のすべての担当者が特定のジョブを実行できるという状況が実際に発生する場合に、この定義は開発時やアプリケーションの迅速なプロトタイピング時に特に便利です。

Department Members

部門のメンバーを検索するときにこの基準を使用します。この基準は Lightweight Directory Access Protocol (LDAP) および virtual member manager 担当者ディレクトリー・プロバイダーによりサポートされています。

パラメーター	使用方法	タイプ	説明
DepartmentName	必須	string	検索するユーザーの部門名。部門名は次のいずれかの値に対応している必要があります。 • virtual member manager の場合は、virtual member manager グループの固有名 • LDAP の場合は、LDAP グループの識別名 (DN)
IncludeNestedDepartments	必須	boolean	ネストされた部門を照会で考慮するかどうかを指定します。
AlternativeDepartmentName1	オプション	string	ユーザーが属することのできる追加の部門。
AlternativeDepartmentName2	オプション	string	ユーザーが属することのできる追加の部門。

Everybody

WebSphere Process Server により認証されるすべてのユーザーをタスク・ロールに割り当てるには、この基準を使用します。この基準はパラメーターをとりません。

この基準は、すべての担当者ディレクトリー・プロバイダーでサポートされています。

Group

この基準は、グループをタスク・ロールに割り当てるときに使用します。この割り当てでは、割り当てユーザーごとにユーザー作業項目を作成する代わりに、グループ作業項目が作成されます。

この基準は、すべての担当者ディレクトリー・プロバイダーでサポートされています。

パラメーター	使用方法	タイプ	説明
GroupId	必須	string	検索するユーザーのグループ名。このパラメーターでは、置換式がサポートされています。グループ ID は、次のいずれかの値に対応している必要があります。 • virtual member manager の場合は、グループ項目の固有名 • LDAP の場合は、グループ項目の DN • ユーザー・レジストリー・プロバイダーの場合は、使用する名前の形式は、タスクが配置されるアプリケーション・サーバーに設定されているユーザー・リポジトリーによって異なります。 – ローカル・オペレーティング・システムの場合は、ローカル・オペレーティング・システムでサポートされているグループ名を使用します。 – 独立したカスタム・レジストリーの場合は、カスタム実装でサポートされているグループ名を使用します。 – 独立した LDAP レジストリーの場合は、グループ項目の DN を使用します。

Group Members

この基準は、1 つのグループのメンバーを検索するときに使用します。この基準は、LDAP、virtual member manager、およびユーザー・レジストリー担当者ディレクトリー・プロバイダーによりサポートされています。

パラメーター	使用方法	タイプ	説明
GroupName	必須	string	検索するユーザーのグループ名。このパラメーターでは、置換式がサポートされています。グループ ID は、次のいずれかの値に対応している必要があります。 • virtual member manager の場合は、グループ項目の固有名 • LDAP の場合は、グループ項目の DN • ユーザー・レジストリー・プロバイダーの場合は、使用する名前の形式は、タスクが配置されるアプリケーション・サーバーに設定されているユーザー・リポジトリーによって異なります。 – ローカル・オペレーティング・システムの場合は、ローカル・オペレーティング・システムでサポートされているグループ名を使用します。 – 独立したカスタム・レジストリーの場合は、カスタム実装でサポートされているグループ名を使用します。 – 独立した LDAP レジストリーの場合は、グループ項目の DN を使用します。
IncludeSubgroups	必須	boolean	ネストされたサブグループを照会で考慮するかどうかを指定します。
AlternativeGroupName1	オプション	string	ユーザーが属することのできる追加のグループ。
AlternativeGroupName2	オプション	string	ユーザーが属することのできる追加のグループ。

Group Members without Named Users

この基準は、明示的に指定されているユーザーを除く、グループのすべてのメンバーを検索するときに使用します。この基準は、LDAP、virtual member manager、およびユーザー・レジストリー担当者ディレクトリー・プロバイダーによりサポートされています。

パラメーター	使用方法	タイプ	説明
GroupName	必須	string	検索するユーザーのグループ名。このパラメーターでは、置換式がサポートされています。グループ ID は、次のいずれかの値に対応している必要があります。 • virtual member manager の場合は、グループ項目の固有名 • LDAP の場合は、グループ項目の DN • ユーザー・レジストリー・プロバイダーの場合は、使用する名前の形式は、タスクが配置されるアプリケーション・サーバーに設定されているユーザー・リポジトリーによって異なります。 – ローカル・オペレーティング・システムの場合は、ローカル・オペレーティング・システムでサポートされているグループ名を使用します。 – 独立したカスタム・レジストリーの場合は、カスタム実装でサポートされているグループ名を使用します。 – 独立した LDAP レジストリーの場合は、グループ項目の DN を使用します。
IncludeSubgroups	必須	boolean	ネストされたサブグループを照会で考慮するかどうかを指定します。
NamedUsers	必須	string	検索されたグループ・メンバーのリストから除外するユーザーのユーザー ID。このパラメーターでは、置換式がサポートされています。

Group Members without Filtered Users

この基準は、検索フィルターで定義されている一連のユーザーを除くすべてのグループ・メンバーを検索するときに使用します。この基準は、LDAP および virtual member manager 担当者ディレクトリー・プロバイダーによりサポートされています。

パラメーター	使用方法	タイプ	説明
GroupName	必須	string	検索するユーザーのグループ名。このパラメーターでは、置換式がサポートされています。グループ ID は、次のいずれかの値に対応している必要があります。 • virtual member manager の場合は、グループ項目の固有名 • LDAP の場合は、グループ項目の DN
IncludeSubgroups	必須	boolean	ネストされたサブグループを照会で考慮するかどうかを指定します。
FilterAttribute	必須	string	検索フィルターで使用する属性の名前。

パラメーター	使用方法	タイプ	説明
FilterValue	必須	string	検索フィルターで使用するフィルター値。フィルターではワイルドカード文字としてアスタリスク (*) を使用できます。

Group Search

この基準は、属性の一致に基づいてグループを検索し、そのグループのメンバーを割り当てるときに使用します。この基準は、LDAP および virtual member manager 担当者ディレクトリー・プロバイダーによりサポートされています。

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

virtual member manager の場合、Group エンティティーに、以下の Group Search 基準パラメーターと同等のプロパティーがあります。

- GS_GroupID: cn
- GS_DisplayName: displayName
- GS_BusinessCategory: businessCategory

Manager of Employee

この基準は、担当者の名前を使用してその担当者の管理者を検索するときに使用します。この基準は、LDAP および virtual member manager 担当者ディレクトリー・プロバイダーによりサポートされています。

パラメーター	使用方法	タイプ	説明
EmployeeName	必須	string	管理者が検索対象となる従業員の名前。従業員名は次のいずれかの値に対応している必要があります。 • virtual member manager の場合は、個人項目の固有名 • LDAP の場合は、個人項目の DN

Manager of Employee by user ID

この基準は、担当者のユーザー ID を使用してその担当者の管理者を検索するときに使用します。この基準は、LDAP および virtual member manager 担当者ディレクトリー・プロバイダーによりサポートされています。

パラメーター	使用方法	タイプ	説明
EmployeeUserID	必須	string	自分の管理者が検索対象となる従業員のログイン・ユーザー ID。このパラメーターでは、置換式がサポートされています。

Native Query

この基準は、ディレクトリー固有のパラメーターに基づいた固有の照会を定義するときに使用します。この基準は、LDAP および virtual member manager 担当者ディレクトリー・プロバイダーによりサポートされています。

パラメーター	使用方法	タイプ	説明
QueryTemplate	必須	string	使用する照会テンプレート。これは、search、user、および usersOfGroup のいずれかの値でなければなりません。
Query	必須	string	照会を指定します。このパラメーターでは、置換式がサポートされています。照会のタイプは、照会テンプレートによって異なります。 • 検索テンプレート: 検索フィルター – virtual member manager の場合は、有効な検索式 – LDAP の場合は、有効な LDAP フィルター • ユーザー・テンプレート: ユーザー DN – virtual member manager の場合は、ユーザー項目の固有名 – LDAP の場合は、ユーザー項目の DN • usersOfGroup: グループ DN – virtual member manager の場合は、グループの固有名 – LDAP の場合は、グループの DN
AdditionalParameter1	オプション	string	照会を指定します。このパラメーターでは、置換式がサポートされています。パラメーターのタイプは、照会テンプレートによって異なります。 • 検索テンプレート。再帰的検索を行うかどうかの指定に使用します。サポートされる値 : yes および no • ユーザー・テンプレート。非サポート • usersOfGroup。再帰的検索を行うかどうかの指定に使用します。サポートされる値 : yes および no

パラメーター	使用方法	タイプ	説明
AdditionalParameter2	オプション	string	この基準は、検索のベース・エントリーを指定するときに使用します。 • virtual member manager の場合は、ベース・エントリーの固有名 (例: dc=mycomp，dc=com) • LDAP の場合は、ベース・エントリーの DN
AdditionalParameter3	オプション	string	この基準は、追加パラメーターを指定するときに使用します。 デフォルトのマッピング XSLT ファイルを使用する場合、このパラメーターはサポートされません。
AdditionalParameter4	オプション	string	この基準は、追加パラメーターを指定するときに使用します。 デフォルトのマッピング XSLT ファイルを使用する場合、このパラメーターはサポートされません。
AdditionalParameter5	オプション	string	この基準は、追加パラメーターを指定するときに使用します。 デフォルトのマッピング XSLT ファイルを使用する場合、このパラメーターはサポートされません。

Nobody

この基準は、タスク・ロールへのユーザーのアクセスを拒否するときに使用します。この基準では、権限継承と担当員解決のデフォルトのみが適用されます。この基準はパラメーターをとりません。

Person Search

この基準は、属性の一致に基づいて担当者を検索するときに使用します。この基準は、LDAP、virtual member manager、およびユーザー・レジストリー担当者ディレクトリー・プロバイダーによりサポートされています。

パラメーター	使用方法	タイプ	説明
UserID	オプション	string	検索するユーザーのユーザー ID。
Profile	オプション	string	検索するユーザーのプロファイル。
LastName	オプション	string	検索するユーザーのラストネーム。
FirstName	オプション	string	検索するユーザーのファーストネーム。
MiddleName	オプション	string	検索するユーザーのミドルネーム。
Email	オプション	string	ユーザーの電子メール・アドレス。
Company	オプション	string	ユーザーが属する会社。
DisplayName	オプション	string	ユーザーの表示名。
Secretary	オプション	string	ユーザーの秘書。
Assistant	オプション	string	ユーザーのアシスタント。
Manager	オプション	string	ユーザーの管理者。
Department	オプション	string	ユーザーが属する部門。

パラメーター	使用方法	タイプ	説明
Phone	オプション	string	ユーザーの電話番号。
Fax	オプション	string	ユーザーの FAX 番号番号。
Gender	オプション	string	ユーザーが男性であるか、女性であるか。
Timezone	オプション	string	ユーザーが配置されている場所の時間帯。
PreferredLanguage	オプション	string	ユーザーの希望する言語。

virtual member manager では、PersonAccount エンティティーに、以下の People Search 基準パラメーターと同等のプロパティーがあります。

- PS_UserID: uid
- PS_LastName: sn
- PS_FirstName: givenName
- PS_MiddleName: initials
- PS_Email: mail
- PS_DisplayName: displayName
- PS_Secretary: secretary
- PS_Manager: manager
- PS_Department: departmentNumber
- PS_Phone: telephoneNumber
- PS_PreferredLanguage: preferredLanguage

Role Members

この基準は、ロールに関連付けられているユーザーを検索するときに使用します。この基準は、LDAP および virtual member manager 担当者ディレクトリー・プロバイダーによりサポートされています。

パラメーター	使用方法	タイプ	説明
RoleName	必須	string	検索するユーザーのロール名。
IncludeNestedRoles	必須	boolean	ネストされたロールを照会で考慮するかどうかを指定します。
AlternativeRoleName1	オプション	string	ユーザーの追加のロール名。
AlternativeRoleName2	オプション	string	ユーザーの追加のロール名。

User Records by User ID

この基準は、ユーザー ID が認識されているユーザーについての照会を定義するときに使用します。この基準は、LDAP および virtual member manager 担当者ディレクトリー・プロバイダーによりサポートされています。この基準では、該当するユーザーのユーザー ID、E メール情報、および優先ロケール (設定されている場合) が戻されます。

パラメーター	使用方法	タイプ	説明
UserID	必須	string	検索するユーザーのユーザー ID。このパラメーターでは、置換式がサポートされています。
AlternativeID1	オプション	string	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
AlternativeID2	オプション	string	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。

Users Records by User ID without Named Users

この基準は、ユーザー ID が認識されているユーザーについての照会 (明示的に指定されたユーザー ID を除く) を定義するときに使用します。この基準は、LDAP、virtual member manager、およびユーザー・レジストリー担当者ディレクトリー・プロバイダーによりサポートされています。この基準では、該当するユーザーのユーザー ID と E メール情報が戻されます。

パラメーター	使用方法	タイプ	説明
UserID	必須	string	検索するユーザーのユーザー ID。このパラメーターでは、置換式がサポートされています。
AlternativeID1	オプション	string	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
AlternativeID2	オプション	string	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
NamedUsers	必須	string	ユーザー ID のリストから除外するユーザーのユーザー ID。このパラメーターでは、置換式がサポートされています。

Users

この基準は、名前で認識されているユーザーについての照会を定義するときに使用します。この基準は、すべての担当者ディレクトリー・プロバイダーでサポートされています。

パラメーター	使用方法	タイプ	説明
Name	必須	string	検索するユーザーの名前。 • virtual member manager の場合は、個人項目の固有名 • LDAP の場合は、個人項目の DN • ユーザー・レジストリー・プロバイダーの場合は、使用する名前の形式は、タスクが配置されるアプリケーション・サーバーに設定されているユーザー・リポジトリーによって異なります。 – ローカル・オペレーティング・システムの場合は、割り当てるユーザーのユーザー ID を使用します。 – 独立したカスタム・レジストリーの場合は、カスタム実装でサポートされている個人名を使用します。 – 独立した LDAP レジストリーの場合は、個人項目の DN を使用します。
AlternativeName1	オプション	string	追加のユーザー名。このパラメーターは、複数のユーザーを検索するときに使用します。
AlternativeName2	オプション	string	追加のユーザー名。このパラメーターは、複数のユーザーを検索するときに使用します。

Users by User ID

この基準は、ユーザー ID が認識されているユーザーについての照会を定義するときに使用します。ショート・ネームを使用して、「wpsadmin」などの値を指定します。この基準では、担当員ディレクトリーへのアクセスは不要です。この基準は、すべての担当者ディレクトリー・プロバイダーでサポートされています。

パラメーター	使用方法	タイプ	説明
UserID	必須	string	検索するユーザーのユーザー ID。このパラメーターでは、置換式がサポートされています。
AlternativeID1	オプション	string	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
AlternativeID2	オプション	string	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。

Users by User ID without Named Users

この基準は、ユーザー ID が認識されているユーザーについての照会 (明示的に指定されたユーザー ID を除く) を定義するときに使用します。ショート・ネームを使用して、「wpsadmin」などの値を指定します。この基準では、担当員リポジトリーへのアクセスは不要です。この基準は、すべての担当者ディレクトリー・プロバイダーでサポートされています。

パラメーター	使用方法	タイプ	説明
UserID	必須	string	検索するユーザーのユーザー ID。このパラメーターでは、置換式がサポートされています。
AlternativeID1	オプション	string	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
AlternativeID2	オプション	string	追加のユーザー ID。このパラメーターは、複数のユーザーを検索するときに使用します。
NamedUsers	必須	string	ユーザー ID のリストから除外するユーザーのユーザー ID。このパラメーターでは、置換式がサポートされています。

関連概念

『担当者割り当て基準定義における置換式』
置換式を、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。

担当者割り当て基準定義における置換式

置換式を、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、以下の担当者割り当て基準定義は、パラメーターとして `htm:input.¥name` 置換式を指定しています。

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

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。担当者解決は、置換式を動的に実際のタスク入力メッセージ値で置き換えます。

関連情報

79 ページの『事前に定義されている担当者割り当て基準』
担当者ディレクトリーから一連のユーザーを検出するための担当者割り当て基準が事前に定義されています。

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。

関連概念

36 ページの『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、J2EE のロールまたはインスタンス・ベースのロールとすることができます。

担当者ディレクトリー

担当者ディレクトリーには、担当者解決に使用するユーザー情報が保管されます。

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。

関連概念

56 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。

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。

関連タスク

227 ページの『LDAP 担当者ディレクトリー・プロバイダーの構成』
このタスクを使用して、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。これにより、Business Process Choreographer では、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担当者割り当てを実行します。

225 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』
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担当者割り当てを実行するための、Business Process Choreographer 用の 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。デフォルトの担当者ディレクトリー・プロバイダーはいつでも使用できるので、カスタムの担当者割り当て基準を導入する場合に必要なのは、単に構成することだけです。

担当者照会への担当者割り当て基準のマッピング

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 を使用する場合は、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。各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

Windows® プラットフォームの場合、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。 Linux®、UNIX®、i5/OS® プラットフォームの場合、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構成に関連している XSL 変換ファイルは、特定のリポジトリーに固有の担当者照会を生成するときに使用します。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・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。

担当者割り当て基準の変換のカスタマイズ

次のような状況では、担当者割り当て基準の変換をカスタマイズしなければならない場合があります。それは、変換を LDAP 担当者およびグループ・スキーマに適合させる場合、またはカスタム担当者割り当て基準を定義する場合です。

カスタマイズした変換を使用するには、変換ファイルを作成する必要があります。デフォルトの変換ファイルを変換したり、これらのファイルの名前を自分の変換フ

ファイルに再利用したりしないでください。LDAP または Virtual Member Manager の担当者ディレクトリー・プロバイダーに対する新規変換ファイルの場合、ファイルに常に「**ユーザー ID 別のユーザー・レコード (User Records by User ID)**」担当者割り当て基準定義を組み込んでください。この定義は、担当者割り当てに明示的に指定されていない場合でも、Business Process Choreographer に必要です。

新規変換ファイルを使用するには、そのファイルを指す新規担当者ディレクトリー構成を定義します。

関連タスク

227 ページの『LDAP 担当者ディレクトリー・プロバイダーの構成』
このタスクを使用して、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。これにより、Business Process Choreographer では、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担当者割り当てを実行します。

225 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』
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担当者割り当てを実行するための、Business Process Choreographer 用の 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。デフォルトの担当者ディレクトリー・プロバイダーはいつでも使用できるので、カスタムの担当者割り当て基準を導入する場合に必要なのは、単に構成することだけです。

カスタム担当者割り当て基準の定義:

事前定義された一連の担当者割り当て基準は、独自の基準による拡張が必要になる場合があります。

カスタム担当者割り当て基準を作成したら、それを以下のファイルに組み込む必要があります。

- VerbSet.xml ファイル内
- 変換ファイル内

カスタマイズした変換を使用するには、変換ファイルを作成する必要があります。デフォルトの変換ファイルを変換したり、これらのファイルの名前を自分の変換ファイルに再利用したりしないでください。

例えば、Mentor of Employee という新規基準を作成したとします。この基準は Manager of Employee 基準に類似していますが、従業員の管理者ではなく、従業員の指導者のユーザー ID を検索します。

1. WebSphere Integration Developer で VerbSet.xml ファイルを変更します。
   a. VerbSet.xml ファイルをワークスペースにインポートします。

      VerbSet.xml ファイルは、com.ibm.wbit.tel.ui_6.1.*version_number*.jar ファイルに含まれています。このファイルは *shared_resources*/plugins ディレクトリーにあります。複数のバージョンのアーカイブ・ファイルが見つかった場合は、バージョン番号が最も大きいファイルを使用してください。

b. インポートした VerbSet.xml ファイルを編集し、以下の XML 断片を組み込みます。

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

c. ヒューマン・タスク・モジュールが新しい verb を使用できるようにします。

新しい verb を使用するモジュールを右クリックし、「**プロパティー**」、「**ヒューマン・タスク**」の順に選択します。「プロパティー」ページで、デフォルトの verb 設定ファイルを使用するためのチェック・ボックスを必ずクリアしてください。

2. 変換ファイルで、ユーザーの指導者の LDAP 属性を含めるよう、次のように新規 XSL 変数を定義します。

```
<xsl:variable name="DefaultMentorAttribute">mentor</xsl:variable>
```

3. 以下の XML 断片を変換ファイルに追加します。

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

    <sldap:intermediateResult>
      <xsl:attribute name="name">mentorvar</xsl:attribute>
      <sldap:user>
        <xsl:attribute name="dn">
          <xsl:value-of select="staff:parameter[@id='EmployeeName']"/>
        </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">
          <xsl:value-of select="$DefaultMentorAttribute"/>
        </xsl:attribute>
        <xsl:attribute name="destination">intermediate</xsl:attribute>
      </sldap:resultAttribute>
    </sldap:resultObject>



    <sldap:user>
```

```
        <xsl:attribute name="dn">%mentorvar%</xsl:attribute>
           <xsl:call-template name="ResultObjectSpecForUserData"/>
      </sldap:user>
    </sldap:staffQueries>
  </xsl:template>
  <!-- End template ManagerOfEmployee -->
```

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。

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。

ユーザーが不在の場合は代替する

- 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。

在社している場合はユーザーを選択

- 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。

関連タスク

233 ページの『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。

406 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。

408 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。

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リーダーになります。
- タスク管理者が、これらすべてのタスクのすべてのサブタスク、後続タスク、およびエスカレーションの管理者になります。

- タスク・リーダーが、これらすべてのタスクのすべてのサブタスク、後続タスク、およびエスカレーションのリーダーになります。
- 任意のタスク・ロールのメンバーが、このタスクのエスカレーション、サブタスク、および後続のタスクのリーダーになります。
- エスカレーション受信側が、エスカレートしたタスクのリーダーになります。

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
リーダー	継承のみが適用される	継承のみが適用される
エスカレーション受信者	管理者がエスカレーション受信側になる	管理者がエスカレーション受信側になる

次の継承規則がスタンドアロン・タスクに適用されます。

- タスク管理者が、これらすべてのタスクのすべてのサブタスク、後続タスク、およびエスカレーションの管理者になります。
- タスク・リーダーが、これらすべてのタスクのすべてのサブタスク、後続タスク、およびエスカレーションのリーダーになります。
- 任意のタスク・ロールのメンバーが、このタスクのエスカレーション、サブタスク、および後続のタスクのリーダーになります。
- エスカレーション受信側が、エスカレートしたタスクのリーダーになります。

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リーダー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リーダー権限を付与されます。

関連概念

75 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 J2EE のロールまたはインスタンス・ベースのロールとすることができます。

担当者割り当て基準および担当者解決結果の管理

タスク許可のロールに関連付けられている担当者割り当て基準は、デプロイ済みのタスク・テンプレートまたはタスク・インスタンスの存続期間中は常に有効です。

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、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。

関連タスク

333 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。

355 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。

337 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。

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. Business Process Choreographer がサーバー上に構成されている場合、「**サーバー**」 → 「**アプリケーション・サーバー**」 → ***server_name*** をクリックします。
2. Business Process Choreographer がクラスター上に構成されている場合、「**サーバー**」 → 「**クラスター**」 → ***cluster_name*** をクリックします。
3. 「**ビジネス・インテグレーション**」の下で、「**Business Process Choreographer**」 → 「**Human Task Manager**」 → 「**追加プロパティー**」-「**カスタム・プロパティー**」をクリックします。
4. **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。
5. 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。

関連タスク

628 ページの『担当者照会結果の後処理を行うプラグインの作成およびインストール』
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戻します。担当者解決で戻される担当者照会の結果を変更するプラグインを作成することができます。例えば、ワークロード・バランシングを改善するために、既にワークロードが高いユーザーを照会結果から除去するプラグインを使用することがあります。

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、 115 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、 115 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、 116 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、 117 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、 122 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。

結果

これで、 171 ページの『第 4 章 Business Process Choreographer の構成』を実行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

153 ページの『Business Process Choreographer について』
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
     - Explorer
     - Observer および 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 と Business Process Choreographer Observer は構成されません。これらのコンポーネントには専用の構成ページがあります。また、スクリプトを実行することでも構成できます。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 と Business Process Choreographer Observer を構成できます。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。

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. 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 4. 構成パスの選択基準

実動システムに適しているかどうか	デプロイメント・ターゲット	Business Process Choreographer 構成	個別の BPEDB データベースを含めることができるかどうか	メッセージング・エンジン向けにサポートされているメッセージ・ストア	適切な構成名、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**
	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 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 4. 構成パスの選択基準 (続き)

実動システムに適しているかどうか	デプロイメント・ターゲット	Business Process Choreographer 構成	個別の BPEDB データベースを含めることができるかどうか	メッセージング・エンジン向けにサポートされているメッセージ・ストア	適切な構成名、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。
		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 は、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 (Users and Groups)**」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、Observer、および Common Event Infrastructure など) に専用のクラスターを設定するか、またはクラスターを共有するかどうかを決定します。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。

   これらのパターンについて詳しくは、『デプロイメント環境パターン』を参照してください。

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 メッセージング・エンジンが使用するメッセージ・ストアを決定します。

   - 専用のメッセージ・ストア (ファイル・ストアまたはデータベース・システムのいずれかにすることができます)。
   - セル内のすべてのメッセージング・エンジンが使用する別個の MEDB データベースを共用する。
   - WPRCSDB データベースを共用する。

g. Business Process Choreographer Observer を使用する場合は、Business Process Choreographer 構成を作成する際に同時に Observer を構成するか、または後で Observer を作成できます。Business Process Choreographer Observer が BPEDB データベースを使用するか、または専用の OBSRVRDB データベースを使用するかどうかを決定します。また、Business Process Choreographer

Observer のコンポーネントのトポロジーについても計画します。詳しくは、149 ページの『Business Process Choreographer Observer の計画』を参照してください。

5. WebSphere Portal から Business Process Choreographer にアクセスする場合は、ポータル・サーバーのベースになるアプリケーション・サーバーとバージョン番号が同じ WebSphere Process Server クライアントを使用して、ポータル・サーバーに Business Process Server クライアント・インストールを作成するよう計画します。 同様に、すべてのカスタム WebSphere Process Server クライアント・アプリケーションが Business Process Choreographer にアクセスできるように WebSphere Process Server クライアント・インストールを作成できます。
6. WebSphere Process Server クライアント・インストールで稼動するリモート Business Process Choreographer クライアント・アプリケーションが必要な場合は、『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。
7. 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。

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。

関連情報

プロファイル

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・インストールで稼動するリモート 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・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 WebSphere Process Server プロファイル・テンプレートは含まれておらず、基盤となる WebSphere Application Server プロファイルを拡張する必要はありません。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 Application Server クライアントのバージョンに一致する既存の WebSphere Application Server 上にインストールできます。例えば、WebSphere Portal Server 6.0.1 は、WebSphere Process Server 6.0.2 クライアント・インストールを必要とする WebSphere Application Server 6.0.2 を含みます。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、617 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 または Java Messaging Service (JMS) API – これらの API に基づくリモート・クライアント・アプリケーションは、WebSphere Process Server インストールを一切必要としません。
   - 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、448 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、ネーム・スペース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。

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。

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

107 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。 どちらの場合でも作成されるサンプルは同一です。唯一異なる点は、作成時に使用するツールです。
2. 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。
3. 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。

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

107 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**

   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 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 は、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 (Users and Groups)**」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。

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

107 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 ステップのコンテキスト・ルートを計画します。

   Business Process Choreographer Explorer コンテキスト・ルート
   これは、Business Process Choreographer Explorer に到達するためにブラウザーが使用する必要がある URL の一部を定義しています。

   Business Process Choreographer Observer コンテキスト・ルート
   これは、Business Process Choreographer Observer に到達するためにブラウザーが使用する必要がある URL の一部を定義しています。
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。
5. 担当者割り当てを使用する場合は、146 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。

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

107 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。
   - 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が参加する必要があります。また、この際データベース管理者には、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. 123 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、デプロイ先を計画します。
   b. Business Process Choreographer Event Collector 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、デプロイ先を計画します。
   c. Business Process Choreographer Observer 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、デプロイ先を計画します。
   d. SCA バインディングのコンテキスト・ルートを計画します。
   e. 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 - コンポーネント WBI_BPC。
   - Business Process Choreographer Observer 用の OBSRVRDB データベース - コンポーネント WBI_BPCEventCollector。
   - Business Process Choreographer メッセージング・エンジン用の BPEMEDB データベース - コンポーネント WBI_BPC_ME。

   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・インスタンス

デフォルト値 WPRCSDB の代わりに使用するデータ・ソースの名前 (例: BPEDB、OBSRVRDB、BPEMEDB など)。デフォルト値を使用すると、共通データベースが共用されます。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

使用するデータベース・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 5. 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

表 5. 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
DB2 UDB – Runtime Client	• ユーザー名 • パスワード • データベース名 • スキーマ名 • 説明 • テーブルの作成
DB2 for i5/OS – ネイティブ・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 • 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 • スキーマ名 • サーバー名 • 説明 • テーブルの作成
DB2 for z/OS and OS/390 V7、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	• ユーザー名 • パスワード • 説明 • テーブルの作成
Derby Embedded	• 説明 • テーブルの作成

表 5. 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 – 組み込みドライバーおよび DataDirect 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	• ユーザー名 • パスワード • サーバー名 • サーバー・ポート番号 • ifxIFXHOST • Informix ロック・モード待機 • 説明 • テーブルの作成
Oracle 10g および Oracle 9i	• 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 または oci8 • 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、130 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 ステップのコンテキスト・ルートを計画します。

   Business Process Choreographer Explorer コンテキスト・ルート
   これは、Business Process Choreographer Explorer に到達するためにブラウザーが使用する必要がある URL の一部を定義しています。

   Business Process Choreographer Observer コンテキスト・ルート
   これは、Business Process Choreographer Observer に到達するためにブラウザーが使用する必要がある URL の一部を定義しています。
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。

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. 担当者割り当てを使用する場合は、146 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。

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

107 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。
   - 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が参加する必要があります。また、この際データベース管理者には、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. 『セキュリティー、ユーザー ID、および許可の計画』
4. 130 ページの『Business Process Choreographer のデータベースの計画』
5. 145 ページの『Business Flow Manager および Human Task Manager の計画』
6. 146 ページの『担当者ディレクトリー・プロバイダーの計画』
7. 148 ページの『Business Process Choreographer Explorer の計画』
8. 149 ページの『Business Process Choreographer Observer の計画』
9. 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。構成ページの値について、 158 ページの『Business Process Choreographer 構成』で説明します。
10. 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 190 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 198 ページの『bpeconfig.jacl スクリプト・ファイル』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。

結果

これで、 171 ページの『第 4 章 Business Process Choreographer の構成』を実行するために必要なすべての事項の計画が完了しました。

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。

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 6. 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、198 ページの『bpeconfig.jacl スクリプト・ファイル』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 7. 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 のデータベース。 **OBSRVDB**: Business Process Choreographer Observer のデータベース。	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: • Windows プラットフォームの場合: *profile_root*¥dbscripts¥ProcessChoreographer¥ • Linux、UNIX、i5/OS、および z/OS プラットフォームの UNIX System Services (USS) の場合: *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。 • Windows プラットフォームの場合: *install_root*¥dbscripts¥ProcessChoreographer¥*database_type* • Linux、UNIX、i5/OS、および z/OS プラットフォームの UNIX System Services (USS) の場合: *install_root*/dbscripts/ProcessChoreographer/*database_type* ここで、*database_type* は以下のいずれかになります。 • DB2 • DB2zOSV7 • DB2zOSV8 (DB2 for z/OS V8 および V9 の場合) • Db2iSeries • Derby • Informix • Oracle • SQLServer	

表 7. install_root のサブディレクトリーへのアクセスの計画 (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
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。 • Windows プラットフォームの場合: *install_root*¥ProcessChoreographer¥Staff • Linux、UNIX、および i5/OS プラットフォームの場合: *install_root*/ProcessChoreographer/Staff Integration 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 8. 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・インスタンスを作成します。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。	
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。	

表 8. *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
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 Observer を構成する場合は、Event Collector がデータベースを作成、構成、およびアクセスするときに使用するユーザー ID を計画します。

表 9. *OBSRVRDB 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	OBSRVRDB データベース・インスタンスを作成します。Oracle の場合は、OBSVRDB データベースを作成します。	データベースを作成します。	
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。	OBSRVRDB データベースの場合: テーブル変更、接続、関数作成、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。 ユーザー定義関数の Java 実装を使用する場合は、ユーザー ID に、JAR ファイルをインストールする権限も必要です。	

表 9. *OBSRVRDB 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
Event Collector データ・ソースのユーザー名	実行時	Observer データベースに接続します。Business Process Choreographer Observer を使用し、その Observer 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 10. *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 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
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 メッセージング・バスのデータ・ソースへの接続と、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) プロバイダーで使用するユーザー ID を計画します。

表 11. *JMS プロバイダーのユーザー 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
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。	

表 11. JMS プロバイダーのユーザー 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
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。	これらは、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 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、-jmsHTMRunAsUser パラメーターと -jmsHTMRunAsPwd パラメーターです。		

8. Business Flow Manager および Human Task Manager の J2EE 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 12. 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。	
管理者グループ	実行時		
モニター・ユーザー	実行時	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。 • -adminBFMUsers および -adminHTMUsers • -adminBFMGroups および -adminHTMGroups • -monitorBFMUsers および -monitorHTMUsers • -monitorBFMGroups および -monitorHTMGroups	
モニター・グループ	実行時		

9. 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 13. 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。	

10. 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 サーバーへのログオンで使用されるユーザー ID を計画します。

表 14. LDAP 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
LDAP プラグイン・プロパティー: 認証別名	実行時	例えば mycomputer/My LDAP Alias など、LDAP に接続するために簡易認証を使用する Lightweight Directory Access Protocol (LDAP) 担当者ディレクトリー・プロバイダーを構成する場合。このユーザー 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。	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。LDAP サーバーで匿名認証を使用する場合、このユーザー ID は不要です。	

11. 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。

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 Observer を使用する場合、Observer 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 OBSRVDB を使用するとパフォーマンスが改善されます。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、110 ページの表 4 を参照してください。

手順

1. 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』Business Process Choreographer 構成を作成する場合は、別の Derby 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 Observer でも使用されます。Business Process Choreographer メッセージング・エンジンの場合、デフォルトでは個別の Derby データベース (BPEME) が作成されますが、『組織付きサンプル』構成を使用する場合は、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 Observer、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。
2. 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、132 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 Observer 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、138 ページの『OBSRVRDB データベースの計画』の説明に従い、個別のデータベース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. 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してパフォーマンスを改善するには、143 ページの『メッセージング・エンジン・データベースの計画』を行います。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。

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。
   - DB2 UDB for Linux、UNIX、および Windows
   - DB2 for iSeries™
   - DB2 for z/OS®
   - Derby
   - Informix Dynamic Server
   - Microsoft® SQL Server
   - Oracle

   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、110 ページの表 4を参照してください。
2. 計画を最小限に抑え、次の特性を備えた**単純**データベース・セットアップを使用する場合:
   - 一般に実動システムで必要とされるパフォーマンス、スケーラビリティー、およびセキュリティーは実現されません。
   - すべてのデータベース・オブジェクトが 1 つのテーブル・スペース (デフォルトのテーブル・スペースなど) に作成されます。
   - データベース・サーバーは、WebSphere Process Server マシンに対してローカルです。
   - データベースへアクセスするためのユーザー ID には管理者権限も付与されます。

   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。

- **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。
- 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。このスクリプトでは、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ます。
- **bpeconfig.jacl** ツールでは、Business Process Choreographer と、すべてのデータベースの JDBC アクセスを構成できます。
  - 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。Oracle データベースをご使用の場合、データベースが既に作成されている必要があります。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。その他のタイプのデータベースでは、bepconfig.jacl はデータベース・インスタンスの作成を試行します。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、198 ページの『bpeconfig.jacl スクリプト・ファイル』を参照してください。
- 管理コンソールの「**Business Process Choreographer の構成**」**ページ**では、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・インスタン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・オブジェクトをデータベース内に作成します。このオプションは、DB2 for 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。

– 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。

3. Business Process Choreographer に対し次の特性を備えた**ハイパフォーマンス・データベース・セットアップ**を使用する場合:
   - 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・サーバーは専用マシンにインストールされていることが理想的ですが、WebSphere Process Server マシンに対しローカルにすることもできます。
   - 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。

   次のすべてのステップについて計画する必要があります。
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、 126 ページの表 8 の内容を実行します。
5. 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・テーブル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。次のいずれかを使用する場合は、以下に説明するようにします。
   - DB2
   - DB2zOSV7
   - DB2zOSV8
   - Informix (テーブル・スペースは名前付き dbspaces と呼ばれています)
   - Oracle

   次に示す BPEDB データベース・テーブル・スペースの配置先を計画します。
   - AUDITLOG
   - COMP
   - INDEXTS
   - INSTANCE
   - LOBTS
   - STAFFQRY

- TEMPLATE
- WORKITEM
- SCHEDTS

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、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 64 KB の RAID-1 アレイを使用します。

DB2 データベースを使用しており、RAID コントローラーで 15 台のディスク・ドライブを使用する予定の場合は、以下の割り振りを検討します。

- オペレーティング・システムおよびページング (Windows ではページ・ファイル、AIX® および HP-UX ではページング・スペース、Solaris ではスワップ・スペースと呼ばれている) に対しディスク 1 台を使用します。
- RAID-1 構成の 8 つのディスク (4 つのプライマリー・ディスクと 4 つのミラー) を、データベース制御ファイルおよびテーブル・スペースのための 1 つの論理ディスクとして使用し、ストライプ・サイズ 256 KB を使用します。
- RAID-1 構成の 6 つのディスク (3 つのプライマリー・ディスクと 3 つのミラー) を、データベース・トランザクション・ログのための 1 つの論理ディスクとして使用し、ストライプ・サイズ 64 KB を使用します。

Oracle データベースを使用している場合は、次のガイドラインを検討します。

- ストライプ幅 1 MB を使用して、すべてのディスクの全ファイルの内容をストライプおよびミラーリング (SAME) します。
- 高可用性を実現するためデータをミラーリングします。
- ディスク・ドライブの外側半分に (テーブル・スペースの) 区画を作成します。
- ディスクではなく区画ごとにデータ・サブセットを作成します。
- Automatic Storage Management (ASM) ファイル・システムを使用します。
- 個々の REDO ログを他のデータ・ファイルから分離しないでください。

6. 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、220 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。

- Windows プラットフォームの場合: *install_root*¥dbscripts¥ProcessChoreographer¥*database_type*
- Linux、UNIX、i5/OS、および z/OS プラットフォームの UNIX System Services (USS) の場合: *install_root*/dbscripts/ProcessChoreographer/*database_type*

ここで、*database_type* は以下のいずれかになります。

- DB2
- DB2zOSV7
- DB2zOSV8 (DB2 for z/OS V8 および V9 の場合)
- Db2iSeries
- Derby
- Informix
- Oracle
- SQLServer

7. データベース・サーバーがプロセス・サーバー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・クライアントをプロセス・サーバー・マシン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。
8. 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。

9. 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。
10. 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。
11. 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。Oracle の場合は、oci または thin ドライバーのどちらを使用するかを決定します。
    - データベース・インスタンス (Oracle の場合はデータベース名、DB2 for z/OS の場合はサブシステム名)。
    - スキーマ修飾子。デフォルトでは、暗黙的なスキーマ修飾子として接続ユーザー ID が使用されます。
    - スキーマを作成するユーザーのユーザー名。
    - タイプ 4 JDBC ドライバーを使用している場合は、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。

    デフォルト値を含むこれらのパラメーターの詳細については、158 ページの『Business Process Choreographer 構成』または 198 ページの『bpeconfig.jacl スクリプト・ファイル』の該当する説明を参照してください。
12. 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 活動化仕様 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。
13. 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。

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。

関連タスク

681 ページの『ハードウェア・リソースの平衡化』
ハードウェア・リソースを平衡化することにより、長期実行ビジネス・プロセスのパフォーマンスを改善します。

OBSRVRDB データベースの計画

Business Process Choreographer Observer 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 Observer は同じデータベースを使用できますが、データベース OBSRVRDB 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。
   - Derby
   - DB2 UDB for Linux、UNIX、および Windows
   - DB2 for iSeries
   - DB2 for z/OS
   - Oracle

   制約事項: Business Process Choreographer Observer では、Informix または SQL Server データベースの使用はサポートされていません。
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、 127 ページの表 9の内容を実行します。
5. Observer と Event Collector が BPEDB データベースを使用するように構成するときに bpeconfig.jacl スクリプトを使用しない場合は、OBSRVRDB 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、 299 ページの『setupEventCollector ツール』を参照してください。

このツールでは、Java ベースのユーザー定義関数 (UDF) または SQL ベースの UDF の作成、これらの 2 つのオプションの切り替え、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または削除を行うことができます。Derby 以外のデータベースの場合、このツールでは、Java ベースのユーザー定義関数 (UDF) または SQL ベースの UDF のいずれかを使用してデータベースを作成できます。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。Derby データベース以外の場合、すべての SQL スクリプトで、SQL 実装を使用して Business Process Choreographer Observer の UDF が作成されます。これにより、SQL ベースの UDF を使用した構成作業が簡素化されます。 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。

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・スキーマを容易に作成する方法として、管理コンソールの Business Process Choreographer 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。Derby 以外のデータベースの場合、SQL ベースの UDF が使用されます。これにより、構成作業が簡素化されます。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: 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、 280 ページの『Business Process Choreographer Observer のユーザー定義関数』を参照してください。

6. DB2 for Linux, UNIX, or Windows データベースを使用している場合は、以下を計画します。
   - データベース名。デフォルト値は `BPEDB` 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・オブジェクトの作成に使用するデータベース・スキーマ名。デフォルト値は接続ユーザー ID です。
   - テーブル・スペース OBSVRTS の完全修飾ロケーションを計画します。
   - デフォルトの Java ベースのユーザー定義関数 UDF ではなく SQL ベースの UDF を使用するかどうかを決定します。

- 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 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)。これはデフォルトです。
    - 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 `50000` です。
  - DB2 JDBC ドライバー・ファイル db2jcc.jar と db2jcc_license_cu.jar がインストールされているディレクトリーを見つけます。

7. DB2 for i5/OS データベースを使用している場合は、以下を計画します。
   - データベース名。ネイティブ i/Series 環境 (q シェルなど) でデータベースを構成する場合は、`*LOCAL` を使用します。それ以外の場合は、`*SYSBAS` を使用します。
   - 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。デフォルト値は接続ユーザー ID です。
   - デフォルトの Java ベースのユーザー定義関数 UDF ではなく SQL ベースの UDF を使用するかどうかを決定します。
   - 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。
     - データベース・サーバーのホスト名。一般に、これは常に `localhost` です。ポート番号は常に 446 です。
     - JDBC ドライバーのディレクトリー:
       - ネイティブ i/Series 環境 (q シェルなど) 内にデータベースがある場合、db2_classes.jar ファイルが含まれているパスは、通常 /QIBM/ProdData/Java400/ext です。
       - データベースがリモート・データベースである場合、このパスは jt400.jar ファイルが含まれているパスです。
8. 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)。
   - ご使用の DB2 のバージョンに応じて、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。

– DB2 for z/OS バージョン 7 では、以下のテーブル・スペースを使用します。
  - OBSVRTS の場合は通常のテーブル・スペース。
  - OS26201、OS26202、OS26203、および OS26204 の場合はラージ・オブジェクト (LOB) テーブル・スペース。
– DB2 for z/OS バージョン 8 では、以下のテーブル・スペースを使用します。
  - 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。

9. 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。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。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。

10. Oracle データベースを使用する場合は、以下を計画します。
    - SID 名。デフォルト値は `BPEDB` です。
    - データベースへの接続に使用する Oracle ユーザー ID を決定します。このユーザー ID には、ロール CONNECT と RESOURCE が必要です。デフォルトのユーザー ID は `system` です。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。デフォルト値は、データベースへの接続に使用するユーザー ID です。
    - 次のテーブル・スペースの完全修飾ロケーションを計画します。
      - OBSVRIDX
      - OBSVRLOB
      - OBSVRTS
    - デフォルトの Java ベースのユーザー定義関数 UDF ではなく SQL ベースの UDF を使用するかどうかを決定します。
    - 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。
      - JDBC ドライバー・ファイルのロケーションは ojdbc14.jar 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 `1521` です。
11. 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が、Business Process Choreographer Observer は、同じデータベース内の同じスキーマを使用します。このため、ハイパフォーマンス・システムでは -createDB yes オプションは適していません。
    - -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、 190 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。

制約事項: 対話モードで bpeconfig.jacl スクリプトを使用して Business Process Choreographer Observer を構成することはできません。

12. 管理コンソールの「**Business Process Choreographer Event Collector**」ページでデータベース・テーブルを作成する場合は、次のいずれかを計画します。
    - DB2 for z/OS 以外のすべてのデータベース・タイプでは、「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・テーブルを作成する場合は、「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。
13. 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。
    - Windows プラットフォームの場合: *install_root*¥dbscripts¥ProcessChoreographer¥ *database_type*
    - Linux、UNIX、i5/OS、および z/OS プラットフォームの UNIX System Services (USS) の場合: *install_root*/dbscripts/ProcessChoreographer/*database_type*

    ここで、*database_type* は以下のいずれかになります。
    - DB2
    - DB2zOSV7
    - DB2zOSV8 (DB2 for Z/OS V8 および V9 の場合)
    - Db2iSeries
    - Derby
    - Oracle

    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。

結果

これで、Business Process Choreographer Observer の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

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スタンドアロン・サーバー上で Business Process Choreographer を構成する場合は、メッセージ・ストアに使用するコンポーネントを決定します。
   - SCA のメッセージング・エンジン・データベースの個別スキーマ
   - 別の Derby データベース
   - bpeconfig.jacl スクリプトの実行前にデプロイメント・ターゲットで作成されたユーザー定義データ・ソース
   - ファイル・ストア
5. 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、次の構成パラメーターを計画します。これらのパラメーターについて詳しくは、 164 ページの『バス』 (「Business Process Choreographer の構成」内) セクションを参照してください。
   - 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、128 ページの表 10 で計画したユーザー ID です。

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 128 ページの表 11 では、「**JMS API 認証ユーザー**」として知られます。
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 128 ページの表 11 では、「**エスカレーション・ユーザーの認証ユーザー**」として知られます。
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、 129 ページの表 12 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. 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。
7. Business Process Choreographer Observer 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。

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方法についての詳細は、『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 (147 ページ) 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 (ファイル・ベースのフェデレーテッド・ユーザー・リポジトリー)

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。

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、227 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。この別名が設定されていない場合は、LDAP サーバーへの匿名ログオンが使用されます。
AuthenticationType	オプション	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。
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。
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、225 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。

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、Business Process Choreographer 構成の作成時に Explorer を構成するか、または後で Explorer を構成することができます。

手順

1. 構成する Business Process Choreographer Explorer インスタンスの数を決定します。Business Process Choreographer を構成する間に、最初のインスタンスを容易に作成できます。 複数のインスタンスを作成する理由としては、以下のものがあります。
   - 各 Business Process Choreographer Explorer インスタンスで管理できる Business Process Choreographer 構成が 1 つのみであるため、環境内で複数の Business Process Choreographer 構成を使用している場合、構成ごとに Business Process Choreographer Explorer インスタンスをセットアップしておくことが適切です。
   - Business Process Choreographer Explorer をカスタマイズする場合は、複数のカスタマイズされた Business Process Choreographer Explorer から 1 つの Business Process Choreographer 構成を管理することがあります。各バージョンは個別にカスタマイズできます。カスタマイズ可能な項目について詳しくは、367 ページの『Business Process Choreographer Explorer のカスタマイズ』を参照してください。

   さらにインスタンスが必要な場合は、インスタンスを個別に作成する必要があります。このとき、167 ページの『Business Process Choreographer Explorer 設定』の説明に従って管理コンソールを使用するか、または

239 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』の説明に従って構成スクリプトを使用します。

2. 必要とする Business Process Choreographer Explorer インスタンスごとに、次の項目を計画します。
   - 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。
   - エスカレーション E メールに挿入される Business Process Choreographer Explorer の URL。

     同じ Business Process Choreographer 構成を複数の Business Process Choreographer Explorer で管理している場合、その Business Process Choreographer 構成の Human Task Manager は、 Business Process Choreographer Explorer インスタンスのうちの 1 つのみにリンクできます。この場合、Human Task Manager が使用する Business Process Choreographer Explorer URL を決定する必要があります。
   - 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。
   - 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)。

   これらの構成パラメーターについて詳しくは、 167 ページの『Business Process Choreographer Explorer 設定』を参照してください。

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 Observer の計画

Business Process Choreographer Observer と Event Collector の構成を計画します。

このタスクについて

Business Process Choreographer Observer を使用する場合は、Business Process Choreographer 構成の作成時に Observer を計画または構成できます。あるいは、Observer の計画と構成を後で行うこともできます。

手順

1. 各種 Business Process Choreographer Observer 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。

   Observer アプリケーション。
   サーバーまたはクラスター上で複数のインスタンスを構成できます。インスタンスは、Business Process Choreographer が構成されているデプロイメント・ターゲット上に存在している必要はありません。各インスタンスは、データ・ソースを介して 1 つのデータベース・スキーマに接続します。

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。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 OBSRVRDB データベースに書き込みます。

OBSRVRDB データベース。
Event Collector と Observer 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、155 ページの『Business Process Choreographer Observer について』を参照してください。

2. セットアップの目的、マシンの要件、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 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、Observer および 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。Observer アプリケーションのインスタンスを、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。

3. Observer のデータベースについてまだ計画していない場合は、138 ページの『OBSRVRDB データベースの計画』を行います。
4. 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、285 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、282 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・システムに対して Business Process Choreographer を構成するときには bpeconfig.jacl を使用しないでください。これは、bpeconfig.jacl では Event Collector アプリケーションと Observer アプリケーションが、Business Process Choreographer 構成と同じデプロイメント・ターゲットで、BPEDB データベースを共有するように構成されるためです。このオプションについて詳しくは、 190 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。

Business Process Choreographer Observer を対話モードで構成する場合は、bpeconfig.jacl を使用できません。

c. データ・ソースを計画します。
   - セキュリティーが重要である場合:
     - Business Process Choreographer Observer 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、Observer 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。
   - セキュリティーが重要ではない場合:
     - Business Process Choreographer Observer が Business Process Choreographer と同じ物理データベース (BPEDB) を共有している場合は、同じデータ・ソースを使用できます。つまり、同じ JNDI 名を使用できます。
     - 同じデータ・ソースを使用する場合は、同じ認証別名も使用します。
   - 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。
   - 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 db2admin 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、294 ページの『Business Process Choreographer Observer の構成パラメーターの変更』を参照してください。

5. 構成する Business Process Choreographer Observer インスタンスごとに、次の項目を計画します。

- このインスタンスの構成方法を決定します。
  - 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、291 ページの『管理コンソールを使用した Business Process Choreographer Observer の構成』を参照してください。
  - 対話式ツール setupObserver を使用する。このオプションについて詳しくは、289 ページの『setupObserver ツールを使用した Business Process Choreographer Observer 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。`-createObserver` 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・システムに対して Business Process Choreographer を構成するときには bpeconfig.jacl を使用しないでください。これは、bpeconfig.jacl では Event Collector アプリケーションと Observer アプリケーションが、Business Process Choreographer 構成と同じデプロイメント・ターゲットで、BPEDB データベースを共有するように構成されるためです。このオプションについて詳しくは、190 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。
- このインスタンスのデプロイメント・ターゲット。
- このインスタンスがモニターする Event Collector により使用されるデータ・ソースの JNDI 名。
- 管理コンソールを使用してこのインスタンスを構成する場合は、以下についても計画します。
  - このインスタンスのコンテキスト・ルート。これにより、ブラウザーでこの Business Process Choreographer Observer インスタンスにアクセスするときに使用される URL の一部が指定されます。デフォルトは `/bpcobserver` です。

– この Business Process Choreographer Observer インスタンスによってデータが視覚化される Event Collector インスタンスのデプロイメント・ターゲット。

これらの構成パラメーターについて詳しくは、 168 ページの『Business Process Choreographer Observer 設定』を参照してください。

6. 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、Business Process Choreographer 構成と同じデプロイメント・ターゲットで Event Collector と Observer アプリケーションも構成されます。
   - bpeconfig.jacl で Event Collector または Observer アプリケーションのいずれかあるいは両方を構成しない場合は、bpeconfig.jacl のオプション `-createEventCollector no` と `-createObserver no` のいずれかまたは両方を使用するように計画します。これらのオプションを指定すると、bpeconfig.jacl がこれらのアプリケーションを構成しません。

結果

これで、Business Process Choreographer Observer と Event Collector の構成オプションの計画が完了しました。

Business Process Choreographer について

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 アプリケーションは、ビジネス・プロセスとヒューマン・タスクの管理機能を提供します。 詳しくは、『Business Process Choreographer Explorer について』を参照してください。
- Business Process Choreographer Observer。この Web アプリケーションによって、ビジネス・プロセスの実行状態を監視できます。. 詳しくは、 155 ページの『Business Process Choreographer Observer について』を参照してください。

関連タスク

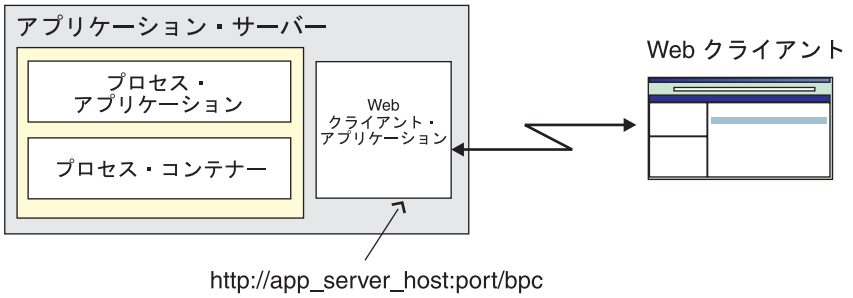
Business Process Choreographer の構成計画
Business Process Choreographer のセットアップと構成パラメーターを計画します。

Business Process Choreographer Explorer について

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

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 プロファイルを使用した WebSphere Process Server インストールを行うか、WebSphere Process Server クライアント・インストールを実行すれば十分です。Business Process Choreographer をサーバーまたはクラスター上で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 Observer について

Business Process Choreographer Observer について説明します。

Business Process Choreographer Observer を使用して、完了した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 Observer は、Common Event Infrastructure (CEI) を使用して、WebSphereProcess Server が発行す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 Observer は、以下の図に示す 2 つの J2EE エンタープライズ・アプリケーションに基づいています。

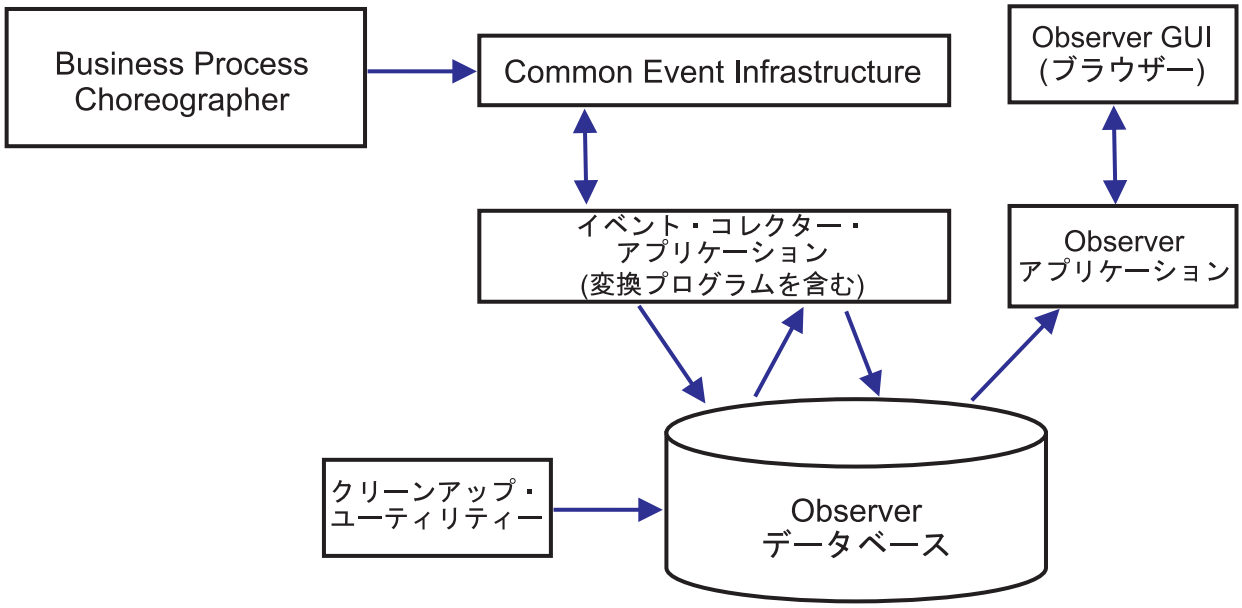


図 1. Business Process Choreographer Observer のアーキテクチャー

- Event Collector の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、Business Process Choreographer Observer データベースの Event Collector 表に保管します。

- Observer データベースは、イベント・データを保管するデータベース表の集合です。
- イベント変換プログラムが定期的にトリガーされ、これにより未加工のイベント・データは、Business Process Choreographer Observer からの照会に適したフォーマットに変換されます。
- Observer のアプリケーションはレポートを生成し、ユーザーが Observer のグラフィカル・ユーザー・インターフェース (GUI) を使用して開始可能な他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・ユーティリティーを使用して Observer データベースからレコードを削除し、これによりパフォーマンスを改善することができます。

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。

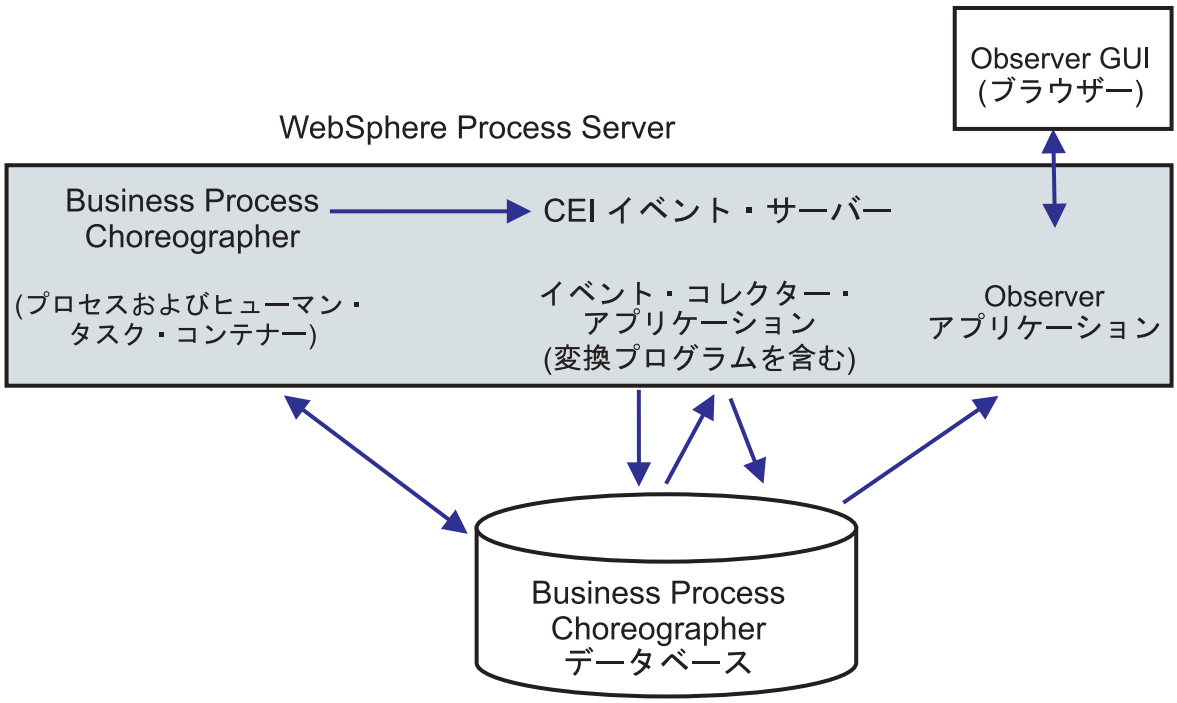


図 2. Business Process Choreographer Observer のスタンドアロン・セットアップ

すべての構成要素が 1 台のマシンにインストールされており、Business Process Choreographer と Business Process Choreographer Observer 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・ターゲット上で Observer アプリケーションと Event Collector アプリケーションが構成されます。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

組み込まれている対話式構成ツールでは、Business Process Choreographer Observer アーキテクチャーの潜在能力を最大限に活用することができます。例えば、実動システムでのパフォーマンスを向上させる理想的な構成は、3 つの Business Process Choreographer エレメントがそれぞれ個別のマシンで稼働し、Business Process Choreographer Observer に独自のデータベースが用意されている構成です。

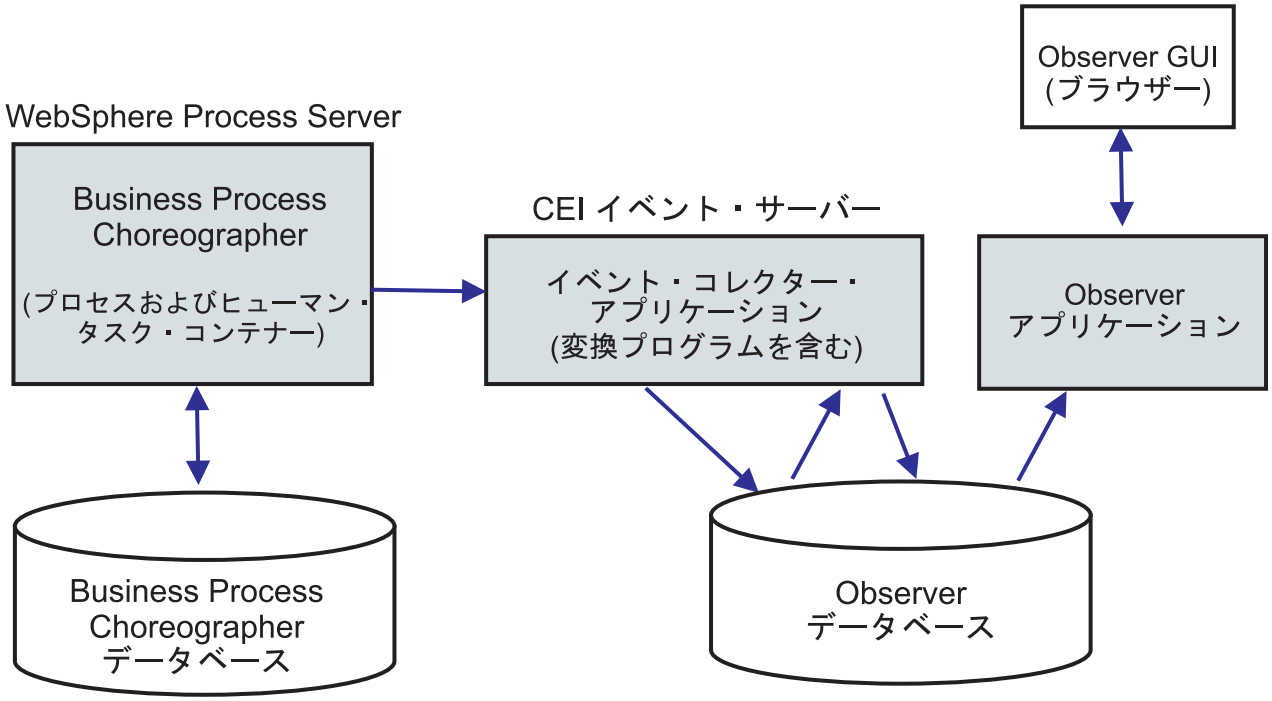


図 3. 実動パフォーマンスを考慮した Business Process Choreographer Observer のセットアップ

クラスター構成のセットアップにおいて、またはより複雑なデータベース・オプションを使用するために、Business Process Choreographer Observer 用に別個のデータベースを使用するか、または Business Process Choreographer Observer を既存の Business Process Choreographer 構成に追加する場合は、242 ページの『Business Process Choreographer Observer の構成』を実行します。

Network Deployment 環境

以下の制約は、Network Deployment 環境で Business Process Choreographer Observer を構成する場合に適用されます。

- 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 Observer アプリケーションを Event Collector と同じマシンにインストールする必要はありません。

Business Process Choreographer 構成

このパネルを使用して、Business Process Choreographer をインストールおよび構成します。

この管理コンソール・ページを表示するには、「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」 または「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックし、「**コンテナー**」の下で「**Business Process Choreographer Container**」をクリックします。

このページはいくつかのセクションに分かれています。各セクションのフィールドについて詳しくは、次の項目を参照してください。

- 『データ・ソース』
- 160 ページの『Human Task Manager のメール・セッション』
- 161 ページの『セキュリティー』
- 163 ページの『状態監視者』
- 164 ページの『SCA バインディング』
- 164 ページの『バス』

データ・ソース

このセクションでは、Business Process Choreographer のデータ・ソースを指定します。

編集

データ・ソースのデータベース固有設定を変更する場合は、これをクリックします。

セットアップによっては、デフォルトを変更する必要があります。例えば、DB2 for z/OS データベースを使用する場合は、以下のようにします。

- ストレージ・グループまたは接続プールの設定を変更します。
- Linux、Windows、UNIX、または i5/OS プラットフォームで DB2 for z/OS データベースを使用するように Business Process Choreographer を構成する場合は、実装タイプに XA データ・ソースを選択する必要があります。

テスト接続

データ・ソースへの接続をテストします。

データベース・インスタンス

Business Flow Manager と Human Task Manager に使用されるデータベースの名前です。

プロパティー	**値**
データ型	ストリング

プロパティー	値
デフォルト	WPRCSDB

スキーマ名

使用するスキーマの名前。

デフォルトのスキーマの代わりに独自のスキーマを使用する場合、スキーマ名を変更するだけで済みます。

プロパティー	値
データ型	ストリング

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には、データベースが存在する必要があり、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。

実動システムでは、このオプションの使用は推奨されません。このオプションを選択しない場合、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。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

ユーザー名

データベースに接続し、データを変更する権限を持つユーザー ID。

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。

プロパティー	値
データ型	ストリング

パスワード

データ・ソースのユーザー ID のパスワード。

プロパティー	値
データ型	ストリング

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 アドレスと、ポート番号を指定します。

プロパティー	値
データ型	ストリング
例	localhost:50000

プロバイダー

Business Process Choreographer の JDBC プロバイダー。

プロパティー	値
データ型	ドロップダウン・リスト

Human Task Manager のメール・セッション

このセクションでは、エスカレーション E メールのパラメーターを指定します。

E メール・サービスを使用可能にする

Human Task Manager がエスカレーションの通知を E メールで送信する場合は、メール・セッションを使用可能にする必要があります。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

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。

プロパティー	値
データ型	ストリング

メール・トランスポートのユーザー

E メール・サービスのユーザー ID。

メール・サーバーで認証が不要な場合は、このフィールドを空白のままにすることができます。

プロパティー	値
データ型	ストリング

メール・トランスポートのパスワード

メール・トランスポート・ユーザー ID のパスワード。

メール・サーバーで認証が不要な場合は、このフィールドを空白のままにすることができます。

プロパティー	値
データ型	ストリング

Business Process Choreographer Explorer の URL

E メールでの Business Process Choreographer Explorer へのリンクに使用する URL を指定します。

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。

プロパティー	値
データ型	ストリング
例	http://www.ibm.com:9080/bpc

セキュリティー

このセクションでは、機能役割からユーザー ID とグループへのマッピング、および Business Process Choreographer で必要な認証資格情報を指定します。

管理者ユーザー

管理者のセキュリティーの役割は、指定したユーザー ID にマップされます。

プロパティー	値
データ型	ストリング
デフォルト	現在ログオンしているユーザー

管理者グループ

管理者のセキュリティーの役割は、指定したグループにマップされます。

プロパティー	値
データ型	ストリング
デフォルト	なし

モニター・ユーザー

システム・モニターのセキュリティーの役割は、指定したユーザー ID にマップされます。

プロパティー	値
データ型	ストリング
デフォルト	現在ログオンしているユーザー

モニター・グループ

システム・モニターのセキュリティーの役割は、指定したグループにマップされます。

プロパティー	値
データ型	ストリング
デフォルト	なし

JMS 認証ユーザー

システム統合バスの認証別名。

プロパティー	値
データ型	ストリング
デフォルト	現在ログオンしているユーザー

JMS 認証パスワードおよび確認パスワード

JMS 認証ユーザー ID のパスワード。

プロパティー	値
データ型	ストリング
デフォルト	なし

JMS API 認証ユーザー

Business Flow Manager メッセージ駆動型 Bean の run-as ユーザー ID。

プロパティー	値
データ型	ストリング
デフォルト	現在ログオンしているユーザー

JMS API 認証パスワードおよび確認パスワード

JMS API 認証ユーザー ID のパスワード。

プロパティー	値
データ型	ストリング
デフォルト	なし

エスカレーション・ユーザーの認証ユーザー

Human Task Manager メッセージ駆動型 Bean の run-as ユーザー ID。

プロパティー	値
データ型	ストリング
デフォルト	現在ログオンしているユーザー

エスカレーション・ユーザーの認証パスワードおよび確認パスワード

エスカレーション・ユーザーの認証ユーザー ID のパスワード。

プロパティー	値
データ型	ストリング

プロパティー	値
デフォルト	なし

状態監視者

このセクションでは、Business Flow Manager と Human Task Manager に対して、監査ロギングおよび Common Event Infrastructure (CEI) ロギングを使用可能に設定できます。

Business Flow Manager の監査ロギング

このオプションを選択すると、Business Flow Manager の監査ロギングが使用可能になります。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

Human Task Manager の監査ロギング

このオプションを選択すると、Human Task Manager の監査ロギングが使用可能になります。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

Business Flow Manager の Common Event Infrastructure ロギング

このオプションを選択すると、Business Flow Manager の Common Event Infrastructure ロギングが使用可能になります。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

Human Task Manager の Common Event Infrastructure ロギング

このオプションを選択すると、Human Task Manager の Common Event Infrastructure ロギングが使用可能になります。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

SCA バインディング

Service Component Architecture (SCA) バインディングでは、Web サービス API のコンテキスト・ルートを設定できます。

ホスト

この読み取り専用フィールドには、コンテキスト・ルートが付加される Business Flow Manager バインディングと Human Task Manager バインディングのホストのコンテキスト・プレフィックスが示されます。

Business Flow Manager のコンテキスト・ルート

Business Flow Manager Web サービスのコンテキスト・ルート。

プロパティー	**値**
データ型	ストリング
サーバーでの構成時デフォルト	/BFMIF_*nodeName_serverName*
クラスターでの構成時デフォルト	/BFMIF_*clusterName*

Human Task Manager のコンテキスト・ルート

Human Task Manager Web サービスのコンテキスト・ルート。

プロパティー	**値**
データ型	ストリング
サーバーでの構成時デフォルト	/HTMIF_*nodeName_serverName*
クラスターでの構成時デフォルト	/HTMIF_*clusterName*

相対パス

この読み取り専用フィールドには、Business Flow Manager と Human Task Manager の SCA バインディングの相対パスが表示されます。

プロパティー	**値**
データ型	読み取り専用ストリング
Business Flow Manager の相対パス	/sca/com/ibm/bpe/spi/sca/BFMWS
Human Task Manager の相対パス	/sca/com/ibm/task/spi/sca/HTMWS

バス

Business Process Choreographer メッセージング・エンジンでは、Service Component Architecture (SCA) で構成したものとは別のデータ・ソースを使用する場合は、このセクションを展開して設定を変更します。

デフォルト構成の使用

選択されている場合は、SCA メッセージング・エンジンの現行構成の設定が使用されます。

別の設定を使用する場合は、チェック・ボックスをクリアしてこのセクションの他のフィールドを有効にします。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

バス・メンバー・ロケーション

メッセージング・エンジンのデータをローカルまたはリモートのどちらで保管するかを決定します。

「**ローカル**」と「**リモート**」のいずれかを選択します。「**リモート**」を選択すると、リモート宛先のロケーション・セレクターと「**新規**」ボタンが使用可能になります。

プロパティー	値
データ型	ラジオ・ボタン
デフォルト	ローカル

リモート宛先ロケーション

リモート・メッセージング・エンジン・ストアのデプロイメント・ターゲットを指定します。

リストが空の場合、または選択するロケーションが含まれていない場合は、「**新規**」をクリックします。

プロパティー	値
データ型	ドロップダウン・リスト
デフォルト	なし

新規

このボタンにより、「デプロイメント・ターゲットの参照」ページが開きます。

デプロイメント・ターゲットを選択すると、そのターゲットがリモート宛先ロケーションのリストに追加されます。

編集

データ・ソースのデータベース固有設定を変更する場合は、これをクリックします。

セットアップによっては、デフォルトを変更する必要があります。例えば、DB2 for z/OS データベースを使用する場合は、以下のようにします。

- ストレージ・グループまたは接続プールの設定を変更します。
- Linux、Windows、UNIX、または i5/OS プラットフォームで DB2 for z/OS データベースを使用するように Business Process Choreographer を構成する場合は、実装タイプに XA データ・ソースを選択する必要があります。

テスト接続

データ・ソースへの接続をテストします。

データベース・インスタンス

データベースの名前。

プロパティー	値
データ型	ストリング
デフォルト	${USER_INSTALL_ROOT}¥databases¥BPEME

スキーマ名

使用するスキーマの名前。

プロパティー	値
データ型	ストリング
デフォルト	MEDBM00

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には、データベースが存在する必要があり、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。実動システムでは、このオプションにより作成されるデフォルトの表を使用しないようにすることがあります。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

ユーザー名

データベースに接続し、データを変更する権限を持つユーザー ID。

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。

プロパティー	値
データ型	ストリング

パスワード

データ・ソースのユーザー ID のパスワード。

プロパティー	値
データ型	ストリング

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 アドレスと、ポート番号を指定します。

プロパティー	値
データ型	ストリング
例	localhost:50000

プロバイダー

Business Process Choreographer メッセージング・エンジンの JDBC プロバイダー。

ファイル・ストアを使用するように SCA を構成している場合、このフィールドには `File Store` が設定されており、データベース・パラメーターのフィールドは使用不可になっています。データベース・プロバイダーが選択されると、データベース・パラメーターが使用可能になります。

プロパティー	値
データ型	ドロップダウン・リスト
デフォルト	SCA 向けに構成されているプロバイダー。

Business Process Choreographer Explorer 設定

一般プロパティーはここで設定できます。

この管理コンソール・ページを表示するには、「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックするか、または「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer**」、 「**Explorer の構成 (Explorer Configuration)**」をクリックします。新規のエクスプローラー構成を作成するには、「**追加**」をクリックします。既存の構成を表示するには、インスタンス名をクリックするか、またはインスタンスを選択して「**編集**」をクリックします。

コンテキスト・ルート

Business Process Choreographer Explorer のコンテキスト・ルートです。

これは、Business Process Choreographer Explorer に到達するためにブラウザーが使用する必要がある URL の一部を定義しています。

プロパティー	値
データ型	ストリング
デフォルト	/bpc

Explorer による検索結果の制限

照会に対して返される結果の最大数。

プロパティー	値
データ型	整数

プロパティー	値
単位	検索結果
デフォルト	10000

管理対象の Business Process Choreographer コンテナー

Business Process Choreographer コンテナーの有効範囲です。

これにより、このインスタンスの接続先 Business Process Choreographer Container を指定します。この接続は多対 1 の関係にすることができます。それぞれの Business Process Choreographer Explorer インスタンスは、確実に 1 つの Business Process Choreographer Container に接続できます。ただし、各 Business Process Choreographer Container は、どの Business Process Choreographer Explorer インスタンスにも接続しないようにすることも、複数のインスタンスに接続することも可能です。

プロパティー	値
データ型	デプロイメント・ターゲットのドロップダウン・リスト
デフォルト	この Business Process Choreographer Explorer インスタンスが構成されるデプロイメント・ターゲット (サーバーまたはクラスター)。

Business Process Choreographer Observer 設定

Business Process Choreographer Observer の特定のインスタンスに対して、コンテキスト・ルートとモニター・ターゲットを表示します。

この管理コンソール・ページを表示するには、「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックするか、または「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer**」、「**Observer の構成 (Observer Configuration)**」をクリックします。新規の Observer を作成するには、「**追加**」をクリックします。既存の構成を表示するには、インスタンス名をクリックするか、またはインスタンスを選択して「**編集**」をクリックします。

コンテキスト・ルート

Business Process Choreographer Observer のコンテキスト・ルートです。

これは、Business Process Choreographer Observer に到達するためにブラウザーが使用する必要がある URL の一部を定義しています。

プロパティー	値
データ型	ストリング
デフォルト	/bpcobserver

この Business Process Choreographer Event Collector からのモニター・データの視覚化

この Business Process Choreographer Observer インスタンスによって視覚化する予定のデータを持つ Event Collector インスタンスを選択します。

プロパティー	値
データ型	ドロップダウン・リスト
デフォルト	この Business Process Choreographer Observer インスタンスが構成されるデプロイメント・ターゲット (サーバーまたはクラスター)。

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・タスクを含むエンタープライズ・アプリケーションをインストールする前に、Business Process Choreographer を構成する必要があります。

始める前に

107 ページの『第 3 章 Business Process Choreographer の構成計画』が完了している。

このタスクについて

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。

- 以下の非実稼働構成パスの場合:
  - 『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』

  『インストーラーおよびプロファイル管理ツールを使用した Business Process Choreographer の構成』を実行します。
- 『実動デプロイメント環境』構成パスの場合は、 174 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での Business Process Choreographer の構成』を実行します。
- 『柔軟なカスタム構成』構成パスの場合は、使用するツールに応じて、以下のいずれかを実行します。
  - 177 ページの『管理コンソールの「Business Process Choreographer の構成」ページの使用』
  - 190 ページの『bpeconfig.jacl スクリプトを使用した Business Process Choreographer の構成』

結果

Business Process Choreographer が構成されます。

次のタスク

セットアップのカスタマイズを開始できます。

インストーラーおよびプロファイル管理ツールを使用した Business Process Choreographer の構成

非実動 Business Process Choreographer 構成を容易に作成する方法として、5 種類の方法があります。

始める前に

115 ページの『基本サンプル Business Process Choreographer 構成の作成の計画』を完了しており、使用する非実動システムのフレーバーを決定している (要約については 110 ページの表 4 を参照)。

手順

1. 選択した構成パスに応じて、ステップ 1a から 1c のいずれか 1 つを行います。
   a. 担当者の割り当てと代替のためのサンプル組織が含まれていない『基本サンプル』Business Process Choreographer 構成を使用する場合:
      1) インストーラーまたはプロファイル管理ツールを開始します。
         - インストーラー:
           – 「**標準的インストール**」オプションが選択されていることを確認します。
           – 「**スタンドアロン・サーバー**」オプションが選択されていることを確認します。
           – 「**管理セキュリティー**」を有効にしていることを確認します。
         - プロファイル管理ツール:
           – **WebSphere Process Server** プロファイルを作成していることを確認します。
           – 「**スタンドアロン・サーバー・プロファイル (Stand-alone server profile)**」オプションが選択されていることを確認します。
           – 「**標準**」プロファイル作成オプションが選択されていることを確認します。
           – 「**管理セキュリティーを使用可能にする**」が選択されていることを確認します。
   b. 『組織付きサンプル』Business Process Choreographer 構成 (担当者の割り当てと代替のための 15 名の担当者からなるサンプル組織が含まれている) を使用する場合:
      1) プロファイル管理ツールを開始します。
      2) **WebSphere Process Server** プロファイルを作成していることを確認します。
      3) 「**スタンドアロン・サーバー・プロファイル (Stand-alone server profile)**」オプションが選択されていることを確認します。
      4) 「**拡張**」オプションが選択されていることを確認します。
      5) 「**開発テンプレートからサーバーを作成する (Create server from development template)**」オプションが選択されていることを確認します。
      6) 「**管理セキュリティーを使用可能にする**」が選択されていることを確認します。
      7) 「**サンプル Business Process Choreographer の構成**」が選択されていることを確認します。
   c. デプロイメント環境パターンに基づいて『非実動デプロイメント環境』Business Process Choreographer 構成を使用する場合:
      1) インストーラーまたはプロファイル管理ツールを開始します。

- インストーラー:
  - 「**デプロイメント環境インストール**」オプションが選択されていることを確認します。
  - デプロイメント・マネージャーを作成することを確認します。
  - 次のいずれかのパターンに基づいて Business Process Choreographer を構成できます。
    - 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
  - 「**管理セキュリティー**」を有効にしていることを確認します。有効にしていないと、Business Process Choreographer 構成を取得できません。
- プロファイル管理ツール:
  - **WebSphere Process Server** プロファイルを作成していることを確認します。
  - 「**デプロイメント・マネージャー・プロファイル**」オプションが選択されていることを確認します。
  - 次のいずれかのパターンに基づいて Business Process Choreographer を構成できます。
    - 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
  - 「**管理セキュリティー**」を有効にしていることを確認します。有効にしていないと、Business Process Choreographer サンプルを取得できません。

2) カスタム・プロファイルを作成して統合します。

2. オプション: 310 ページの『Business Process Choreographer の作動確認』を実行します。
3. 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。
4. オプション: Human Task Manager の設定を変更します。
   - エスカレーション E メールの Human Task Manager 設定 (送信者のアドレス、Business Process Choreographer Explorer の URL プレフィックスなど) のいずれかを変更する場合は、「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスターで構成されている場合は「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer Container**」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またはポート番号、E メールのユーザー ID またはパスワードを変更するには、「**リソース**」 → 「**メール**」 → 「**メール・**

セッション」の順にクリックし、「**セル**」スコープを選択し、「**HTM メール・セッション**」をクリックして変更を加えます。

5. 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、 227 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、 225 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。
6. 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、 233 ページの『担当者の代替の構成』を行います。
7. 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**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスターで構成されている場合は「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer Container**」を展開し、「**Human Task Manager**」をクリックして「**グループ作業項目の使用可能化**」を選択します。
8. オプション: WebSphere Process Server クライアントを使用するリモート Business Process Choreographer クライアントを構成したい場合は、 305 ページの『リモート・クライアント・アプリケーションの構成』を実行します。

結果

Business Process Choreographer が構成されます。

管理コンソールのデプロイメント環境ウィザードでの Business Process Choreographer の構成

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して、Business Process Choreographer を含むパターン・ベースの構成を作成できます。Business Process Choreographer 構成に専用のデータベースがある場合、この構成は実動システムに適しています。

始める前に

117 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』を実行している。

手順

1. デプロイメント環境ウィザードを開始します。管理コンソールで「**サーバー**」 → 「**デプロイメント環境**」 → 「**新規**」をクリックします。他の構成パラメーターを入力する際に、 117 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』で計画した値を入力してください。

a. 次のいずれかのパターンに基づいて Business Process Choreographer を構成できます。
   - 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

b. セキュリティー・ページで、Business Process Choreographer の認証別名として使用するユーザー名とパスワードを設定できます。Business Process Choreographer はコンポーネント WBI_BPC として示されます。

c. Business Process Choreographer、Business Process Choreographer Explorer、または 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場合は、データベース・ページで、デフォルト・データ・ソースをデフォルト値から計画した値に変更します。

d. Business Process Choreographer ページで、この構成について計画したコンテキスト・ルート、セキュリティー・パラメーター、およびメール・セッション・パラメーターを指定します。

2. Business Process Choreographer に個別のデータベースを指定した場合は、 220 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』を行います。 個別のデータベースを指定せず、Derby データベースを使用しない場合は、空のデータベースが存在することを確認してください。これにより、Business Process Choreographer が初めてデータベースにアクセスするときに、データベース内にデフォルト・スキーマを作成できます。
3. Business Process Choreographer Observer に個別のデータベースを指定した場合は、 244 ページの『Business Process Choreographer Observer のためのデータベースの準備』を行います。 指定していない場合、Derby 以外のデータベースでは、空のデータベースが存在していることを確認してください。これにより、Business Process Choreographer が初めてデータベースにアクセスするときに、データベース内にデフォルト・スキーマを作成できます。
4. 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指定した場合は、そのデータベースが存在していることを確認します。
   - 「**テーブルの作成**」オプションを使用して、メッセージング・エンジンが初めてデータベースを使用するときにデフォルト・スキーマを作成するように設定するには、データベース・ユーザー ID に、使用する予定のスキーマ内にテーブルとビューを作成できる権限を付与します。
   - 「**テーブルの作成**」オプションを**使用しない**場合は、デフォルトのメッセージング・プロバイダーがデータベースにアクセスする前に、テーブルを作成します。*install_root* ディレクトリーの `bin` サブディレクトリーにある sibDDLGenerator ユーティリティーを使用すると、表を作成するときに使用できる DDL ファイルを生成できます。
5. Business Process Choreographer が構成されている各ノードで、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。
   - Derby の場合は、環境変数を設定する必要はありません。
   - DB2 for Linux, UNIX and Windows、または DB2 for z/OS で CLI ドライバーを使用している場合は、DB2_JDBC_DRIVER_PATH を選択します。
   - DB2 for Linux, UNIX and Windows、または DB2 for 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。
   - DB2 for i5/OS で Native ドライバーを使用している場合は、OS400_NATIVE_JDBC_DRIVER_PATH を選択します。
   - DB2 for i5/OS で Toolbox ドライバーを使用している場合は、OS400_TOOLBOX_JDBC_DRIVER_PATH を選択します。
   - Oracle の場合は、ORACLE_JDBC_DRIVER_PATH を選択します。
   - Informix の場合は、INFORMIX_JDBC_DRIVER_PATH を選択します。
   - SQL Server で WebSphere 埋め込み ConnectJDBC ドライバーを使用している場合は、環境変数を設定する必要はありません。
   - SQL Server で DataDirect ConnectJDBC タイプ 4 ドライバーを使用している場合は、CONNECTJDBC_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。

6. Business Process Choreographer をアクティブにします。 310 ページの『Business Process Choreographer の活動化』を実行します。
7. 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。 310 ページの『Business Process Choreographer の作動確認』を実行します。
8. オプション: Human Task Manager の設定を変更します。
   - エスカレーション E メールの Human Task Manager 設定 (送信者のアドレス、Business Process Choreographer Explorer の URL プレフィックスなど) のいずれかを変更する場合は、「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスターで構成されている場合は「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer Container**」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またはポート番号、E メールのユーザー ID またはパスワードを変更するには、「**リソース**」 → 「**メール**」 → 「**メール・セッション**」の順にクリックし、「**セル**」スコープを選択し、「**HTM メール・セッション**」をクリックして変更を加えます。
9. 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、 227 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、 225 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。

10. 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、 233 ページの『担当者の代替の構成』を行います。
11. 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**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスターで構成されている場合は「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer Container**」を展開し、「**Human Task Manager**」をクリックして「**グループ作業項目の使用可能化**」を選択します。
12. オプション: WebSphere Process Server クライアントを使用するリモート Business Process Choreographer クライアントを構成したい場合は、 305 ページの『リモート・クライアント・アプリケーションの構成』を実行します。
13. 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 Version 2 インバウンド認証の構成』を参照してください。

結果

これで、選択したデプロイメント環境向けに Business Process Choreographer が構成されました。

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。

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. デフォルト・プロファイルを作成したときに Business Process Choreographer サンプル構成オプションを選択した場合、Business Flow Manager、Human Task Manager、Business Process Choreographer Explorer、および Business Process Choreographer Observer は既に構成されています。

これらのコンポーネントが構成されているかどうかを確認するには、管理コンソールで、以下の文字列で始まる名前のエンタープライズ・アプリケーションの有無を確認します。

- BPCObserver
- BPCECollector
- BPEContainer
- BPCExplorer
- TaskContainer

サンプル構成は Derby データベースを使用しており、実動システムには適していません。Business Process Choreographer の構成はデプロイメント・ターゲットごとに 1 つに限られるため、Business Process Choreographer の構成を続行するには、315 ページの『第 5 章 Business Process Choreographer 構成の除去』で説明するようにサンプル構成を削除する必要があります。

2. Network Deployment 環境の場合、Service Component Architecture (SCA) が構成されていることを確認します。
   a. サーバーで Business Process Choreographer を構成するには、「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***serverName***」をクリックし、「**ビジネス・インテグレーション**」 セクションで「**Service Component Architecture**」をクリックします。
   b. クラスターで Business Process Choreographer を構成するには、「**サーバー**」 → 「**クラスター**」 → 「***clusterName***」をクリックし、「**ビジネス・インテグレーション**」セクションで「**Service Component Architecture**」をクリックします。
   c. 「**Service Component Architecture コンポーネントのサポート**」が有効になっていない場合は、これを選択します。
3. Business Process Choreographer メッセージング・エンジンのデータ・ストア用データベースを作成します。
   - 「Business Process Choreographer の構成」ページの「**テーブルの作成**」オプションを使用して、メッセージング・エンジンが初めてデータベースを使用するときにデフォルト・スキーマを作成するには、以下を実行します。
     a. データベースが存在していない場合は、作成してください。
     b. データベースのユーザー ID に、使用するスキーマにテーブルとビューを作成できる権限を付与します。
   - 「**テーブルの作成**」オプションを**使用しない**場合は、デフォルトのメッセージング・プロバイダーがデータベースにアクセスする前に、テーブルを作成します。*install_root* ディレクトリーの `bin` サブディレクトリーにある sibDDLGenerator ユーティリティーを使用すると、表を作成するときに使用できる DDL ファイルを生成できます。
4. Business Process Choreographer が構成されている各ノードで、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。
   - Derby の場合は、環境変数を設定する必要はありません。
   - DB2 for Linux, UNIX and Windows、または DB2 for z/OS で CLI ドライバーを使用している場合は、DB2_JDBC_DRIVER_PATH を選択します。
   - DB2 for Linux, UNIX and Windows、または DB2 for 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。
   - DB2 for i5/OS で Native ドライバーを使用している場合は、OS400_NATIVE_JDBC_DRIVER_PATH を選択します。
   - DB2 for i5/OS で Toolbox ドライバーを使用している場合は、OS400_TOOLBOX_JDBC_DRIVER_PATH を選択します。
   - Oracle の場合は、ORACLE_JDBC_DRIVER_PATH を選択します。
   - Informix の場合は、INFORMIX_JDBC_DRIVER_PATH を選択します。
   - SQL Server で WebSphere 埋め込み ConnectJDBC ドライバーを使用している場合は、環境変数を設定する必要はありません。
   - SQL Server で DataDirect ConnectJDBC タイプ 4 ドライバーを使用している場合は、CONNECTJDBC_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。

5. 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。 以下のいずれかをクリックします。
   - 「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***serverName***」
   - 「**サーバー**」 → 「**クラスター**」 → 「***clusterName***」

   ここで、*serverName* または *clusterName* は、サーバーまたはクラスターの名前です。

6. 「Business Process Choreographer の構成」ページに進みます。 「**コンテナー設定**」セクションで「**Business Process Choreographer Container の設定**」を展開し、「**Business Process Choreographer Container**」をクリックします。

7. Business Process Choreographer が構成されていないことを確認します。 Business Process Choreographer コンテナー (Business Flow Manager および Human Task Manager) が現在インストールされていないことを示すメッセージが表示されます。

   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、315 ページの『第 5 章 Business Process Choreographer 構成の除去』を行います。

8. 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。 詳しくは、158 ページの『Business Process Choreographer 構成』を参照してください。これは、以下のセクションで構成されています。
   a. 158 ページの『データ・ソース』
   b. 160 ページの『Human Task Manager のメール・セッション』
   c. 161 ページの『セキュリティー』

d. 163 ページの『状態監視者』
e. 164 ページの『SCA バインディング』
f. 164 ページの『バス』

9. 「**Apply**」をクリックします。 Business Process Choreographer のデプロイおよび構成の進行を報告する情報が表示されます。
10. インストールが正常に実行された場合は、「**マスター構成の保管 (Save Master Configuration)**」をクリックしてから「**保管**」をクリックします。 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。
11. データベース・スキーマを作成するには、ステップ 12 で Business Process Choreographer をアクティブにする前に、 220 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』で説明するアクションをユーザーまたはデータベース管理者が実行する必要があります。

    注: データベースがローカル・データベースであり、ステップ 9 (196 ページ) で Business Process Choreographer をアクティブにする時点でこのデータベースが既に作成されており、 220 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』で説明されているアクションを実行しない場合は、Business Process Choreographer が初めてデータベースにアクセスするときに、デフォルト・スキーマが作成されます。
12. Business Process Choreographer をアクティブにします。 310 ページの『Business Process Choreographer の活動化』を実行します。
13. 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。 310 ページの『Business Process Choreographer の作動確認』を実行します。
14. オプション: Human Task Manager の設定を変更します。
    - エスカレーション E メールの Human Task Manager 設定 (送信者のアドレス、Business Process Choreographer Explorer の URL プレフィックスなど) のいずれかを変更する場合は、「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスターで構成されている場合は「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer Container**」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またはポート番号、E メールのユーザー ID またはパスワードを変更するには、「**リソース**」 → 「**メール**」 → 「**メール・セッション**」の順にクリックし、「**セル**」スコープを選択し、「**HTM メール・セッション**」をクリックして変更を加えます。
15. 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、 227 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、 225 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。

16. 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、 233 ページの『担当者の代替の構成』を行います。
17. 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**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスターで構成されている場合は「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer Container**」を展開し、「**Human Task Manager**」をクリックして「**グループ作業項目の使用可能化**」を選択します。
18. 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 Version 2 インバウンド認証の構成』を参照してください。
19. 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。 238 ページの『Business Process Choreographer Explorer の構成』を実行します。
20. オプション: Business Process Choreographer Observer のインストールおよび構成が完了していない場合は、この時点で構成できます。 242 ページの『Business Process Choreographer Observer の構成』を実行します。
21. オプション: WebSphere Process Server クライアントを使用するリモート Business Process Choreographer クライアントを構成したい場合は、 305 ページの『リモート・クライアント・アプリケーションの構成』を実行します。

結果

Business Process Choreographer が構成されます。

Business Process Choreographer 構成

このパネルを使用して、Business Process Choreographer をインストールおよび構成します。

この管理コンソール・ページを表示するには、「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」 または「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックし、「**コンテナー**」の下で「**Business Process Choreographer Container**」をクリックします。

このページはいくつかのセクションに分かれています。各セクションのフィールドについて詳しくは、次の項目を参照してください。

- 158 ページの『データ・ソース』

- 160 ページの『Human Task Manager のメール・セッション』
- 161 ページの『セキュリティー』
- 163 ページの『状態監視者』
- 164 ページの『SCA バインディング』
- 164 ページの『バス』

データ・ソース

このセクションでは、Business Process Choreographer のデータ・ソースを指定します。

編集

データ・ソースのデータベース固有設定を変更する場合は、これをクリックします。

セットアップによっては、デフォルトを変更する必要があります。例えば、DB2 for z/OS データベースを使用する場合は、以下のようにします。

- ストレージ・グループまたは接続プールの設定を変更します。
- Linux、Windows、UNIX、または i5/OS プラットフォームで DB2 for z/OS データベースを使用するように Business Process Choreographer を構成する場合は、実装タイプに XA データ・ソースを選択する必要があります。

テスト接続

データ・ソースへの接続をテストします。

データベース・インスタンス

Business Flow Manager と Human Task Manager に使用されるデータベースの名前です。

プロパティー	**値**
データ型	ストリング
デフォルト	WPRCSDB

スキーマ名

使用するスキーマの名前。

デフォルトのスキーマの代わりに独自のスキーマを使用する場合、スキーマ名を変更するだけで済みます。

プロパティー	**値**
データ型	ストリング

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には、データベースが存在する必要があり、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。

実動システムでは、このオプションの使用は推奨されません。このオプションを選択しない場合、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。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

ユーザー名

データベースに接続し、データを変更する権限を持つユーザー ID。

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。

プロパティー	値
データ型	ストリング

パスワード

データ・ソースのユーザー ID のパスワード。

プロパティー	値
データ型	ストリング

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 アドレスと、ポート番号を指定します。

プロパティー	値
データ型	ストリング
例	localhost:50000

プロバイダー

Business Process Choreographer の JDBC プロバイダー。

プロパティー	値
データ型	ドロップダウン・リスト

Human Task Manager のメール・セッション

このセクションでは、エスカレーション E メールのパラメーターを指定します。

E メール・サービスを使用可能にする

Human Task Manager がエスカレーションの通知を E メールで送信する場合は、メール・セッションを使用可能にする必要があります。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

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。

プロパティー	値
データ型	ストリング

メール・トランスポートのユーザー

E メール・サービスのユーザー ID。

メール・サーバーで認証が不要な場合は、このフィールドを空白のままにすることができます。

プロパティー	値
データ型	ストリング

メール・トランスポートのパスワード

メール・トランスポート・ユーザー ID のパスワード。

メール・サーバーで認証が不要な場合は、このフィールドを空白のままにすることができます。

プロパティー	値
データ型	ストリング

Business Process Choreographer Explorer の URL

E メールでの Business Process Choreographer Explorer へのリンクに使用する URL を指定します。

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。

プロパティー	値
データ型	ストリング
例	http://www.ibm.com:9080/bpc

セキュリティー

このセクションでは、機能役割からユーザー ID とグループへのマッピング、および Business Process Choreographer で必要な認証資格情報を指定します。

管理者ユーザー

管理者のセキュリティーの役割は、指定したユーザー ID にマップされます。

プロパティー	値
データ型	ストリング
デフォルト	現在ログオンしているユーザー

管理者グループ

管理者のセキュリティーの役割は、指定したグループにマップされます。

プロパティー	値
データ型	ストリング
デフォルト	なし

モニター・ユーザー

システム・モニターのセキュリティーの役割は、指定したユーザー ID にマップされます。

プロパティー	値
データ型	ストリング
デフォルト	現在ログオンしているユーザー

モニター・グループ

システム・モニターのセキュリティーの役割は、指定したグループにマップされます。

プロパティー	値
データ型	ストリング
デフォルト	なし

JMS 認証ユーザー

システム統合バスの認証別名。

プロパティー	値
データ型	ストリング
デフォルト	現在ログオンしているユーザー

JMS 認証パスワードおよび確認パスワード

JMS 認証ユーザー ID のパスワード。

プロパティー	値
データ型	ストリング
デフォルト	なし

JMS API 認証ユーザー

Business Flow Manager メッセージ駆動型 Bean の run-as ユーザー ID。

プロパティー	値
データ型	ストリング
デフォルト	現在ログオンしているユーザー

JMS API 認証パスワードおよび確認パスワード

JMS API 認証ユーザー ID のパスワード。

プロパティー	値
データ型	ストリング
デフォルト	なし

エスカレーション・ユーザーの認証ユーザー

Human Task Manager メッセージ駆動型 Bean の run-as ユーザー ID。

プロパティー	値
データ型	ストリング
デフォルト	現在ログオンしているユーザー

エスカレーション・ユーザーの認証パスワードおよび確認パスワード

エスカレーション・ユーザーの認証ユーザー ID のパスワード。

プロパティー	値
データ型	ストリング
デフォルト	なし

状態監視者

このセクションでは、Business Flow Manager と Human Task Manager に対して、監査ロギングおよび Common Event Infrastructure (CEI) ロギングを使用可能に設定できます。

Business Flow Manager の監査ロギング

このオプションを選択すると、Business Flow Manager の監査ロギングが使用可能になります。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

Human Task Manager の監査ロギング

このオプションを選択すると、Human Task Manager の監査ロギングが使用可能になります。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

Business Flow Manager の Common Event Infrastructure ロギング

このオプションを選択すると、Business Flow Manager の Common Event Infrastructure ロギングが使用可能になります。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

Human Task Manager の Common Event Infrastructure ロギング

このオプションを選択すると、Human Task Manager の Common Event Infrastructure ロギングが使用可能になります。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

SCA バインディング

Service Component Architecture (SCA) バインディングでは、Web サービス API のコンテキスト・ルートを設定できます。

ホスト

この読み取り専用フィールドには、コンテキスト・ルートが付加される Business Flow Manager バインディングと Human Task Manager バインディングのホストのコンテキスト・プレフィックスが示されます。

Business Flow Manager のコンテキスト・ルート

Business Flow Manager Web サービスのコンテキスト・ルート。

プロパティー	値
データ型	ストリング
サーバーでの構成時デフォルト	/BFMIF_*nodeName_serverName*
クラスターでの構成時デフォルト	/BFMIF_*clusterName*

Human Task Manager のコンテキスト・ルート

Human Task Manager Web サービスのコンテキスト・ルート。

プロパティー	値
データ型	ストリング
サーバーでの構成時デフォルト	/HTMIF_*nodeName_serverName*
クラスターでの構成時デフォルト	/HTMIF_*clusterName*

相対パス

この読み取り専用フィールドには、Business Flow Manager と Human Task Manager の SCA バインディングの相対パスが表示されます。

プロパティー	値
データ型	読み取り専用ストリング
Business Flow Manager の相対パス	/sca/com/ibm/bpe/spi/sca/BFMWS
Human Task Manager の相対パス	/sca/com/ibm/task/spi/sca/HTMWS

バス

Business Process Choreographer メッセージング・エンジンでは、Service Component Architecture (SCA) で構成したものとは別のデータ・ソースを使用する場合は、このセクションを展開して設定を変更します。

デフォルト構成の使用

選択されている場合は、SCA メッセージング・エンジンの現行構成の設定が使用されます。

別の設定を使用する場合は、チェック・ボックスをクリアしてこのセクションの他のフィールドを有効にします。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

バス・メンバー・ロケーション

メッセージング・エンジンのデータをローカルまたはリモートのどちらで保管するかを決定します。

「**ローカル**」と「**リモート**」のいずれかを選択します。「**リモート**」を選択すると、リモート宛先のロケーション・セレクターと「**新規**」ボタンが使用可能になります。

プロパティー	値
データ型	ラジオ・ボタン
デフォルト	ローカル

リモート宛先ロケーション

リモート・メッセージング・エンジン・ストアのデプロイメント・ターゲットを指定します。

リストが空の場合、または選択するロケーションが含まれていない場合は、「**新規**」をクリックします。

プロパティー	値
データ型	ドロップダウン・リスト
デフォルト	なし

新規

このボタンにより、「デプロイメント・ターゲットの参照」ページが開きます。

デプロイメント・ターゲットを選択すると、そのターゲットがリモート宛先ロケーションのリストに追加されます。

編集

データ・ソースのデータベース固有設定を変更する場合は、これをクリックします。

セットアップによっては、デフォルトを変更する必要があります。例えば、DB2 for z/OS データベースを使用する場合は、以下のようにします。

- ストレージ・グループまたは接続プールの設定を変更します。
- Linux、Windows、UNIX、または i5/OS プラットフォームで DB2 for z/OS データベースを使用するように Business Process Choreographer を構成する場合は、実装タイプに XA データ・ソースを選択する必要があります。

テスト接続

データ・ソースへの接続をテストします。

データベース・インスタンス

データベースの名前。

プロパティー	値
データ型	ストリング
デフォルト	${USER_INSTALL_ROOT}¥databases¥BPEME

スキーマ名

使用するスキーマの名前。

プロパティー	値
データ型	ストリング
デフォルト	MEDBM00

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には、データベースが存在する必要があり、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。実動システムでは、このオプションにより作成されるデフォルトの表を使用しないようにすることがあります。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

ユーザー名

データベースに接続し、データを変更する権限を持つユーザー ID。

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。

プロパティー	値
データ型	ストリング

パスワード

データ・ソースのユーザー ID のパスワード。

プロパティー	値
データ型	ストリング

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 アドレスと、ポート番号を指定します。

プロパティー	値
データ型	ストリング
例	localhost:50000

プロバイダー

Business Process Choreographer メッセージング・エンジンの JDBC プロバイダー。

ファイル・ストアを使用するように SCA を構成している場合、このフィールドには `File Store` が設定されており、データベース・パラメーターのフィールドは使用不可になっています。データベース・プロバイダーが選択されると、データベース・パラメーターが使用可能になります。

プロパティー	値
データ型	ドロップダウン・リスト
デフォルト	SCA 向けに構成されているプロバイダー。

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。 107 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。バッチ・ファイルまたはコマンド行に必要なパラメーターとオプションをすべて入力する必要があります。このようにしないと、入力されていない必須パラメーターを求めるプロンプトが出されます。スクリプト、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、 198 ページの『bpeconfig.jacl スクリプト・ファイル』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`

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。
	`{-adminBFMUsers userList | -adminBFMGroups groupList}`
	`{-monitorBFMUsers userList | -monitorBFMGroups groupList}`
	`-jmsBFMRunAsUser userID`
	`-jmsBFMRunAsPwd password`
	`{-adminHTMUsers userList | -adminHTMGroups groupList}`
	`{-monitorHTMUsers userList | -monitorHTMGroups groupList}`
	`-jmsHTMRunAsUser userID`
	`-jmsHTMRunAsPwd password`
	`-contextRootBFM contextRootBFM`
	`-contextRootHTM contextRootHTM`
	
	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`

オプション	説明
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 }` `yes` を選択すると、bpeconfig.jacl スクリプトは、SQL ファイルを生成して実行し、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します。このスクリプトは、ハイパフォーマンス・システムには適していません。この場合も、サーバーを停止し、`-conntype NONE` オプションを使用することを計画してください。 `no` を選択したが、データベースが存在していない場合、ユーザーまたはデータベース管理者が生成された SQL スクリプトを実行する必要があります。ハイパフォーマンス・システムの場合、SQL スクリプトを実行する前にカスタマイズする必要があるため、`no` を指定します。また、データベース作成権限がない場合は `no` を指定します。これにより、SQL スクリプトをデータベース管理者に提供し、カスタマイズと実行を依頼できます。 また、サポートが制限されているデータベースを使用する場合にも `no` と指定してください。 **制約事項:** このスクリプトでは、次のタイプのデータベースは作成できません。 • DB2 for z/OS • Oracle • リモート Microsoft SQL Server • リモート Informix Dynamic Server `yes` を選択し、接続モードでスクリプトを実行し、実行時間がデフォルトのタイムアウトである 3 分を超えると、データベースまたはスキーマの作成が失敗することがあります。

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	次のパラメーターを指定します。
	`-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 (詳しくは 198 ページの『bpeconfig.jacl スクリプト・ファイル』を参照)。
	`-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`
	`-dbInstance InformixInstance`
	
	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`
	
	注: 次のいずれかのデータベースを使用している場合は、bpeconfig.jacl でデータベース・インスタンスも作成できます。
	• ローカルの DB2 for Linux, UNIX, or Windows
	• DB2 for iSeries
	• Derby Embedded
	• Derby Network データベース (サーバーが稼働している)

オプション	説明

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 198 ページの『bpeconfig.jacl スクリプト・ファイル』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。

-mqType WPM オプションを使用する場合にメッセージング・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。
	`-meStoreType { FILESTORE | DATASTORE }`
	`-mqCreateTables { true | false }`
	`-mqSchemaName mqSchemaName`
	`-medbUser meDatabaseUser`
	`-medbPwd meDatabasePassword`
	
	さらに、オプションで以下を指定します。
	`-mqDataSource datasourceName`

スクリプトでは常に Business Process Choreographer Explorer が構成される	次のオプション・パラメーターを指定します。
	`-hostName explorerVirtualHostname`
	`-explorerHost explorerURL`
	`-remoteNode nodeName`
	`-remoteServer serverName`
	`-remoteCluster clusterName`
	`-contextRootExplorer explorerContextRoot`
	`-maxListEntries maximum`
	
	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、 239 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。

Business Process Choreographer Observer または Event Collector アプリケーションをデプロイメント・ターゲットにインストールして構成するかどうかが決定している。	次のオプションを使用します。
	`-createEventCollector { yes | no }`
	`-createObserver { yes | no }`
	
	これらのオプションは、bpeconfig.jacl をバッチ・モードで実行する場合にのみ使用できます。また、ハイパフォーマンス・システムには適していません。実動システムの場合、242 ページの『Business Process Choreographer Observer の構成』を行います。

2. デフォルト・プロファイルを作成したときに Business Process Choreographer サンプル構成オプションを選択した場合、Business Flow Manager、Human Task Manager、Business Process Choreographer Explorer、および Business Process Choreographer Observer は既に構成されています。

   これらのコンポーネントが構成されているかどうかを確認するには、管理コンソールで、以下の文字列で始まる名前のエンタープライズ・アプリケーションの有無を確認します。

   - BPCObserver
   - BPCECollector
   - BPEContainer
   - BPCExplorer
   - TaskContainer

   サンプル構成は Derby データベースを使用しており、実動システムには適していません。Business Process Choreographer の構成はデプロイメント・ターゲットごとに 1 つに限られるため、Business Process Choreographer の構成を続行するには、315 ページの『第 5 章 Business Process Choreographer 構成の除去』で説明するようにサンプル構成を削除する必要があります。
3. Network Deployment 環境の場合、Service Component Architecture (SCA) が構成されていることを確認します。
   a. サーバーで Business Process Choreographer を構成するには、「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***serverName***」をクリックし、「**ビジネス・インテグレーション**」 セクションで「**Service Component Architecture**」をクリックします。
   b. クラスターで Business Process Choreographer を構成するには、「**サーバー**」 → 「**クラスター**」 → 「***clusterName***」をクリックし、「**ビジネス・インテグレーション**」セクションで「**Service Component Architecture**」をクリックします。
   c. 「**Service Component Architecture コンポーネントのサポート**」が有効になっていない場合は、これを選択します。
4. WebSphere Platform Messaging (WPM) を JMS プロバイダーとして使用しており、Derby Embedded データベースとオプション `-meStoreType DATASTORE`、または `-meStoreType FILESTORE` を使用しなかった場合は、Business Process Choreographer メッセージング・エンジンのデータ・ストア用のデータベースを作成します。
   - `-mqCreateTables yes` オプションを使用して、メッセージング・エンジンが初めてデータベースを使用するときにデフォルトのスキーマを作成するようにするには、以下を実行します。
     a. データベースが存在していない場合は、作成してください。
     b. データベースのユーザー ID に、使用するスキーマにテーブルとビューを作成できる権限を付与します。
   - このようにしない場合、`-mqCreateTables no` オプションを使用するときには、デフォルトのメッセージング・プロバイダーがデータベースにアクセスする前に、テーブルを作成してください。*install_root* ディレクトリーの `bin`

サブディレクトリーにある sibDDLGenerator ユーティリティーを使用すると、表を作成するときに使用できる DDL ファイルを生成できます。

5. オプション `-createDB yes` を使用して生成済み SQL スクリプトを実行し、データベース・スキーマを作成する予定の場合は、以下のようにします。
   a. 次のいずれかのデータベースを使用している場合:
      - DB2 for z/OS
      - Oracle
      - リモート Microsoft SQL Server
      - リモート Informix Dynamic Server

      かつ、データベースが存在していない場合は、ご使用のデータベースの資料に基づいて、空のデータベースを手動で作成します。
   b. データベース・クライアント (例: db2.exe) が、スクリプト・クライアントのパスに含まれていることを確認します。
   c. アプリケーション・サーバーが停止していることを確認します。
6. 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、 198 ページの『bpeconfig.jacl スクリプト・ファイル』を参照してください。
7. 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、 215 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。
8. `-createDB no` オプションを使用してデータベースの作成を遅延したか、または bpeconfig.jacl スクリプトによるデータベースの作成が失敗した場合は、ステップ 9 で Business Process Choreographer をアクティブにする前に、ユーザーまたはデータベース管理者が、 220 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』で説明するアクションを実行する必要があります。

   注: データベースがローカル・データベースであり、ステップ 9 で Business Process Choreographer をアクティブにする時点でこのデータベースが既に作成されており、 220 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』で説明されているアクションを実行しない場合は、Business Process Choreographer が初めてデータベースにアクセスするときに、デフォルト・スキーマが作成されます。
9. Business Process Choreographer をアクティブにします。 310 ページの『Business Process Choreographer の活動化』を実行します。
10. 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。 310 ページの『Business Process Choreographer の作動確認』を実行します。

11. 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。
12. オプション: Human Task Manager の設定を変更します。
    - エスカレーション E メールの Human Task Manager 設定 (送信者のアドレス、Business Process Choreographer Explorer の URL プレフィックスなど) のいずれかを変更する場合は、「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスターで構成されている場合は「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer Container**」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またはポート番号、E メールのユーザー ID またはパスワードを変更するには、「**リソース**」 → 「**メール**」 → 「**メール・セッション**」の順にクリックし、「**セル**」スコープを選択し、「**HTM メール・セッション**」をクリックして変更を加えます。
13. 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、 227 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、 225 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。
14. 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、 233 ページの『担当者の代替の構成』を行います。
15. 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**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスターで構成されている場合は「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer Container**」を展開し、「**Human Task Manager**」をクリックして「**グループ作業項目の使用可能化**」を選択します。
16. 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 Version 2 インバウンド認証の構成』を参照してください。
17. 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。 238 ページの『Business Process Choreographer Explorer の構成』を実行します。

18. オプション: Business Process Choreographer Observer のインストールおよび構成が完了していない場合は、この時点で構成できます。 242 ページの『Business Process Choreographer Observer の構成』を実行します。
19. オプション: WebSphere Process Server クライアントを使用するリモート Business Process Choreographer クライアントを構成したい場合は、 305 ページの『リモート・クライアント・アプリケーションの構成』を実行します。

結果

Business Process Choreographer が構成されます。

bpeconfig.jacl スクリプト・ファイル

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 Observer 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。

- Linux、UNIX、および i5/OS プラットフォームの場合: *install_root*/ProcessChoreographer/config ディレクトリー
- Windows プラットフォームの場合: *install_root*¥ProcessChoreographer¥config ディレクトリー

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。

DB2 データベースの場合
DB2 がローカルにインストールされていても、Universal Driver タイプ 4 が選択されている場合は、bpeconfig.jacl スクリプトではデータベースを作成できません。

Oracle データベースの場合
bpeconfig.jacl スクリプトで、Oracle データベースを作成することはできません。Business Process Choreographer に Oracle データベースを使用する場合は、データベースを手動で作成する必要があります。

Microsoft SQL Server データベースの場合
bpeconfig.jacl スクリプトで、リモート・データベースを作成することはできません。ローカル・データベースを作成するには、タイプ 2 JDBC ドライ

バーを使用してください。また、-dbServerName パラメーターを指定しないでください。Business Process Choreographer にリモート Microsoft SQL Server データベースを使用する場合は、データベースを手動で作成する必要があります。

スタンドアロン・サーバー環境でのスクリプトの実行

スタンドアロン・サーバー環境の場合:

- `-conntype NONE` オプションを指定するのはアプリケーション・サーバーが稼働していない場合に限定します。
- サーバーが稼働していて 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。
- デフォルト・プロファイルを構成していない場合は、`-profileName` オプションを追加します。

Network Deployment 環境でのスクリプトの実行

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な場合は、`-user` および `-password` オプションを指定します。
- デフォルト・プロファイルを構成していない場合は、`-profileName` オプションを追加します。

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

コマンド行に必要なパラメーターを指定すると、そのパラメーターについてのプロンプトが出されなくなります。Business Process Choreographer を構成するには、以下のコマンドのいずれかを入力します。

Linux および UNIX プラットフォームで、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
```

i5/OS プラットフォームで、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 -f ProcessChoreographer/config/bpeconfig.jacl parameters
```

Windows プラットフォームで、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 -f ProcessChoreographer/config/bpeconfig.jacl parameters
```

ここで、*parameters* は、以下のとおりです。

```
-adminBFMUsers userList
-adminBFMGroups groupList
-adminHTMUsers userList
-adminHTMGroups groupList
-cluster clusterName
-conntype NONE
-contextRootBFM contextRootBFM
-contextRootExplorer explorerContextRoot
-contextRootHTM contextRootHTM
-createDB { yes | no }
-createEventCollector { yes | no }
-createObserver { yes | no }
-createQM { yes | no }
-dbConnectionTarget DB2zOSSubSystem
-dbHome databaseInstallPath
-dbInstance InformixInstance
-dbJava JDBCDriverPath
-dbName databaseName
-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
-meStoreType { FILESTORE | DATASTORE }
-monitorBFMGroups groupList
-monitorBFMUsers userList
-monitorHTMGroups groupList
-monitorHTMUsers userList
-mqClusterName mqClusterName
-mqCreateTables  { true | false }
-mqDataSource datasourceName
-mqHome MQInstallationDirectory
-mqPwd JMSProviderPassword
-mqSchemaName mqSchemaName
-mqType JMSProviderType
-mqUser JMSProviderUserID
-node nodeName
-password  userPassword
-precompileJSPs { yes | no }
-profileName profileName
-qmNameGet getQueueManagerName
-qmNamePut putQueueManagerName
-remoteCluster clusterName
-remoteNode nodeName
-remoteServer serverName
-server serverName}
-user  userName
```

注: 上記の一部のパラメーターは、他のパラメーターに指定する値によってオプションになる場合があります。パラメーター間の依存関係、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パラメーターがある場合は、リストの順番に従って対話式にプロンプトが出されます。

パラメーター

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

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

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

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

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

-conntype NONE
この指定によって、管理接続が使用不可になります。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 (Network Deployment 環境の場合) 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。

-contextRootBFM *contextRootBFM*

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

-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 WebSphere セル内で固有のものでなければなりません。

-contextRootHTM *contextRootHTM*

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

-createDB { yes | no }

指定可能な値は、yes または no です。 yes に設定すると、スクリプトはデータベースを作成します。z/OS データベースおよび Oracle の場合、このスクリプトはデータベースを作成することはできず、テーブル・スペースおよびテーブルだけを作成できます。他のデータベース・タイプの場合、デフォルト値は yes です。実動システムでは no を使用してください。 yes を使用する場合は、bpeconfig.jacl を呼び出すコマンド・プロンプトに、対応するデータベース・コマンド (db2.exe など) を実行するための適切なパスが設定されている必要があります。

-createEventCollector *{ yes | no}*

バッチ・モードで実行する場合、デフォルトは yes です。この設定では、Business Process Choreographer Event Collector アプリケーションが構成されます。これは、Business Process Choreographer Observer では必要な設定です。このオプションを使用する場合、別のデータベースは指定できません。デフォルトでは BPEB データベースが使用されます。つまり、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。インストールしない場合は、このパラメーターの値を no に設定します。

-createObserver *{ yes | no}*

バッチ・モードで実行する場合、デフォルトは yes です。この設定では、Business Process Choreographer Observer アプリケーションが構成されます。このオプションは非対話モードでのみ使用できます。このオプションを使用する場合、別のデータベースは指定できません。デフォルトでは BPEB データベースが使用されます。つまり、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。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・テーブルおよびデータ・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。このパラメーターは、DB2 for z/OS の場合にのみ必須です。デフォルト値は BPEDB です。

-dbHome *databaseInstallPath*

ここで *databaseInstallPath* は、データベース・システムのインストール・ディレクトリーです。このパラメーターは、Informix でのみ必要です。DB2 では、createDB パラメーターが **Yes** に設定されている場合にオプションで指定できます。これは、データベースまたはデータベース・テーブルの作成、およびデータ・ソースの作成に使用されます。デフォルト値および要件は、以下のようにデータベースおよびプラットフォームによって異なります。

DB2 の場合:

- Windows プラットフォームの場合、デフォルトは *current_drive*¥Program Files¥IBM¥SQLLIB です。ここで *current_drive* は、現行ドライブ名です。
- Solaris プラットフォームの場合、デフォルトは /export/home/${dbUser}/sqllib です。
- 他のプラットフォームの場合、デフォルトは /home/${dbUser}/sqllib です。

ディレクトリー ${dbHome}/bnd および ${dbHome}/bin が存在する必要があります。

Informix の場合:

- Windows プラットフォームの場合、デフォルトは *current_drive*¥Program Files¥Informix です。ここで *current_drive* は、現行ドライブ名です。
- Solaris および HP-UX プラットフォームの場合、デフォルトは /opt/informix です。
- Linux および AIX プラットフォームの場合、デフォルトは /usr/informix です。

ファイル ${dbHome}/jdbc/lib/ifxjdbc.jar が存在する必要があります。

-dbInstance *InformixInstance*

ここで *InformixInstance* は、Business Process Choreographer Informix データベースのインスタンス名です。デフォルト値は ids1 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。このパラメーターは、以下のようにデータベースとドライバー・タイプを組み合わせる場合にのみ必須です。

- DB2 Universal とタイプ 4 ドライバー。デフォルト値は *databaseInstallPath*/java です。

- DB2 for i5/OS とタイプ 2 (Native) ドライバー。デフォルト値は /QIBM/ProdData/Java400/ext です。
- DB2 for i5/OS とタイプ 4 (Toolbox) ドライバー。デフォルト値は /QIBM/ProdData/HTTP/Public/jt400/lib/java 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。
- Informix。デフォルト値は *databaseInstallPath*/jdbc/lib です。
- MSSQL DataSource と DataDirect ドライバー・タイプ。デフォルト値はありません。
- Oracle。デフォルト値は *databaseInstallPath*/jdbc/lib 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。

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*

JDBCDriverPath は JDBC ドライバーのパス、*nodeName* はノードの名前です。

-dbName *databaseName*

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

- Oracle の場合、これは TNS です。
- Derby Network (Derby Embedded ではない) の場合、これは絶対パス名である必要があります。
- i5/OS の場合、これはデータベース名または IASP ハードウェア・デバイス名です。Toolbox JDBC ドライバーを使用している場合のデフォルトは *SYSBAS、Native ドライバーを使用している場合のデフォルトは *LOCAL 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*

i5/OS の場合、*schemaQualifier* はコレクション名です。デフォルト値は BPEDB です。その他のすべてのプラットフォームでは、*schemaQualifier* は Business Process Choreographer データベース・テーブルおよびデータ・ソースを作成する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。

-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。 DB2 UDB の場合、このパラメーターはオプションです。指定しない場合は、タイプ 2 JDBC ドライバーが DB2 用に構成されます。指定する場合は、タイプ 4 JDBC プロバイダーが構成されます。
- DB2 for i5/OS の場合、サーバーのショート・ネームを指定します。Toolbox ドライバーを使用している場合、デフォルトはローカル・ホストのショート・ネームです。
- 他のすべてのデータベース・タイプの場合、デフォルト値はローカル・ホストの完全修飾ホスト名です。

-dbServerPort *databaseServerPort*

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

- DB2 の場合、デフォルト値は `50000` です。
- Derby Network の場合、デフォルト値は `1527` です。
- Informix の場合、デフォルト値は `1526` です。
- MSSQL の場合、デフォルト値は `1433` です。
- ドライバー・タイプが thin の Oracle の場合、デフォルト値は `1521` です。

-dbStorageGroup *DB2zOSStorageGroup*

ここで *DB2zOSStorageGroup* は、Business Process Choreographer データベース・テーブルを作成するために使用されるストレージ・グループです。このパラメーターは、DB2 for z/OS の場合にのみ必須です。デフォルト値はなく、空にすることはできません。

-dbTablespaceDir *databaseTablespacePath*

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

- Oracle の場合、デフォルト値はありません。値を指定する必要があります。
- 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。これは、ビジネス・プロセス・コンテナーのインストール、データベースまたはデータベース・テーブルの作成、およびデータ・ソースの作成を行う場合に必要です。デフォルト値はありません。指定可能な値は以下のとおりです。

- Derby
- DB2
- zOS-DB2
- Informix
- iSeries-DB2
- MSSQL
- Oracle

-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、データベースおよびプラットフォームによって以下のように異なります。

- Windows プラットフォーム上の DB2 の場合: 「db2admin」
- 他のプラットフォーム上の DB2 の場合: 「db2inst1」
- Derby Network の場合: 現在ログオンしているユーザーのユーザー ID
- Informix の場合: 「informix」
- Oracle の場合:「system」
- MSSQL の場合: 現在ログオンしているユーザーのユーザー ID

-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。これは、次のデータベース・タイプの場合にのみ必要です

- DB2 for z/OS の場合、*version* の値は `7`、`8`、または `9` のいずれかでなければなりません。
- Oracle の場合、*version* は `9` か `10` の値でなければなりません。
- MSSQL では、データベースでユニコードがサポートされていない場合、*version* は `2000` の値、データベースでユニコードがサポートされている場合、`2000U` の値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 の場合、指定可能な値は `Universal` または `CLI`.
- DB2 for i5/OS の場合、指定可能な値は `native` または `toolbox` です。
- Derby の場合、指定可能な値は `Embedded` または `Network` です。
- Oracle の場合、指定可能な値は `oci8` または `thin` です。
- MSSQL の場合、指定可能な値は `Embedded` または `DataDirect` です。

-explorerHost *explorerURL*

ここで *explorerURL* は、Business Process Choreographer Explorer の URL です。このパラメーターを非クラスター環境で指定しない場合は、デフォルト値が計算されます (例えば、http://localhost:9080)。このパラメーターの値は、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*

VirtualHostname は、Business Process Choreographer と、Business Flow Manager API および Human Task Manager API の Web サービス・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 *userID*

ここで *userID* は、J2EE 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 *userID*

ここで *userID* は、J2EE 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。このパラメーターは、meStoreType パラメーターが *DATASTORE* に設定されており、メッセージング・エンジン・データベースへのアクセスに Derby Embedded JDBC プロバイダーを使用しない場合にのみ必要です。

-meStoreType *{ FILESTORE | DATASTORE }*

Business Process Choreographer メッセージ・エンジンのメッセージ・ストア・タイプを設定します。mqDataSource パラメーターが指定されている場合、このパラメーターは DATASTORE に設定されます。Service Component Architecture (SCA) が FILESTORE を使用する場合、このパラメーターは FILESTORE に設定されます。Network Deployment 環境では FILESTORE はサポートされていません。mqDataSource が設定されておらず、 SCA がメッセージ・ストア・タイプとして DATASTORE を使用する場合、SCA メッセージ・エンジン・データベース設定 (データベース・タイプ、JDBC プロバイダー、データベース・サーバーなど) が継承されます。この場合、個別のデータベース・スキーマを設定する必要があります (後述する mqSchemaName を参照)。また、mqCreateTables フラグも上書きできます。

-monitorBFMGroups *groupList*

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

-monitorBFMUsers *userList*

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

-monitorHTMGroups *groupList*

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

-monitorHTMUsers *userList*

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

-mqType *JMSProviderType*

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

ここで *JMSProviderType* は、以下のいずれかの値です。

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

MQSeries
WebSphere MQ の場合。このオプションを使用する場合は、WebSphere MQ 製品がインストールされている必要があります。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・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。 Service Component Architecture (SCA) についてこのフラグが `true` に設定されている場合、このパラメーターも `true` に設定されます。SCA についてこのフラグが `false` に設定されている場合は、このパラメーターも `false` に設定されます。このオプションは、mqType オプションが `WPM` に、meStoreType が `DATASTORE` に設定されている場合にのみ使用されます。

-mqDataSource *datasourceName*
このオプション・パラメーターを使用すると、基盤となっている SCA 構成からメッセージ・ストアの設定を継承するというデフォルトの動作を上書きできます (meStoreType パラメーターの説明を参照してください)。*datasourceName* は、Business Process Choreographer の宛先ロケーション (SCA メッセージ・エンジンが定義されているサーバーまたはクラスター) で既に定義されているデータ・ソースの JNDI 名です。

-mqHome *MQInstallationDirectory*
ここで、*MQInstallationDirectory* は、WebSphere MQ のインストール・ディレクトリーです。これは、キュー・マネージャーとキュー (Windows プラットフォームのみ) の作成、およびリスナー・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。

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

Windows プラットフォーム:
current_drive¥Program Files¥IBM¥WebSphere MQ

AIX: /usr/mqm

i5/OS: /QIBM/ProdData/mqm

Solaris、HP-UX、および Linux:
/opt/mqm

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

-mqSchemaName *mqSchemaName*
ここで *mqSchemaName* は、デフォルトの JMS プロバイダーのメッセージング・エンジン用のデータベース・スキーマの名前です。デフォルト値は `BPEME` です。このオプションは、meStoreType が `DATASTORE` に設定されている場合にのみ使用されます。

-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 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。
- mqType の値が `MQSeries` の場合、このパラメーターは Linux および UNIX プラットフォーム上で、キュー・マネージャーおよびキューを作成するために使用されます。*JMSProviderUserID* のデフォルト値は `mqm` 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。

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

-profileName *profileName*
ここで *profileName* は、ユーザー定義プロファイルの名前です。このオプションは、デフォルト・プロファイルを構成しない場合に指定します。

-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。

-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。

-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。

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

構成スクリプトの対話式実行

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

制約事項: 対話式に実行する場合、このスクリプトでは Business Process Choreographer Observer も、必要な Event Collector アプリケーションも構成できません。Business Process Choreographer Observer を使用する場合は、 242 ページの『Business Process Choreographer Observer の構成』を実行する必要があります。

1. サーバー上で以下のスクリプトを開始します。Network Deployment 環境の場合はデプロイメント・マネージャー上で開始します。
   - Linux および UNIX プラットフォームの場合は、次のコマンドを入力します。

     ```
     install_root/bin/wsadmin.sh
            -f install_root/ProcessChoreographer/config/bpeconfig.jacl
      ( [-user userName][-password password]|[-conntype NONE])
        [-profileName profileName]
     ```

   - i5/OS プラットフォームの場合は、次のコマンドを入力します。

     ```
     install_root/bin/wsadmin
            -f install_root/ProcessChoreographer/config/bpeconfig.jacl
      ( [-user userName][-password password]|[-conntype NONE])
        [-profileName profileName]
     ```

- Windows プラットフォームの場合は、次のコマンドを入力します。

```
install_root¥bin¥wsadmin.bat
       -f install_root¥ProcessChoreographer¥config¥bpeconfig.jacl
( [-user userName][-password password]|[-conntype NONE])
  [-profileName profileName]
```

2. 表示される質問に対して以下のように対話式に応答します。
   a. Network Deployment 環境では、サーバー、クラスターのいずれかが構成先として示されます。サーバーまたはクラスターが正しくない場合は、**No** と入力して次のサーバーまたはクラスターを表示します。サーバーまたはクラスターが正しい場合は、**Yes** と入力します。
   b. `Install the business process container?` という質問に対しては、**Yes** と入力します。
   c. `User(s) to add to role BPESystemAdministrator` という質問に対しては、ビジネス・プロセス管理者のロールを果たすユーザーのユーザー ID を入力します。
   d. `Group(s) to add to role BPESystemAdministrator` という質問に対しては、ビジネス・プロセス管理者のロールにマップされているグループをドメイン・ユーザー・レジストリーから入力します。
   e. `User(s) to add to role BPESystemMonitor` という質問に対しては、ビジネス・プロセス・モニターのロールを果たすユーザーのユーザー ID を入力します。
   f. `Group(s) to add to role BPESystemMonitor` という質問に対しては、ビジネス・プロセス・モニターのロールにマップされているグループをドメイン・ユーザー・レジストリーから入力します。
   g. `Run-as UserId for role JMSAPIUser` という質問に対しては、JMSAPIUser のロールに使用される別名実行ユーザー ID を入力します。
   h. 別名実行ユーザー ID のパスワードを入力します。
   i. `Use WebSphere default messaging or WebSphere MQ [WPM/MQSeries]?` という質問に対しては、使用する JMS プロバイダーを選択します。
   j. 以下を入力します。
      1) `Virtual Host for the SCA Web Service [default_host]:` という質問に対しては、**Enter** を押して、Service Component Architecture (SCA) Web サーバー仮想ホストのデフォルト値 `default_host` を受け入れます。
      2) `Context root for the SCA Web Service [/BFMIF_PNODE_server1]:` という質問に対しては、**Enter** を押して、デフォルト値 `BFMIF_nodeName_serverName` を受け入れます。
   k. `Create the DataSource for the Process Choreographer database?` という質問に対しては、**Yes** と入力します。
   l. この例での `Create DataSource for a Derby, a DB2, an Informix, an Oracle, or an SQL Server database [Derby/DB2/zOS-DB2/iSeries-DB2/Informix/Oracle/MSSQL]?` という質問に対しては、**DB2** を入力します。他のデータベースを選択した場合は、他のデータベース固有の質問になります。
   m. データベース名を入力してください。
   n. `Database schema name (may be empty)` プロンプトで、**Enter** を押し、暗黙のスキーマ修飾子を使用します。

o. Universal or CLI? という質問に対しては、**Enter** を押し、デフォルトの Universal JDBC ドライバーを選択します。

p. DB2 User ID という質問に対しては、データベースにアクセスするために使用するユーザー ID を入力します。

q. データベース・ユーザー ID のパスワードを入力します。

r. Database server name (may be empty, set to use the type 2 driver) という質問に対しては、データベースをホストするサーバーの名前を入力します。

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

t. JDBC driver directory on [yourHost] プロンプトで、DB2 JDBC ドライバーの JAR ファイルが格納されているディレクトリーを入力します。

u. Create the Process Choreographer database objects? という質問に対しては、データベースを作成する十分な権限が現在のログオン・ユーザー ID にあり、現在の環境で DB2 がセットアップされている場合 (パスに「db2」実行可能ファイルがある場合など) は **Yes** と入力します。データベースを作成する十分な権限が現在のログオン・ユーザー ID にない場合は **No** と入力します。

   Yes と入力した場合は、次のようにします。

   1) DB2 tablespace directory (may be empty) という質問に対しては、**Enter** を押し、空のままにします。

   2) Is 'BPEDB' an existing database (the Process Choreographer schema must not yet exist) という質問に対しては、ご使用の環境に従ってプロンプトが出されます。

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

w. SI バス認証ユーザー ID のパスワードを入力します。

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

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

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

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

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

ac. エスカレーション・ユーザー ID のパスワードを入力します。ステップ 2g (212 ページ) と同じユーザー ID を使用している場合、このプロンプトは表示されません。

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

ae. `Create the mail notification session for the human task manager?` という質問に対しては、Human Task Manager に対してメール通知セッションを作成しない場合、**No** と入力します。作成する場合は **Yes** を入力し、メール・トランスポート・ホストを指定します。オプションで、ユーザー ID およびパスワードを指定できます。

af. `Maximum number of list entries for the Process Choreographer Explorer` という質問に対しては、**Enter** を押して、デフォルト値 `10000` を使用します。

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

ah. `Install the Business Process Choreographer Explorer?` という質問に対しては、**Yes** と入力し、Business Process Choreographer Explorer をインストールします。次に、`Precompile JSPs?` という質問に対しては、Java Server Page (JSP) をプリコンパイルする場合は **Yes** と入力し、それ以外の場合は **No** と入力します。リモート Business Process Choreographer Explorer の場合、`Node of Process Choreographer to connect to [PNODE]:` という質問に対しては、接続先の Business Process Choreographer ノードの名前を入力し、`Server of Process Choreographer to connect to [server1]:` という質問に対しては、接続先の Business Process Choreographer サーバーの名前を入力するか、**Enter** を押してデフォルトを受け入れます。

ai. Business Process Choreographer Explorer のURL を提供する情報など、さまざまな情報が表示されます。例:

```
*******************************************************************************
* NOTE: The Process Choreographer URL will be used by the
* Human Task Manager on server server1 of node viennaNode01
* to link to this Explorer instance. Set an empty URL to not create this link.
*******************************************************************************
URL for this Process Choreographer Explorer [http://host_name:9080/bpc]:
```

この Business Process Choreographer Explorer のインスタンスの URL を入力するか、**Enter** を押してデフォルトを受け入れます。

aj. Business Process Choreographer Observer の構成に使用できるスクリプト・ファイルの場所に関する注意が表示されます。

```
To interactively configure the EventCollector, please use the script
setupEventCollector located in install_root\ProcessChoreographer\config.
To interactively configure the Observer, please use the script
setupObserver located in install_root\ProcessChoreographer\config.
```

3. 問題がある場合は、ログ・ファイルを確認します。

ログ・ファイル

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

- bpeconfig.log

- wsadmin.traceout

どちらのファイルも、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。

- Linux、UNIX、および i5/OS プラットフォームの場合: ディレクトリー *profile_root*/logs 内
- Windows プラットフォームの場合: 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

190 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。 Windows プラットフォームの場合は、次のように入力します。

```
cd install_root¥ProcessChoreographer¥config
createQueues.bat queueManager
```

UNIX および Linux プラットフォームの場合は、次のように入力します。

```
cd install_root/ProcessChoreographer/config
createQueues.sh queueManager
```

ここで、*queueManager* は、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 マシンの 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。
      - 中央キュー・マネージャーが Windows ワークステーションにある場合は、ファイル createQueues.bat をコピーします。
      - 管理コンソールを使用した中央キュー・マネージャーが UNIX または Linux サーバーにある場合は、ファイル createQueues.sh をコピーします。
   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。 Windows プラットフォームの場合は、次のように入力します。

      ```
      cd install_root¥ProcessChoreographer¥config
      createQueues.bat queueManager
      ```

      Linux および UNIX プラットフォームの場合は、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:

      Windows プラットフォームの場合は、次のように入力します。

      ```
      runmqlsr -t tcp -p port -m queueManager
      ```

      Linux および UNIX プラットフォームの場合は、次のように入力します。

      ```
      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。

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

このタスクについて

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

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

      ```
      DEFINE CHANNEL('SYSTEM.ADMIN.SVRCONN') TYPE(CHLTYPE)
      ```
   e. このキュー・マネージャーが WebSphere MQ クラスターのリポジトリーである場合は、MQ コマンドを入力します。

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

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

各クラスター送信側チャネルに対して:

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

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。

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

```
runmqlsr -t tcp -p port -m QueueManager
```

2. WebSphere クラスターが Windows ノードから構成されている場合は、各ノードで次のアクションを実行します。

a. ご使用のユーザー ID に WebSphere MQ キューを作成する権限があることを確認します。

b. ″get″ キュー・マネージャーを作成し、それを 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.bat getQueueManager clusterName putQueueManager
```

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。

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

c. 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 queueManager
```

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

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

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

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

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

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

各クラスター送信側チャネルに対して:

```
DEFINE CHANNEL('TO.repositoryQueueManager.TCP') +
     CHLTYPE(CLUSSDR) +
     CONNAME('repositoryIP-Address(port)') +
     CLUSTER('clusterName') +
     CLUSNL(' ') +
     DESCR('Cluster sender channel to repositoryQueueManager TCPIP') +
     MAXMSGL(4194304) +
     TRPTYPE(TCP) +
     MCAUSER('principal') +
     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
使用する MCAUSER。この値についての詳細は、WebSphere MQ の資料を参照してください。

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

```
runmqlsr -t tcp -p port -m QueueManager
```

3. 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(*)
```

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たか、または bpeconfig.jacl スクリプトがデータベースの作成に失敗しました。

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 dmgr のノード上に生成されます。

- 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、スタンドアロン・サーバーに Business Process Choreographer を構成した場合、createSchema.sql スクリプト・ファイルは、wsadmin が呼び出された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。

オプション	説明
Linux および UNIX の場合	• スキーマ修飾子を指定した場合、生成されるスクリプトは *profile_root*/dbscripts/ProcessChoreographer/*database_type*/*database_name*/*database_schema*/createSchema.sql です。 • スキーマ修飾子を指定**しなかった**場合、生成されるスクリプトは *profile_root*/dbscripts/ProcessChoreographer/*database_type*/*database_name*/createSchema.sql です。
i5/OS の場合	生成されるスクリプトは *profile_root*/dbscripts/ProcessChoreographer/*database_type*/ *collection_name*/createSchema.sql です。
Windows の場合	• スキーマ修飾子を指定した場合、生成されるスクリプトは *profile_root*¥dbscripts¥ProcessChoreographer¥*database_type*¥*database_name*¥*database_schema*¥createSchema.sql です。 • スキーマ修飾子を指定**しなかった**場合、生成されるスクリプトは *profile_root*¥dbscripts¥ProcessChoreographer¥*database_type*¥*database_name*¥createSchema.sql です。 **注:** SQL Server の場合は、createSchemaUnicode.sql という名前のバージョンもあります。このバージョンは、データベースが Unicode 向けに構成されている場合に使用してください。

オプション	説明
z/OS の場合	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
以下のいずれかです。
- DB2
- DB2zOSV7
- DB2zOSV8 (DB2 for z/OS V8 および V9 の場合)
- Db2iSeries
- Derby
- Informix
- Oracle
- SQLServer

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。iSeries 上の DB2 用のみです。

2. データベースがまだ存在していない場合は、 132 ページの『BPEDB データベースの計画』および 123 ページの『セキュリティー、ユーザー ID、および許可の計画』で計画した値に従ってデータベースとユーザー ID を作成するように、データベース管理者に依頼してください。

注: データベースが次のいずれかである場合は、生成されたスクリプトによってデータベース・インスタンスが作成されるので、このステップは不要です。

- Derby Embedded
- Derby Network (データベース・サーバーが稼働している)
- DB2 on iSeries
- DB2 for Linux、UNIX、または Windows ローカル・データベース

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。
      - `@location@` (Oracle の場合は `&1`) を、テーブル・スペース・ディレクトリーに置き換えます。
   b. ハイパフォーマンス・システムが必要な場合には、 132 ページの『BPEDB データベースの計画』のステップ 5 (134 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
Linux、UNIX、または Windows 上の DB2 の場合	`db2 -tf createSchema.sql`
iSeries 上の DB2 の場合	`db2 -tf createSchema.sql`

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.dll`

Derby データベースの場合	`java -Dij.protocol=jdbc:derby:`
	`-Dij.database=BPEDB`
	`org.apache.derby.tools.ij`
	`createSchema.sql`

Informix データベースの場合	`dbaccess` *`databaseName`* `createSchema.sql`
Oracle データベースの場合	`sqlplus` *`userID/password`* *`@database_name`*`@createSchema.sql`

SQL Server データベースの場合	ASCII データベースの場合:
	`sqlcmd -U` *`userID`*
	`-P` *`password`*
	`-d` *`database_name`*
	`-i createSchema.sql`
	
	Unicode データベースの場合:
	`sqlcmd -U` *`userID`*
	`-P` *`password`*
	`-d` *`database_name`*
	`-i createSchemaUnicode.sql`

6. リモート・データベースにアクセスするよう 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。
- ローカル・データベースを持たない Business Process Choreographer を実行しているサーバー。

サーバーがローカル・データベースを持っている場合は、次のステップは実行しないでください。

a. アプリケーション・サーバーをホストしているサーバーに、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。タイプ 2 の JDBC ドライバーを使用している場合は、以下を実行して、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。

Derby の場合

タイプ 4 JDBC プロバイダーだけがサポートされるので、このステップは適用されません。

DB2 Universal Database™ の場合

データベースはカタログされ、別名でアクセス可能です。

DB2 for iSeries の場合

データベースはカタログされ、別名でアクセス可能です。

DB2 for z/OS の場合

データベースはカタログされ、別名でアクセス可能です。

Informix Dynamic Server の場合

タイプ 4 JDBC プロバイダーだけがサポートされるので、このステップは適用されません。

Microsoft SQL Server の場合

タイプ 4 JDBC プロバイダーのみがサポートされるので、このステップは適用されません。

Oracle の場合

TCP ネット・サービス名 (TNS) が、データベースへアクセスするために使用されます。

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。

結果

Business Process Choreographer データベースが存在し、Business Process Choreographer が構成されている任意のリモート・サーバーまたはクラスター・メンバーからそのデータベースにアクセスできます。

担当者ディレクトリー・プロバイダーの構成

このタスクを使用して、Lightweight Directory Access Protocol (LDAP) または Virtual Member Manager (VMM) の担当者ディレクトリー・プロバイダーを構成します。Business Process Choreographer はこれを使用して、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判別します。

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・プロバイダーが必要です。以下の担当者ディレクトリー・プロバイダーがサポートされています。

表 15. 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

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・オプション
Lightweight Directory Access Protocol (LDAP) ディレクトリー	`LDAP people directory provider`
Virtual Member Manager	`VMM 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`

これらのプラグインはすべて、既にデフォルト構成でインストールされています。ユーザー・レジストリーおよびシステム・プラグインはデフォルト構成で使用できます。VMM の場合は、通常はデフォルト構成で十分です。

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

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担当者割り当てを実行するための、Business Process Choreographer 用の 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。デフォルトの担当者ディレクトリー・プロバイダーはいつでも使用できるので、カスタムの担当者割り当て基準を導入する場合に必要なのは、単に構成することだけです。

始める前に

フェデレーテッド・リポジトリーの構成を完了しています。

手順

1. VMM 用の標準変換ファイルをコピーして、それに別の名前 (例えば myVMMTransformation.xsl) を付けます。

- Windows プラットフォームでは、
  install_root¥ProcessChoreographer¥Staff¥VMMTransformation.xsl
- Linux、 UNIX、および i5/OS プラットフォームでは、*install_root*/
  ProcessChoreographer/Staff/VMMTransformation.xsl

2. 229 ページの『LDAP 変換ファイルの適合』の説明に従って、組織の LDAP 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。

   重要: オリジナル・バージョンの変換ファイルは変更しないでください。このファイルは、将来サービス・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、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可能にします。 変換ファイルは ProcessChoreographer ディレクトリーの Staff サブディレクトリーに置く必要があります。
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. 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 が単一サーバー上に構成されているネットワーク・デプロイメント環境:	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ネットワーク・デプロイメント環境:	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 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変換ファイルを選択できます。ここで選択されたファイルは、ローカルの ProcessChoreographer ディレクトリーの Staff サブディレクトリーにコピーされます。
   d. 「**次へ**」をクリックします。 選択したノードにファイルがコピーされます。
   e. 「**一般プロパティー**」セクションで、新規担当者ディレクトリー構成の管理名を入力します。

f. オプション: 説明を入力します。

g. この構成をシステムに識別させる固有の Java Naming and Directory Interface (JNDI) 名を入力します。例えば、bpe/staff/myvmmconfiguration です。

注: 他に構成パラメーターはありません。

h. 「**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。
8. これらのステップのいずれか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。

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。

LDAP 担当者ディレクトリー・プロバイダーの構成

このタスクを使用して、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。これにより、Business Process Choreographer では、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担当者割り当てを実行します。

始める前に

146 ページの『担当者ディレクトリー・プロバイダーの計画』で説明されている LDAP の計画を実行しました。

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て、それに別の名前 (例えば myLDAPTransformation.xsl) を付けます。
   - Windows プラットフォームの場合は、このファイルは *install_root*¥ProcessChoreographer¥Staff¥LDAPTransformation.xsl にあります。
   - Linux、 UNIX、および i5/OS プラットフォームの場合は、このファイルは *install_root*/ProcessChoreographer/Staff/LDAPTransformation.xsl にあります。
2. 229 ページの『LDAP 変換ファイルの適合』の説明に従って、組織の LDAP 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。

   重要: オリジナル・バージョンの変換ファイルは変更しないでください。このファイルは、将来サービス・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、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可能にします。 変換ファイルは ProcessChoreographer ディレクトリーの Staff サブディレクトリーに置く必要があります。

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. 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 が単一サーバー上に構成されているネットワーク・デプロイメント環境:	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ネットワーク・デプロイメント環境:	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」の下の「**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 (227 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変換ファイルを選択できます。ここで選択されたファイルは、ローカルの ProcessChoreographer ディレクトリーの Staff サブディレクトリーにコピーされます。
   d. 「**次へ**」をクリックします。 選択したノードにファイルがコピーされます。
   e. 担当者ディレクトリー構成の管理名を入力します。
   f. 説明を入力します。
   g. 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 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。
   h. 「**適用**」をクリックします。
   i. 「**カスタム・プロパティー**」をクリックします。
   j. ステップ 2 (147 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。

注: オプションの追加プロパティーの場合は、例えば LDAP 参照を使用可能にするために JNDI に定義されているプロパティーを設定できます。providerURL に対しては、`ldap://` または `ldaps://` で始まる URL を指定できます。

k. 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。

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、WebSphere が前提とするデフォルトの LDAP スキーマのエレメントを使用する LDAP 照会に、事前定義の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 DN を含むグループ項目属性は member である。
- 個人項目の LDAP オブジェクト・クラスは inetOrgPerson である。
- 個人項目のログイン ID を含む属性は uid である。
- 個人の E メール・アドレスを含む個人項目属性は mail である。
- 個人の管理者の識別名を含む個人項目属性は manager である。

ご使用の LDAP スキーマが、異なるオブジェクト・クラスや属性名を使用している場合は、使用する LDAP 変換ファイル内でこれらの設定を変更する必要があります。 227 ページの『LDAP 担当者ディレクトリー・プロバイダーの構成』の説明に従い、元の LDAPTransformation.xsl ファイルのコピーを作成します。

重要: オリジナル・バージョンの変換ファイルには変更を加えないでください。このファイルは、将来サービス・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ことがあるからです。

通常、このファイルの変数宣言部分を編集して、すべての担当者割り当て基準の設定を変更するだけで十分です。

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

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

以下の例に示すように、個々の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、225 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 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. 2 つの属性、すなわち一価ストリングとしての「isAbsent」、および多値ストリングとしての「substitutes」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。
      - Linux、UNIX、および i5/OS プラットフォームの場合は、*profile_root*/config/cells/*cell_name*/wim/model にあります。
      - Windows プラットフォームの場合は、*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>
```

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 (236 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。 確認するには、「**環境**」 → 「**WebSphere 変数**」をクリックします。必要に応じて、JDBC ドライバーをクラスパスに追加します。それには、「**アプリケーション・サーバー**」 → 「***server_name***」 → 「**プロセス定義**」 → 「**Java 仮想マシン**」 → 「**構成**」をクリックします。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。
      - Linux、UNIX、および i5/OS プラットフォームの場合は、*profile_root*/config/cells/*cell_name*/wim/model にあります。
      - Windows プラットフォームの場合は、*profile_root*¥config¥cells¥*cell_name*¥wim¥model 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>
```

g. アプリケーション・サーバー (またはネットワーク・デプロイメント環境では、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。 例えば、Windows プラットフォームで DB2 データベースを使用する場合は、以下のようになります。

```
$AdminTask setupIdMgrPropertyExtensionRepositoryTables {
-reportSqlError true
-schemaLocation install_root¥etc¥wim¥setup
-laPropXML  install_root¥etc¥wim¥setup¥wimlaproperties.xml
-databaseType db2
-dbURL jdbc:db2:
-dbDriver com.ibm.db2.jcc.DB2Driver
-dbAdminId userID
-dbAdminPassword password }
```

i. Lightweight Directory Access Protocol (LDAP) ユーザー・リポジトリーを使用している場合は、使用しているセルの win サブディレクトリーの config サブディレクトリー内の wimconfig.xml ファイルを編集し、以下の項目を追加して 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: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: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: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」および「substitutes」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」および「substitutes」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**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックし、「**ビジネス・インテグレーション**」の下で、「**Business Process Choreographer**」を展開し、「**Human Task Manager**」 → 「**構成**」 → 「**カスタム・プロパティー**」をクリックして、カスタム・プロパティー `Substitution.SubstitutesAttribute` および `Substitution.AbsenceAttribute` 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. 管理コンソールを使用して、「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」、または「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」 をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer**」を展開し、「**Human Task Manager**」をクリックして、「**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した場合は、「*Troubleshooting WebSphere Process Server*」(PDF) を参照してください。

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。

概要: Business Process Choreographer Explorer の構成

このタスクについて

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

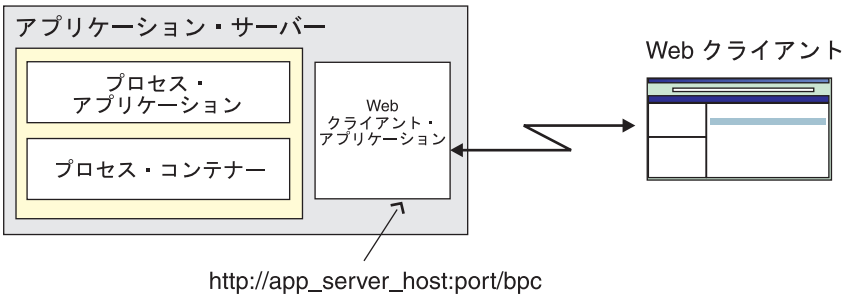
- 154 ページの『Business Process Choreographer Explorer について』
- 238 ページの『Business Process Choreographer Explorer の構成』

Business Process Choreographer Explorer について

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

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 プロファイルを使用した WebSphere Process Server インストールを行うか、WebSphere Process Server クライアント・インストールを実行すれば十分です。Business Process Choreographer をサーバーまたはクラスター上で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 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。

Business Process Choreographer Explorer を構成するには、以下のいずれかを実行します。

- スクリプトを使用する場合は、239 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。
- 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。

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。

次のタスク

Business Process Choreographer Explorer を開始します。

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 を構成できます。

手順

1. 「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」または「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer**」を展開し、「**Business Process Choreographer Explorer**」をクリックします。

2. 新規のエクスプローラー構成を作成するには、「**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。
   - 「**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。**

   これらのフィールドについて詳しくは、167 ページの『Business Process Choreographer Explorer 設定』を参照してください。
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。
6. 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。

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。

次のタスク

Business Process Choreographer Explorer を開始します。

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。

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。

- Linux、UNIX、および i5/OS プラットフォームの場合: *install_root*/ProcessChoreographer/config ディレクトリー
- Windows プラットフォームの場合: *install_root*¥ProcessChoreographer¥config ディレクトリー

スタンドアロン・サーバー環境でのスクリプトの実行

スタンドアロン・サーバー環境の場合:

- -conntype NONE オプションを指定するのはアプリケーション・サーバーが稼働していない場合に限定します。
- サーバーが稼働していて 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。
- デフォルト・プロファイルを構成していない場合は、-profileName オプションを追加します。

Network Deployment 環境でのスクリプトの実行

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な場合は、-user および -password オプションを指定します。
- デフォルト・プロファイルを構成していない場合は、-profileName オプションを追加します。

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、以下のアクションを実行します。

Linux および UNIX プラットフォームの場合は、次のコマンドを入力します。

```
bin/wsadmin.sh -f ProcessChoreographer/config/clientconfig.jacl options
```

i5/OS の場合は、次のコマンドを入力します。

```
bin/wsadmin -f ProcessChoreographer/config/clientconfig.jacl options
```

Windows プラットフォームの場合は、次のコマンドを入力します。

```
bin¥wsadmin.bat -f ProcessChoreographer/config/clientconfig.jacl options
```

options は以下のとおりです。

```
  ( [-user userName][-password password] | [-conntype NONE] )
    [-profileName profileName]
  ( [-node nodeName][-server serverName] )
    [-cluster clusterName]
    [-contextRootExplorer explorerContextRoot]
    [-hostName explorerVirtualHostname]
    [-precompileJSPs { yes | no }]
( ( [-remoteNode nodeName][-remoteServer serverName] )
        | [-remoteCluster clusterName] )
    [-maxListEntries maximum]
    [-explorerHost explorerURL]
```

注: コマンド行に必要なパラメーターをすべて指定すると、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。コマンド行で指定されていないパラメーターがある場合は、リストの順番に従って対話式にプロンプトが出されます。

パラメーター

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

-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。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、この場合リンクが作成されません。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 と、Business Flow Manager API および Human Task Manager API の Web サービス・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。

-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。

-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。

- Linux、UNIX、および i5/OS プラットフォームの場合: ディレクトリー *profile_root*/logs 内
- Windows プラットフォームの場合: 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 Observer の構成

Business Process Choreographer Observer の使用はオプションですが、使用するには、事前にデータベースをセットアップしてアプリケーションをインストールする必要があります。

始める前に

149 ページの『Business Process Choreographer Observer の計画』を実行済みです。

このタスクについて

独自のデータベースを持つ Business Process Choreographer Observer を構成します。

手順

1. 244 ページの『Business Process Choreographer Observer のためのデータベースの準備』を実行します。
2. 282 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. 289 ページの『Business Process Choreographer Observer アプリケーションの構成』を実行します。
4. 294 ページの『Business Process Choreographer Observer の構成パラメーターの変更』を実行します。
5. 292 ページの『Business Process Choreographer のロギング可能化』を実行します。
6. 304 ページの『Business Process Choreographer Observer の検査』を実行します。

結果

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

Business Process Choreographer Observer を使用してレポートを生成できるようになりました (413 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』を参照してください)。

Business Process Choreographer Observer バージョン 6.0.1 サンプルの除去

バージョン 6.0.1 で提供されている Business Process Choreographer Observer のサンプルを除去する方法について説明します。

このタスクについて

WebSphere Process Server バージョン 6.0.1 からマイグレーションして、Business Process Choreographer Observer のサンプルを使用した場合、Business Process Choreographer Observer の最新バージョンを構成する前にそのサンプルを除去しておく必要があります。これらのアプリケーション自体はマイグレーションされていませんが、データベース・ビューおよび索引は除去する必要があります。このサンプルによって収集されたデータはマイグレーションされていないため、使用することはできません。

手順

Business Process Choreographer Observer および Event Collector が使用しているデータベース・ビューおよび索引を除去します。

1. DB2 データベースを使用している場合:
   a. Observer スキーマがあるデータベースに接続します。
   b. Observer のサンプルの DB2 スキーマを除去するスクリプトを見つけて、それを実行します。
      - Windows プラットフォームの場合: *install_root*¥ProcessChoreographer¥sample¥observer¥dropObserverSampleSchema_DB2.sql
      - Linux および UNIX プラットフォームの場合: *install_root*/ProcessChoreographer/sample/observer/dropObserverSampleSchema_DB2.sql

      例えば、以下のように入力します。

      ```
      db2 -tf dropObserverSampleSchema_DB2.sql
      ```
2. Cloudscape™ データベースを使用していた場合は、コマンド・ウィンドウで以下を実行します。
   a. derby.jar および derbytools.jar ファイルを CLASSPATH に追加します。
      - Windows プラットフォームの場合は *install_root*¥derby¥bin¥embedded にあります。
      - Linux および UNIX プラットフォームの場合は *install_root*/derby/bin/embedded にあります。
   b. このスクリプトを見つけ、このファイルのコメントの説明に従ってそれを実行し、Observer のサンプルの Cloudscape スキーマを除去します。

- Windows プラットフォームの場合: *install_root*¥ProcessChoreographer¥sample¥observer¥dropObserverSampleSchema_Cloudscape.sql
- Linux および UNIX プラットフォームの場合: *install_root*/ProcessChoreographer/sample/observer/dropObserverSampleSchema_Cloudscape.sql.

例えば、以下のコマンドを入力します。

```
java -Dij.protocol=jdbc:derby:
     -Dij.database=OBSVRDB
     org.apache.derby.tools.ij
     dropObserverSampleSchema_Cloudscape.sql
```

結果

Business Process Choreographer Observer のサンプルが除去されました。

Business Process Choreographer Observer のための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。

Business Process Choreographer Observer のための DB2 Universal Database 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、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 Observer のための DB2 Universal Database の準備:

ここでは、createTablespace_Observer.sql および createSchema_Observer.sql スクリプトを使用して、DB2 Universal Database を Linux、UNIX、および Windows のプラットフォームに準備する方法について説明します。

このタスクについて

データベースが既に存在している必要があります。既存のデータベースを使用することも、データベースの資料に従って新規に作成することもできます。このタスクを実行するには、宛先データベースに対する管理権限が必要です。

手順

1. データベース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。
   - Linux および UNIX プラットフォームの場合は、ディレクトリーを *install_root*/dbscripts/ProcessChoreographer/DB2 に変更します。
   - Windows プラットフォームの場合は、ディレクトリーを *install_root*¥dbscripts¥ProcessChoreographer¥DB2 に変更します。
2. すべての *Observer.sql スクリプト・ファイルを、ご使用のデータベース・サーバーにコピーします。
3. このデータベース・サーバーで、スクリプト・ファイルをコピーしたディレクトリーに移動します。

4. テーブル・スペースを作成します。
   a. ファイルの先頭にある指示に従い、createTablespace_Observer.sql スクリプト・ファイルを編集します。
   b. テーブル・スペース作成スクリプト・ファイルを実行し、次のコマンドを入力します。

      ```
      db2 -tf createTablespace_Observer.sql
      ```

   c. スクリプト出力にエラーがないことを確認します。エラーが発生した場合は、dropTablespace_Observer.sql スクリプト・ファイルを使用してそのテーブル・スペースを除去できます。
5. スキーマ (テーブル、索引、およびビュー) を作成します。
   a. createSchema_Observer.sql スクリプト・ファイルの先頭にある指示に従い、このファイルを編集します。
   b. DB2 コマンド行プロセッサーで、次のコマンドを入力します。

      ```
      db2 -tf createSchema_Observer.sql
      ```

   c. スクリプト出力にエラーがないことを確認します。スキーマを除去する場合は、dropSchema_Observer.sql スクリプト・ファイルを使用します。
6. SQL 実装ではなく、Java 実装の Business Process Choreographer Observer UDF を使用する場合は、 275 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、接続をテストします。

結果

Business Process Choreographer Observer 用のデータベース・スキーマが作成されました。

関連概念

280 ページの『Business Process Choreographer Observer のユーザー定義関数』
Business Process Choreographer Observer 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 Observer では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

275 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』
setupEventCollector ツールを使用するか、またはスクリプトを実行して、Business Process Choreographer Observer 用のデータベースで 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ことができます。

setupEventCollector ツールを使用した Business Process Choreographer Observer のための DB2 Universal Database の準備:

ここでは、対話式メニュー方式のツールおよび createTablespace_Observer.sql スクリプトを使用して、Linux、UNIX、および Windows プラットフォームに DB2 Universal Database を準備する方法について説明します。

始める前に

データベースが既に存在している必要があります。

手順

1. タイプ 2 の JDBC 接続を使用する場合:
   a. 以下のようにコマンド行環境を準備します。
      - Linux および UNIX プラットフォームの場合は、DB2 インスタンスに対して db2profile を実行します。
      - Windows の場合は、DB2 コマンド・ウィンドウを開きます。
   b. データベースがリモート環境にある場合は、ローカルの DB2 インスタンスでデータベースをカタログします。
2. テーブル・スペースを作成します。
   a. 使用しているデータベースに対応した SQL 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。
      - Linux および UNIX プラットフォームの場合は、ディレクトリーを *install_root*/dbscripts/ProcessChoreographer/DB2 に変更します。
      - Windows プラットフォームの場合は、ディレクトリーを *install_root*¥dbscripts¥ProcessChoreographer¥DB2 に変更します。
   b. createTablespace_Observer.sql スクリプト・ファイルをコピーします。
   c. ファイルの先頭にある指示に従い、createTablespace_Observer.sql スクリプト・ファイルのコピーを編集します。
   d. SYSCTRL または SYSADM 権限を持つユーザー ID を使用してデータベースに接続します。
   e. テーブル・スペース作成スクリプト・ファイルを実行し、次のコマンドを入力します。

      ```
      db2 -tf createTablespace_Observer.sql
      ```

   f. スクリプト出力にエラーがないことを確認します。 エラーが発生した場合は、dropTablespace_Observer.sql スクリプト・ファイルを使用してそのテーブル・スペースを除去できます。
3. データベース管理者でないユーザー ID を使用している場合は、以下の権限を持っていることを確認します。

   ```
   GRANT CREATETAB, CONNECT, CREATE_EXTERNAL_ROUTINE ON DATABASE
         TO USER user_name;
   GRANT USE OF TABLESPACE tablespace_name TO USER user_name;
   ```

   ここで、*user_name* はユーザー ID で、*tablespace_name* は、スクリプト createTablespace_Observer.sql 内にあるすべての Observer テーブル・スペース名のリストです。
4. 構成スクリプトがある Business Process Choreographer ディレクトリーに移動します。

   Linux および UNIX プラットフォームの場合は、次のように入力します。

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

   Windows プラットフォームの場合は、次のように入力します。

```
cd install_root¥ProcessChoreographer¥config
```

5. 299 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。
6. 以下のようにデータベースを準備します。
   a. 以下が表示された場合:

```
1) Prepare a database for the Event Collector and Observer
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 Observer
6) Administer Observer related user-defined functions

0) Exit Menu
```

   オプション 1 を選択して、Event Collector アプリケーションおよび Observer アプリケーション用にデータベースを準備します。

   b. 以下が表示された場合:

```
Prepare a database for the WebSphere Business Process Choreographer
Event Collector and Observer

Select the type of your database provider:

    c) Derby
    d) DB2 Universal
    i) DB2 iSeries
    7) DB2 V7 on z/OS
    8) DB2 V8/V9 on z/OS
    o) Oracle

    0) Exit Menu
```

   d と入力して、DB2 Universal を選択します。

   c. 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。

   d. 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。

e. 以下のプロンプトのいずれかが表示された場合:

```
Specify the name of your database:  [BPEDB]
Specify the name of database in local catalog:  [BPEDB]
```

データベース名または別名。

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。ハイパフォーマンス・システムの場合は、別のデータベースを使用してください。別のデータベースを使用する場合は、操作を続ける前にそのデータベースが存在していなければなりません。

f. 以下が表示された場合:

```
Specify the hostname of the database server:  [localhost]
```

使用しているデータベース・サーバーのホスト名または IP アドレスを入力します。

g. 以下が表示された場合:

```
Specify the port where the database server is listening:  [50000]
```

使用しているデータベース・サーバーのポート番号を入力します。

h.

```
Specify the directory of your JDBC driver:  [D:\opt\SQLLIB\java]
```

JDBC ドライバー用の db2jcc.jar および db2jcc_license_cu.jar JAR ファイルがあるディレクトリーを入力します。

i. 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。 以下が表示されます。

```
 Trying to connect to database 'database_name', using user 'user_ID'
Connected to 'database_name'
```

j. 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されるデータベース・スキーマ (コレクション名) を入力します。スペース文字を入力するかフィールドを空にしておくと、ユーザー ID のスキーマが使用されます。

k. 以下が表示された場合:

```
Choose the implementation of the Observer user-defined functions.

  Note: The Java UDFs are more precise, but they require a jar file
  installed to the database.
  Visit the Observer 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。

以下のようなメッセージが表示されます。

```
Checking for required tablespace(es) ['OBSVRTS']
All required tablespaces were found.
Loading the jar file 'install_root\lib\bpcodbutil.jar' into the database.
The jar file 'install_root\lib\bpcodbutil.jar' was successfully installed.

The setup of the database completed successfully.
```

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、接続をテストします。

結果

Business Process Choreographer Observer 用のデータベース・スキーマが作成されました。

関連概念

280 ページの『Business Process Choreographer Observer のユーザー定義関数』
Business Process Choreographer Observer 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 Observer では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

275 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』
setupEventCollector ツールを使用するか、またはスクリプトを実行して、Business Process Choreographer Observer 用のデータベースで 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ことができます。

Business Process Choreographer Observer のための DB2 for iSeries 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、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 Observer のための DB2 for iSeries データベースの準備:

ここでは、createSchema_Observer.sql スクリプトを i5/OS qshell 環境で使用して、DB2 for iSeries データベースを準備する方法について説明します。

始める前に

コレクションが既に存在している必要があります。既存のコレクションを使用するか、またはデータベースの資料に従って新規コレクションを作成することができます。データベースの管理権限 (*ALLOBJ) が必要です。

手順

1. qshell 環境では、データベースの構成スクリプトがある 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/dbscripts/ProcessChoreographer/DB2iSeries
   ```

2. すべての *Observer.sql スクリプト・ファイルを、ご使用のデータベース・サーバーにコピーします。
3. このデータベース・サーバーで、スクリプト・ファイルをコピーしたディレクトリーに移動します。
4. スキーマ (テーブル、索引、およびビュー) を作成します。
   a. createSchema_Observer.sql スクリプト・ファイルを、このファイルの先頭にある指示に従って編集します。
   b. DB2 コマンド行プロセッサーまたは qshell で、以下のコマンドを入力します。

      ```
      db2 -tf createSchema_Observer.sql
      ```

   c. スクリプト出力にエラーがないことを確認します。スキーマを除去する場合は、dropSchema_Observer.sql スクリプト・ファイルを使用します。
5. 必要な Java 実装のユーザー定義関数 (UDF) を使用する場合は、 275 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。
6. 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。

結果

Business Process Choreographer Observer 用のデータベース・スキーマが作成されました。

関連概念

280 ページの『Business Process Choreographer Observer のユーザー定義関数』
Business Process Choreographer Observer 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 Observer では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

275 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』
setupEventCollector ツールを使用するか、またはスクリプトを実行して、Business

Process Choreographer Observer 用のデータベースで 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ことができます。

setupEventCollector ツールを使用した Business Process Choreographer Observer のための DB2 for iSeries 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を使用して、i5/OS qshell 環境内から DB2 for iSeries データベース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

既存のコレクションを使用するか、またはデータベースの資料に従って新規コレクションを作成することができます。

手順

1. qshell 環境を開始します。
2. 構成スクリプトがある Business Process Choreographer 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/config
   ```

3. 299 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. 以下のようにデータベースを準備します。
   a. 以下が表示された場合:

      ```
      1) Prepare a database for the Event Collector and Observer
      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 Observer
      6) Administer Observer related user-defined functions

      0) Exit Menu
      ```

      オプション 1 を選択して、Event Collector アプリケーションおよび Observer アプリケーション用にデータベースを準備します。

   b. 以下が表示された場合:

      ```
      Prepare a database for the WebSphere Business Process Choreographer
      Event Collector and Observer

      Select the type of your database provider:

           c) Derby
           d) DB2 Universal
           i) DB2 iSeries
           7) DB2 V7 on z/OS
           8) DB2 V8/V9 on z/OS
           o) Oracle

           0) Exit Menu
      ```

      i を入力して DB2 for iSeries を選択します。

   c. 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。

d. 以下が表示された場合:

```
Specify the JDBC driver to be used:

    1) Connect using the IBM Toolbox for Java JDBC driver
    2) Connect using the native JDBC driver
 Your selection: [2]
```

- リモート・データベースを構成している場合は、1 を入力して、IBM® Toolbox for Java JDBC ドライバーを選択します。
- ローカル・データベースを構成している場合は、2 を入力してネイティブの JDBC ドライバーを選択します。

e. 以下が表示された場合:

```
Specify the name of database in local catalog:  [*LOCAL]
```

または

```
Specify the name of your database:  [*SYSBAS]
```

サービス ID を入力するか、またはデフォルトを受け入れます。

f. 以下が表示された場合:

```
Specify the hostname of the database server:  [localhost]
```

使用しているデータベース・サーバーのホスト名または IP アドレスを入力します。

g. 以下が表示された場合:

```
Specify the directory of your JDBC driver:
[/QIBM/ProdData/HTTP/Public/jt400/lib]
```

JDBC ドライバー JAR ファイルがあるディレクトリーを入力します。

- ネイティブ・ドライバー (db2_classes.zip) の場合、このディレクトリーは通常 /QIBM/ProdData/Java400/ext です。
- ツールボックス・ドライバー (jt400.jar) の場合、このディレクトリーは通常 /QIBM/ProdData/HTTP/Public/jt400/lib です。

h. 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。

i. 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されるデータベース・スキーマ (コレクション名) を入力します。既存のスキーマを指定する必要があります。スペース文字を入力するかフィールドを空にしておくと、ユーザー ID のスキーマが使用されます。

j. 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 Observer 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。

k. データベースが正常に準備されると、以下が表示されます。

```
The setup of the database completed successfully.
```

5. BPEDB ではなく、別個のデータベースを使用した場合は、管理コンソールを使用して、そ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。

結果

Business Process Choreographer Observer 用のデータベース・スキーマが作成されました。

関連概念

280 ページの『Business Process Choreographer Observer のユーザー定義関数』
Business Process Choreographer Observer 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 Observer では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

275 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』
setupEventCollector ツールを使用するか、またはスクリプトを実行して、Business Process Choreographer Observer 用のデータベースで 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ことができます。

リモート・システムからの setupEventCollector ツールを使用した DB2 for iSeries 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を使用して、リモートの Linux、Windows、または UNIX システムから Business Process Choreographer Observer のための DB2 for iSeries データベース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

既存のコレクションを使用するか、またはデータベースの資料に従って新規コレクションを作成することができます。使用されるコレクションが既に存在している必要があります。

手順

1. データベースをリモートで準備するには、IBM Toolbox JDBC ドライバーをダウンロードして、使用している iSeries マシンに接続する必要があります。ダウンロード後に、jar ファイル jt400.jar のロケーションを書き留めておいてください。
2. コマンド行環境を開始します。
3. 構成スクリプトがある Business Process Choreographer ディレクトリーに移動します。
   - Windows プラットフォームの場合は、次のように入力します。

     ```
     cd install_root¥ProcessChoreographer¥config
     ```
   - Linux および UNIX プラットフォームの場合は、次のように入力します。

     ```
     cd install_root/ProcessChoreographer/config
     ```
4. 299 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。
5. 以下のようにデータベースを準備します。
   a. 以下が表示された場合:

      ```
      1) Prepare a database for the Event Collector and Observer
      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 Observer
      6) Administer Observer related user-defined functions

      0) Exit Menu
      ```

      オプション 1 を選択して、Event Collector アプリケーションおよび Observer アプリケーション用にデータベースを準備します。

   b. 以下が表示された場合:

      ```
      Prepare a database for the WebSphere Business Process Choreographer
      Event Collector and Observer

      Select the type of your database provider:

          c) Derby
          d) DB2 Universal
          i) DB2 iSeries
          7) DB2 V7 on z/OS
          8) DB2 V8/V9 on z/OS
          o) Oracle

          0) Exit Menu
      ```

i を入力して DB2 for iSeries を選択します。

c. 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。

d. 以下が表示された場合:

```
Specify the name of your database:  [*SYSBAS]
```

サービス ID を入力するか、またはデフォルトを受け入れます。

e. 以下が表示された場合:

```
Specify the hostname of the database server:  [localhost]
```

使用しているデータベース・サーバーのホスト名または IP アドレスを入力します。

f. 以下が表示された場合:

```
Specify the directory of your JDBC driver:
[/QIBM/ProdData/HTTP/Public/jt400/lib]
```

JDBC ドライバー・ファイル jt400.jar をダウンロードしたディレクトリーを入力します。

g. 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。

h. 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されるデータベース・スキーマ (コレクション名) を入力します。既存のスキーマを指定する必要があります。スペース文字を入力するかフィールドを空にしておくと、ユーザー ID のスキーマが使用されます。

i. 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 Observer 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。

j. データベースが正常に準備されると、以下が表示されます。

```
The setup of the database completed successfully.
```

6. BPEDB ではなく、別個のデータベースを使用した場合は、管理コンソールを使用して、そ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。

結果

Business Process Choreographer Observer 用のデータベース・スキーマが作成されました。

関連概念

280 ページの『Business Process Choreographer Observer のユーザー定義関数』
Business Process Choreographer Observer 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 Observer では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

275 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』
setupEventCollector ツールを使用するか、またはスクリプトを実行して、Business Process Choreographer Observer 用のデータベースで 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ことができます。

Business Process Choreographer Observer のための DB2 for z/OS データベースの準備

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 Business Process Choreographer Observer 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、z/OS マシン上の UNIX System Services (USS) で使用して、DB2 for z/OS データベースを作成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 (_SQLID) 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 提供のストアード・プロシージャーの使用可能化、および DB2 Universal JDBC ドライバーによって使用されるテーブルの定義

2) 解釈済み Java ルーチン向け環境のセットアップ

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. テーブル・スペースを作成します。
   a. 使用しているデータベース・システムに対応した Business Process Choreographer Observer 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/DB2zOSV7
      cd install_root/dbscripts/ProcessChoreographer/DB2zOSV8
      ```

      注: DB2 for z/OS V9 を使用する場合は、V8 オプションを使用してください。

   b. 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_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
      ```

   c. 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。
4. 299 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。
5. 以下のようにデータベースを準備します。 以下が表示された場合:

   ```
   1) Prepare a database for the Event Collector and Observer
   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 Observer
   6) Administer Observer related user-defined functions

   0) Exit Menu
   ```

   a. 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。
   b. 7 または 8 を入力して、z/OS 上の DB2 のバージョン番号を選択します。
   c. 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。

d. 以下が表示された場合:

```
Specify the database location name:
  (as returned by SELECT CURRENT SERVER FROM SYSIBM.SYSDUMMY1):
```

使用しているデータベースのロケーション名を入力します。これは、ステップ 1d (257 ページ) で書き留めておいた値です。

e. 以下が表示された場合:

```
Specify the name of the database as known by the susbsystem [subsystem]
```

z/OS ホスト上のサブシステムにおけるご使用のデータベースの名前を入力します。これは、ステップ 1f (257 ページ) で書き留めておいた値です。

f. 以下が表示された場合:

```
Specify the hostname of the z/OS DB2 database server: [localhost]
```

使用しているデータベース・サーバーのホスト名を入力します。

g. 以下が表示された場合:

```
Specify the port where the database subsystem is listening:
```

データベース・サブシステムが使用しているポート番号を入力します。これは、ステップ 1c (257 ページ) で書き留めておいた値です。

h. 以下が表示された場合:

```
Specify userid to connect to the database 'database_alias' [db2admin] :
```

データベースに接続するユーザー ID を入力します。これは、ステップ 1h (257 ページ) で説明されているユーザー ID *user_ID* です。

i. 以下が表示された場合:

```
Specify the password for userid 'user_ID' :
```

ユーザー ID のパスワードを入力します。

j. 以下が表示された場合:

```
Trying to connect to database 'database_alias', using user 'user_ID'
Connected to 'database_alias'
Specify the database schema to be used. [user_ID] :
```

データベース・オブジェクトに使用されるデータベース・スキーマを入力するか、または Enter を押してデフォルトを受け入れます。デフォルトは、データベースとの接続に使用されるユーザー ID です。これはスキーマ修飾子 _SQLID です。

k. 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 Observer 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。

l. 以下が表示された場合:

```
Specify the DB2 storage group name to be used. [OBSVRSG] :
```

ステップ 1e (257 ページ) のストレージ・グループ名を入力するか、または Enter を押してデフォルト値を受け入れます。

m. 以下が表示された場合:

```
Specify the WLM environment name where the UDF should run. [] :
```

ステップ 1k (257 ページ) で書き留めておいた WLM 環境を入力します。

n. 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.
```

6. 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。

結果

Business Process Choreographer Observer 用のデータベース・スキーマが作成されました。

関連概念

280 ページの『Business Process Choreographer Observer のユーザー定義関数』
Business Process Choreographer Observer 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 Observer では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

275 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』
setupEventCollector ツールを使用するか、またはスクリプトを実行して、Business

Process Choreographer Observer 用のデータベースで 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ことができます。

リモート・システムからの Business Process Choreographer Observer 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 Linux、UNIX、または Windows システムで使用して、DB2 for z/OS データベースを作成する方法について説明します。

始める前に

WebSphere Process Server が Linux、UNIX、または Windows 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 (_SQLID) 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 提供のストアード・プロシージャーの使用可能化、および DB2 Universal JDBC ドライバーによって使用されるテーブルの定義
2) 解釈済み Java ルーチン向け環境のセットアップ

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. 適切な DB2 クライアントをインストールします。

注: ネイティブの DB2 クライアントを使用してリモート・データベースに接続する場合は (タイプ 2 の JDBC 接続を使用)、DB2 Connect™ ゲートウェイがインストールされていることを確認してください。DB2 Connect ゲートウェイは、DB2 UDB ESE パッケージに組み込まれていますが、単体でインストールすることもできます。

b. 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 マシンの 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
```

c. 使用しているデータベース・システムに対応した Business Process Choreographer Observer 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。

   - Linux および UNIX プラットフォームでは、DB2 のバージョンに応じて、次のコマンドのいずれかを入力します。

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

   - Windows プラットフォームでは、ご使用の DB2 のバージョンに応じて、次のコマンドのいずれかを入力します。

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

   注: DB2 for z/OS V9 を使用する場合は、V8 オプションを使用してください。

d. 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。

e. 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。 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。

f. 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。

   Linux および UNIX プラットフォームの場合は、次のように入力します。

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

   Windows プラットフォームの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥config
   ```

g. 299 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。

h. 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。

i. 7 または 8 を入力して、z/OS 上の DB2 のバージョン番号を選択します。

j. 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。

k. 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。

l. 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 2b (262 ページ) で *database_alias* に使用した値です。

m. 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
```

n. 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。

o. 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。

p. 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。

q. 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 (261 ページ) で説明されているユーザー ID *user_ID* です。

r. 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。

s. 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 Observer 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。

t. 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 (261 ページ) のストレージ・グループ名を入力します。

u. 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 (262 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。

結果

Business Process Choreographer Observer 用のデータベース・スキーマが作成されました。

関連概念

280 ページの『Business Process Choreographer Observer のユーザー定義関数』
Business Process Choreographer Observer 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 Observer では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

275 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』 setupEventCollector ツールを使用するか、またはスクリプトを実行して、Business Process Choreographer Observer 用のデータベースで 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ことができます。

Business Process Choreographer Observer 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、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 Observer 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、Derby データベース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

Business Process Choreographer Observer 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. データベース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。
   - Linux、UNIX、および i5/OS プラットフォームの場合は、ディレクトリーを *install_root*/dbscripts/ProcessChoreographer/Derby に変更します。
   - Windows プラットフォームの場合は、ディレクトリーを *install_root*¥dbscripts¥ProcessChoreographer¥Derby に変更します。
2. 以下を実行します。
   a. 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。また、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。
   b. テキスト・エディターで、スクリプト・ファイル `createSchema_Observer.sql` のヘッダーに記載されている説明を読み取ります。 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。
   c. 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。

d. 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。

e. 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。

結果

Business Process Choreographer Observer 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 Observer 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 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。

   Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

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

   Windows プラットフォームの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥config
   ```

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. 299 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 Observer
   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 Observer
   6) Administer Observer 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 WebSphere Business Process Choreographer
   Event Collector and Observer
   ```

```
Select the type of your database provider:

     c) Derby
     d) DB2 Universal
     i) DB2 iSeries
     7) DB2 V7 on z/OS
     8) DB2 V8/V9 on z/OS
     o) Oracle

     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 JDBC driver
  2) Connect using the network 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: データベースの名前 [*database_name*] を指定します。

   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 /
    Cloudscape 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 (267 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 (267 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。

結果

Business Process Choreographer Observer 用のデータベース・スキーマが作成されました。

Business Process Choreographer Observer のための Oracle 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、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 Observer のための Oracle データベースの準備:

ここでは、createTablespace_Observer.sql および createSchema_Observer.sql スクリプトを使用して、Oracle データベース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

データベースが既に存在している必要があります。既存のデータベースを使用することも、データベースの資料に従って新規に作成することもできます。

手順

1. データベース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。
   - Linux、UNIX、および i5/OS プラットフォームの場合は、ディレクトリーを *install_root*/dbscripts/ProcessChoreographer/Oracle に変更します。
   - Windows プラットフォームの場合は、ディレクトリーを *install_root*¥dbscripts¥ProcessChoreographer¥Oracle に変更します。
2. すべての *Observer.sql スクリプト・ファイルを、ご使用のデータベース・サーバーにコピーします。
3. Java ベースの Business Process Choreographer ユーザー定義関数を使用する場合:
   a. JAR ファイル bpcodbutil.jar も、*install_root* ディレクトリーの lib サブディレクトリーから、SQL スクリプト・ファイルを含んでいるディレクトリーにコピーします。
   b. Business Process Choreographer Observer の UDF を含む JAR ファイルをインストールします。

1) Oracle 管理権限を持つユーザーとしてデータベース・サーバーにログオンし、JAR ファイル bpcodbutil.jar があるディレクトリーに移動します。
   - データベースがアプリケーション・サーバーと同じサーバーにある場合は、*install_root* ディレクトリーの lib サブディレクトリーに移動します。
   - データベースがアプリケーション・サーバーと同じマシンにない場合は、JAR ファイル bpcodbutil.jar をコピーしたディレクトリーに移動します。
2) 次のコマンドを入力して、Oracle loadjava ユーティリティーを実行し、JAR ファイル bpcodbutil.jar をインストールします。

```
loadjava -user user/password@database
     -schema  schema_name
     -resolve bpcodbutil.jar
```

   ここで、

   user、*password*、および *database* は、ユーザー ID、パスワード、およびデータベース名に対して有効な値です。

   schema_name は、クラスを保管するときのスキーマ名です。これは、Observer テーブルに使用されているのと同じスキーマである必要があります。
3) 問題が発生した場合は、以下のコマンドを使用して JAR ファイルを除去できます。

```
dropjava -user user/password@database
     -schema  schema_name bpcodbutil.jar
```

   .

4. テーブル・スペースの作成
   a. ファイルの先頭にある指示に従い、createTablespace_Observer.sql スクリプト・ファイルのコピーを編集します。
   b. ファイルの先頭にある指示に従い、createTablespace_Observer.sql スクリプト・ファイルのコピーを実行します。
   c. スクリプト出力にエラーがないことを確認します。エラーが発生した場合は、dropTablespace_Observer.sql スクリプト・ファイルを使用してそのテーブル・スペースを除去できます。
5. スキーマ (テーブル、索引、およびビュー) を作成します。
   a. ファイルの先頭にある指示に従い、createSchema_Observer.sql スクリプト・ファイルのコピーを編集および実行します。
   b. スクリプト出力にエラーがないことを確認します。スキーマを除去する場合は、dropSchema_Observer.sql スクリプト・ファイルを使用します。
6. 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。

結果

Business Process Choreographer Observer 用のデータベース・スキーマが作成されました。

関連概念

280 ページの『Business Process Choreographer Observer のユーザー定義関数』
Business Process Choreographer Observer 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 Observer では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

275 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』
setupEventCollector ツールを使用するか、またはスクリプトを実行して、Business Process Choreographer Observer 用のデータベースで 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ことができます。

setupEventCollector ツールを使用した Business Process Choreographer Observer のための Oracle 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、および createTablespace_Observer.sql スクリプトを使用して、Oracle データベース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

データベースが既に存在している必要があります。既存のデータベースを使用することも、データベースの資料に従って新規に作成することもできます。

注: i5/OS プラットフォームからリモートの Oracle データベースを作成するには、270 ページの『SQL スクリプトを使用した Business Process Choreographer Observer のための Oracle データベースの準備』を実行します。

手順

1. Linux および UNIX プラットフォームの場合: $ORACLE_HOME/bin を PATH 変数に追加します。
2. テーブル・スペースを作成します。
   a. データベース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。
      - Linux および UNIX プラットフォームの場合は、ディレクトリーを *install_root*/dbscripts/ProcessChoreographer/Oracle に変更します。
      - Windows プラットフォームの場合は、ディレクトリーを *install_root*¥dbscripts¥ProcessChoreographer¥Oracle に変更します。
   b. ファイルの先頭にある指示に従い、createTablespace_Observer.sql スクリプト・ファイルを編集します。
   c. テーブル・スペース作成スクリプト・ファイルの先頭にある指示に従い、このスクリプト・ファイルを実行します。
   d. スクリプト出力にエラーがないことを確認します。 エラーが発生した場合は、dropTablespace_Observer.sql スクリプト・ファイルを使用してそのテーブル・スペースを除去できます。
3. 構成スクリプトがある Business Process Choreographer ディレクトリーに移動します。

Linux および UNIX プラットフォームの場合は、次のように入力します。

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

Windows プラットフォームの場合は、次のように入力します。

```
cd install_root¥ProcessChoreographer¥config
```

4. 299 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。
5. 以下のようにデータベースを準備します。 以下が表示された場合:

```
1) Prepare a database for the Event Collector and Observer
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 Observer
6) Administer Observer related user-defined functions

0) Exit Menu
```

以下を実行します。

a. オプション 1 を選択して、Event Collector アプリケーションおよび Observer 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 WebSphere Business Process Choreographer
Event Collector and Observer

Select the type of your database provider:

     c) Derby
     d) DB2 Universal
     i) DB2 iSeries
     7) DB2 V7 on z/OS
     8) DB2 V8/V9 on z/OS
     o) Oracle

     0) Exit Menu
```

b. o を入力して Oracle を選択します。

c. 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。

d. 以下が表示された場合:

```
Specify the database to be used.

Note: Database must already exist.

Specify the name of your database [BPEDB] :
```

データベースの SID 名を入力します。

e. 以下が表示された場合:

```
Specify the hostname where the oracle database resides: [localhost]
```

データベース・サーバーのホスト名または IP アドレスを入力します。

f. 以下が表示された場合:

```
Specify the port where the oracle listener is listening: [1521]
```

Oracle リスナーのポート番号を入力します。

g. 以下が表示された場合:

```
Specify userid to connect to the database 'database_name' [system] :
```

データベースに接続するユーザー ID を入力します。デフォルトは system です。

h. 以下が表示された場合:

```
Specify the password for userid 'user_ID' :
```

ユーザー ID のパスワードを入力します。

i. 以下が表示された場合:

```
Choose the implementation of the Observer user-defined functions.

  Note: The Java UDFs are more precise, but they require a jar file
  installed to the database.
  Visit the Observer 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。

以下のようなメッセージが表示されます。

```
Trying to connect to database 'database_name', using user 'user_ID'
Connected to 'database_name'
Checking for required tablespace(es) ['OBSVRTS', 'OBSVRLOB', 'OBSVRIDX']
All required tablespaces were found.
Loading the jar file 'install_root\lib\bpcodbutil.jar' into the database.
The jar file 'install_root\lib\bpcodbutil.jar' was successfully installed.

The setup of the database completed successfully.
```

6. 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。

結果

Business Process Choreographer Observer 用のデータベース・スキーマが作成されました。

関連概念

280 ページの『Business Process Choreographer Observer のユーザー定義関数』
Business Process Choreographer Observer 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 Observer では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

『Java ユーザー定義関数または SQL ユーザー定義関数の選択』
setupEventCollector ツールを使用するか、またはスクリプトを実行して、Business Process Choreographer Observer 用のデータベースで 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ことができます。

Java ユーザー定義関数または SQL ユーザー定義関数の選択

setupEventCollector ツールを使用するか、またはスクリプトを実行して、Business Process Choreographer Observer 用のデータベースで 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ことができます。

関連概念

280 ページの『Business Process Choreographer Observer のユーザー定義関数』
Business Process Choreographer Observer 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 Observer では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。

スクリプトを使用した 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、スクリプトを使用して、Business Process Choreographer Observer 用のデータベースで 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

SQL 実装ではなく、Java 実装の Business Process Choreographer Observer UDF を使用します。

手順

1. Java 実装の Business Process Choreographer Observer ユーザー定義関数 (UDF) を使用する場合は、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。
2. Java 実装の Business Process Choreographer Observer UDF を使用する場合は、jar ファイル `bpcodbutil.jar` をインストールします。

a. まだデータベースに接続していない場合、次のコマンドを DB2 コマンド行プロセッサーに入力して、データベースに接続します。

```
db2 connect to databaseName
```

b. 次のコマンドを入力して、jar ファイルをインストールします。

```
db2 call sqlj.install_jar('file:pathURL','schema.BPCODBUTIL')
```

ここで、*pathURL* は jar ファイルへの完全修飾 URL、*schema* は Business Process Choreographer データベースのスキーマの名前です。以下に例を示します。

- Linux および UNIX プラットフォームで、jar ファイルがディレクトリー /tmp にある場合、次のコマンドを入力する必要があります。

```
db2 call sqlj.install_jar('file:/tmp/bpcodbutil.jar',
    'schema.BPCODBUTIL')
```

- Windows プラットフォームで、jar ファイルがディレクトリー c:¥tmp にある場合、次のコマンドを入力する必要があります。

```
db2 call sqlj.install_jar('file:c:/tmp/bpcodbutil.jar',
    'schema.BPCODBUTIL')
```

注: jar ファイルを除去する場合は、次のコマンドを使用します。

```
db2 call sqlj.remove_jar('schema.BPCODBUTIL')
```

3. SQL 実装の UDF を除去します。 dropFunctions_Observer.sql スクリプト・ファイルの先頭にある指示に従い、このファイルを編集します。例えば、DB2 の場合は、DB2 のコマンド行プロセッサーで以下のコマンドを入力します。

```
db2 -tf dropFunctions_Observer.sql
```

4. UDF の Java 実装を作成します。 createFunctionsJava_Observer.sql スクリプト・ファイルの先頭にある指示に従い、このファイルを編集します。例えば、DB2 の場合は、DB2 のコマンド行プロセッサーで以下のコマンドを入力します。

```
db2 -tf createFunctionsJava_Observer.sql
```

結果

使用される UDF 実装に切り替わりました。

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 Observer 用のデータベースで 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

Derby データベースの場合、setupEventCollector は常に Java ベースの UDF を使用します。それ以外のデータベース・タイプの場合、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. 299 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 Observer
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 Observer
6) Administer Observer related user-defined functions

0) Exit Menu
```

2. オプション 6 を選択して、Observer 関連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
d) DB2 Universal
i) DB2 iSeries
7) DB2 V7 on z/OS
8) DB2 V8/V9 on z/OS
o) Oracle
```

3. DB2 for Linux、UNIX、または Windows、あるいは DB2 for z/OS のいずれかを使用している場合は、データベース・バージョンのオプション d、7、または 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

   JDBC ドライバーのディレクトリー

   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

   JDBC ドライバーのディレクトリー

   データベース・ユーザー ID

   パスワード

4. Oracle を使用している場合は、オプション o を選択します。
   a. 以下の接続情報を入力します。

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

データベース名

データベース・ユーザー ID

パスワード

JDBC ドライバーのディレクトリー

5. データベースとの接続が確立すると、Observer 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 Observer 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
```

注: 「アクティブ化」オプションは、Derby データベースには適用されません。

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 Observer 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) 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 Observer 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 Observer functions
     into the database
  2) Remove the jar file containing the Observer 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 (277 ページ) で表示されているメニューに戻ります。

結果

Business Process Choreographer Observer データベースでは、選択した UDF を使用します。

Business Process Choreographer Observer のユーザー定義関数

Business Process Choreographer Observer 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 Observer では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、275 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

275 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』
setupEventCollector ツールを使用するか、またはスクリプトを実行して、Business

Process Choreographer Observer 用のデータベースで 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ことができます。

249 ページの『SQL スクリプトを使用した Business Process Choreographer Observer のための DB2 for iSeries データベースの準備』
ここでは、createSchema_Observer.sql スクリプトを i5/OS qshell 環境で使用して、DB2 for iSeries データベースを準備する方法について説明します。

251 ページの『setupEventCollector ツールを使用した Business Process Choreographer Observer のための DB2 for iSeries 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を使用して、i5/OS qshell 環境内から DB2 for iSeries データベースを準備する方法について説明します。

253 ページの『リモート・システムからの setupEventCollector ツールを使用した DB2 for iSeries 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を使用して、リモートの Linux、Windows、または UNIX システムから Business Process Choreographer Observer のための DB2 for iSeries データベースを準備する方法について説明します。

256 ページの『USS における Business Process Choreographer Observer 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、z/OS マシン上の UNIX System Services (USS) で使用して、DB2 for z/OS データベースを作成する方法について説明します。

261 ページの『リモート・システムからの Business Process Choreographer Observer 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 Linux、UNIX、または Windows システムで使用して、DB2 for z/OS データベースを作成する方法について説明します。

270 ページの『SQL スクリプトを使用した Business Process Choreographer Observer のための Oracle データベースの準備』
ここでは、createTablespace_Observer.sql および createSchema_Observer.sql スクリプトを使用して、Oracle データベースを準備する方法について説明します。

272 ページの『setupEventCollector ツールを使用した Business Process Choreographer Observer のための Oracle 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、および createTablespace_Observer.sql スクリプトを使用して、Oracle データベースを準備する方法について説明します。

244 ページの『SQL スクリプトを使用した Business Process Choreographer Observer のための DB2 Universal Database の準備』
ここでは、createTablespace_Observer.sql および createSchema_Observer.sql スクリプトを使用して、DB2 Universal Database を Linux、UNIX、および Windows のプラットフォームに準備する方法について説明します。

245 ページの『setupEventCollector ツールを使用した Business Process Choreographer Observer のための DB2 Universal Database の準備』
ここでは、対話式メニュー方式のツールおよび createTablespace_Observer.sql スクリプトを使用して、Linux、UNIX、および Windows プラットフォームに DB2 Universal Database を準備する方法について説明します。

関連資料

299 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 Event Collector アプリケーションの構成

対話式ツールまたは管理コンソールを使用して、Event Collector アプリケーションをインストールし、構成します。

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. 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。

   Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

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

   Windows プラットフォームの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥config
   ```

2. 299 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。 例えば、ツールを開始して、server1 というサーバーを操作する場合は、次のいずれかのコマンドを入力します。

   Linux および UNIX プラットフォームの場合:

   ```
   setupEventCollector.sh -server server1
   ```

   i5/OS プラットフォームの場合:

   ```
   setupEventCollector -server server1
   ```

   Windows プラットフォームの場合:

   ```
   setupEventCollector.bat -server server1
   ```

「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 Observer
  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 Observer
  6) Administer Observer 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. ツールがデプロイメント・ターゲット上の既存の 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 ロギングが使用可能に設定されていない場合は、以下のメッセージが表示されます。

   ```
   Checking if CEI event logging is enabled ...

   Warning: The Business process container of server_name has CEI event
   logging disabled.
   To allow the Event Collector to work correctly, CEI event logging is required.
   Do you want to enable the CEI event logging on server_name? (y/n)
   ```

   - 指定のサーバー上のCEI ロギングをスクリプトで使用可能にする場合は、y を入力します。
   - 指定のサーバー上のCEI ロギングをスクリプトで使用可能にしない場合は、n を入力します。

   注: Business Process Choreographer Observer での作業を開始するときには、必ず CEI ロギングを使用可能にしてください。

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。エラーが発生していた場合は、n を入力して変更を破棄し、元の構成を維持します。setupEventCollector.log という名前のログ・ファイルを確認してください。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。

   例えば、Windows では、プロファイル名が myServer で、プロファイルが *install_root*¥profiles に保管されている場合、ログ・ファイルは *install_root*¥profiles¥myServer¥logs に置かれます。

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。

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

Business Process Choreographer Observer 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**」 → 「***cluster_name***」をクリックするか、または「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer**」を展開し、「**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・インスタンス名。
      - 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**」 → 「**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。

結果

Business Process Choreographer Event Collector が構成されています。

Business Process Choreographer Event Collector:

Event Collector は、Business Process Choreographer Observer の使用前に構成しておく必要があります。

この管理コンソール・ページを表示するには、「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックするか、または「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer**」、「**Event Collector の構成 (Event Collector Configuration)**」をクリックします。

データ・ソース:

このセクションでは、Business Process Choreographer Event Collector のデータ・ソースを構成します。

編集:

データ・ソースを編集する場合にクリックします。

テスト接続:

選択されているデータ・ソースへの接続をテストします。

データベース・インスタンス:

データベースの名前。

プロパティー	**値**
データ型	ストリング
デフォルト	${USER_INSTALL_ROOT}¥databases¥BPOBEC00

スキーマ名:

データベース・スキーマの名前。

プロパティー	**値**
データ型	ストリング
デフォルト	なし

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には、データベースが存在する必要があり、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。

実動システムでは、このオプションの使用は推奨されません。このオプションを選択しない場合、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。

プロパティー	値
データ型	チェック・ボックス
デフォルト	選択

ユーザー名:

データベースに接続し、データを変更する権限を持つユーザー ID。

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

プロパティー	値
データ型	ストリング
デフォルト	現在ログオンしているユーザー。

パスワード:

データ・ソースのユーザー ID のパスワード。

プロパティー	値
データ型	ストリング
デフォルト	なし

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。

プロパティー	値
データ型	ストリング
デフォルト	なし

ポート:

データベース・サーバーが使用するポート番号。

プロパティー	値
データ型	ストリング
デフォルト	どの JDBC プロバイダーを選択するかによって異なります。

プロバイダー:

JDBC プロバイダー。

プロパティー	値
データ型	ドロップダウン・リスト
デフォルト	DERBY_EMBEDDED

JMS ユーザー名:

プロパティー	値
データ型	ストリング
デフォルト	Business Process Choreographer JMS ユーザーに指定されたユーザー ID。

JMS パスワード:

プロパティー	値
データ型	ストリング
デフォルト	Business Process Choreographer JMS ユーザーに指定されたパスワード。

監視ターゲット:

このセクションでは、Event Collector のターゲットを指定します。

プロパティー	値
データ型	ラジオ・ボタン
選択項目	• **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**

管理対象の Business Process Choreographer コンテナー:

構成されている Business Process Choreographer コンテナーを選択します。

プロパティー	値
データ型	ドロップダウン・リスト
内容	すべての Business Process Choreographer 構成

イベント・グループ・プロファイル・リスト:

グループ・プロファイルを選択します。

プロパティー	値
データ型	ドロップダウン・リスト

イベント・グループ・プロファイル・リストの入力 (Enter event group profile list):

イベント・グループ・プロファイル・リストに入力します。

プロパティー	値
データ型	ストリング
デフォルト	なし

Business Process Choreographer Observer アプリケーションの構成

ツールまたは管理コンソールのいずれかを使用して、Business Process Choreographer Observer アプリケーションを構成できます。

始める前に

Business Process Choreographer Event Collector を構成済みです。

このタスクについて

Business Process Choreographer Observer を構成するには、以下のいずれかを実行します。

setupObserver ツールを使用した Business Process Choreographer Observer 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、Business Process Choreographer Observer アプリケーションのインスタンスをインストールし、特定の Event Collector のデータ・ソースに接続するようこのインスタンスを構成する方法について説明します。

始める前に

Business Process Choreographer Observer データベースの準備を完了し、管理コンソールを使用してこのデータベースのデータ・ソースを作成しました。

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。

   Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

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

   Windows プラットフォームの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥config
   ```

2. 302 ページの『setupObserver ツール』の説明に従って、このツールを開始して Observer 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) Install the Observer application
   2) Remove the Observer application and related objects
   3) Change configuration settings of an installed Observer

   0) Exit Menu
   ```

3. オプション 1 を選択して、Business Process Choreographer Observer をインストールします。 以下が表示されます。

```
Create required objects and install the WebSphere Business Process
Choreographer Observer application ...
```

4. 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'.
   ```

5. 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
   ```

6. 以下が表示された場合:

   ```
   Specify the JNDI name of the database containing the event tables.
   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
   ```

7. 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 BPC Observer installed successfully!
   ```

8. 「Do you want to save the changes? (y/n)」というプロンプトが出されたときに、エラー・メッセージがなければ、y を入力して構成を保存します。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。setupObserver.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)。例えば、Windows では、プロファイル名が myServer で、プロファイルが *install_root*¥profiles に保管されている場合、ログ・ファイルは *install_root*¥profiles¥myServer¥logs に置かれます。
9. 0 を入力して、メニューを終了します。

結果

Business Process Choreographer Observer 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 Observer の構成

ここでは、管理コンソールを使用して Business Process Choreographer Observer アプリケーションのインスタンスをインストールし、特定の Event Collector 用のデータ・ソースに接続するようこのアプリケーションを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 の構成が完了しています。

手順

1. 管理コンソールで、次のようにして Business Process Choreographer Observer の構成ページに移動します。 「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックするか、または「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer**」を展開し、「**Business Process Choreographer Observer**」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. 「**追加**」をクリックします。
   b. 以下のフィールドで値を入力するか、または選択します。
      - このインスタンスのコンテキスト・ルート
      - この Business Process Choreographer Event Collector からのモニター・データの視覚化

      これらの構成パラメーターについて詳しくは、 168 ページの『Business Process Choreographer Observer 設定』を参照してください。
   c. 「**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。
   d. 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。
   e. 「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」をクリックしてアプリケーション開始し、アプリケーション `BPCObserver_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。

結果

Business Process Choreographer Observer が構成され、いつでも使用できます。

次のタスク

このタスクを繰り返すことにより、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 Observer のインスタンスを構成することができます。ただし、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 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 Observer 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 Observer 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、『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、334 ページの『管理コンソールを使用した Common Base Event および監査証跡の使用可能化』を実行します。

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。

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、Business Process Choreographer で Common Event Infrastructure (CEI) イベントまたは監査イベント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。

スクリプトの実行

setStateObserver スクリプトを実行するには、次のように入力します。

Linux および UNIX プラットフォームの場合は、次のように入力します。

```
install_root/bin/wsadmin.sh
       -f install_root/ProcessChoreographer/config/setStateObserver.py
```

i5/OS プラットフォームの場合は、次のように入力します。

```
install_root/bin/wsadmin
       -f install_root/ProcessChoreographer/config/setStateObserver.py
```

Windows プラットフォームの場合は、次のように入力します。

```
install_root¥bin¥wsadmin.bat
       -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 | CEI;AuditLog)
オプションで、CEI ロギング、監査ロギング、またはこの両方を使用可能にするかどうかを指定できます。

-disable (CEI | AuditLog | CEI;AuditLog)
オプションで、CEI ロギング、監査ロギング、またはこの両方を使用不可にするかどうかを指定できます。

-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*
ここで、*profileName* は使用するプロファイルの名前です。

-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

Linux または UNIX プラットフォームで 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
```

注: Windows では wsadmin.bat を使用し、i5/OS では wsadmin を使用します。

Business Process Choreographer Observer の構成パラメーターの変更

Business Process Choreographer Observer 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 Observer 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。
BPCEventTransformer EventCount	整数/イベント数	500	収集されたイベントを Observer アプリケーション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、Observer 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 Observer 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 Observer 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、Observer 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。

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。Observer のレポートでは、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、Observer 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. 299 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 Observer
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 Observer
6) Administer Observer 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。

Observer 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	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。 これは、Event Collector の値と一致している必要があります。
ReportAtSnapshotRange	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。 ReportAtSnapshotRange は、スナップショット・レポートに含めることができるイベントの期間を定義します。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	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、Observer 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。

Observer の構成パラメーターの変更

Observer パラメーターを変更するには、以下を実行します。

1. 302 ページの『setupObserver ツール』の説明に従って、このツールを開始して Observer 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) Install the Observer application
2) Remove the Observer application and related objects
3) Change configuration settings of an installed Observer

0) Exit Menu
```

2. オプション 3 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) ReportAtSnapshotRange
2) ObserverCreateTables
3) 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、BPCObserver アプリケーションを再始動します。

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。

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。

Linux、UNIX、および i5/OS プラットフォームの場合: *install_root*/ProcessChoreographer/config。

Windows プラットフォームの場合: *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。
- i5/OS では、このツールを実行するには qshell を使用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*
グローバル・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、次のコマンドのいずれかを入力します。

Linux および UNIX プラットフォームの場合:

```
setupEventCollector.sh -server server1
```

i5/OS プラットフォームの場合:

```
setupEventCollector -server server1
```

Windows プラットフォームの場合:

```
setupEventCollector.bat -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 Observer
 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 Observer
 6) Administer Observer 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

280 ページの『Business Process Choreographer Observer のユーザー定義関数』
Business Process Choreographer Observer 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 Observer では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

282 ページの『Business Process Choreographer Event Collector アプリケーションの構成』
対話式ツールまたは管理コンソールを使用して、Event Collector アプリケーションをインストールし、構成します。

245 ページの『setupEventCollector ツールを使用した Business Process Choreographer Observer のための DB2 Universal Database の準備』
ここでは、対話式メニュー方式のツールおよび createTablespace_Observer.sql スクリプトを使用して、Linux、UNIX、および Windows プラットフォームに DB2 Universal Database を準備する方法について説明します。

251 ページの『setupEventCollector ツールを使用した Business Process Choreographer Observer のための DB2 for iSeries 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を使用して、i5/OS qshell 環境内から DB2 for iSeries データベースを準備する方法について説明します。

256 ページの『USS における Business Process Choreographer Observer 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、z/OS マシン上の UNIX System Services (USS) で使用して、DB2 for z/OS データベースを作成する方法について説明します。

261 ページの『リモート・システムからの Business Process Choreographer Observer 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 Linux、UNIX、または Windows システムで使用して、DB2 for z/OS データベースを作成する方法について説明します。

267 ページの『setupEventCollector ツールを使用した Business Process Choreographer Observer 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 Derby データベースを準備する方法について説明します。

272 ページの『setupEventCollector ツールを使用した Business Process Choreographer Observer のための Oracle 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、および createTablespace_Observer.sql スクリプトを使用して、Oracle データベースを準備する方法について説明します。

275 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』
setupEventCollector ツールを使用するか、またはスクリプトを実行して、Business Process Choreographer Observer 用のデータベースで 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ことができます。

315 ページの『スクリプトを使用した Business Process Choreographer 構成の削除』
このタスクを使用して、Business Flow Manager、Human Task Manager、Business Process Choreographer Explorer、および Business Process Choreographer Observer 構成と、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。

関連資料

294 ページの『Business Process Choreographer Observer の構成パラメーターの変更』
Business Process Choreographer Observer 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。

setupObserver ツール

setupObserver では、Business Process Choreographer Observer アプリケーションを対話式に構成または除去し、構成パラメーターを変更することができます。このツールでは wsadmin スクリプトが使用されます。

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。

Linux、UNIX、および i5/OS プラットフォームの場合: *install_root*/ProcessChoreographer/config。

Windows プラットフォームの場合: *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。
- i5/OS では、このツールを実行するには qshell を使用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*
グローバル・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、次のコマンドのいずれかを入力します。

Linux および UNIX プラットフォームの場合:

```
setupObserver.sh -server server1
```

i5/OS プラットフォームの場合:

```
setupObserver -server server1
```

Windows プラットフォームの場合:

```
setupObserver.bat -server server1
```

「コマンド (Commands)」メニューが表示されます。

```
1) Install the Observer application
2) Remove the Observer application and related objects
3) Change configuration settings of an installed Observer

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。

関連タスク

289 ページの『setupObserver ツールを使用した Business Process Choreographer Observer 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、Business Process Choreographer Observer アプリケーションのインスタンスをインストールし、特定の Event Collector のデータ・ソースに接続するようこのインスタンスを構成する方法について説明します。

315 ページの『スクリプトを使用した Business Process Choreographer 構成の削除』
このタスクを使用して、Business Flow Manager、Human Task Manager、Business Process Choreographer Explorer、および Business Process Choreographer Observer 構成と、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。

関連資料

294 ページの『Business Process Choreographer Observer の構成パラメーターの変更』
Business Process Choreographer Observer 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。

Business Process Choreographer Observer の検査

Business Process Choreographer Observer をインストールして構成した後に、Observer が正しく動作することを確認します。

始める前に

Business Process Choreographer Observer データベースは最初は空です。

手順

1. ビジネス・イベントを生成するアクションを実行します。例えば、Business Process Choreographer Explorer を使用して、プロセス・インスタンスを開始します。
2. ブラウザーで URL `http://host:port/context_root/` を開いて、Business Process Choreographer Observer 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* は一般には `bpcobserver` です。

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。 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、 294 ページの『Business Process Choreographer Observer の構成パラメーターの変更』を参照してください。

4. 問題がある場合は、 741 ページの『Business Process Choreographer Observer のトラブルシューティング』を参照してください。

結果

Business Process Choreographer Observer が作動しています。

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

113 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・インストールが、そのクライアントが接続する 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) 「**クライアント・インストール**」オプションを使用して WebSphere Process Server クライアントをインストールします。

         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、「**カスタム・プロファイル**」を選択します。

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 クライアントのノード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) 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。

3) 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) 管理コンソールで、「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」をクリックします。

b) 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
```

4) 変更を保存して、同期化します。

5) 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有効にするには、『リモート成果物ローダー』を参照してください。

2. 「クロス・セル」シナリオの場合、WebSphere Process Server クライアントは、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は配置されません。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 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できます。最低条件として、この Network Deployment セルには WebSphere Application Server Deployment Manager のみ必要です。このような環境に WebSphere Process Server クライアント・インストールをセットアップして、Business Process Choreographer 構成が存在するセルにアクセスできるように構成するには、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) 「**クライアント・インストール**」オプションを使用して WebSphere Process Server クライアントをインストールします。

         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、「**カスタム・プロファイル**」を選択します。

         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 クライアントのノード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ネーム・スペース・バインディング (複数可) を新規に定義します。

      1) 管理コンソールを使用して、クライアント・セルで「**環境**」 → 「**ネーミング (Naming)**」 → 「**ネーム・スペース・バインディング (Name Space Bindings)**」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、ネーム・スペース内のスラッシュを下線文字に置き換えることで、ネーム・スペース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

         というネーム・スペースは、

         bpc_*remoteCellName_remoteNode_remoteServer*_com_ibm_bpe_api_BusinessFlowManagerHome

         というバインディング ID 名になります。
      ii. バインディングに使用されるクライアントのネーム・スペース。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. 接続先の Business Process Choreographer 構成をホストするサーバーまたはクラスターのネーミング・サーバーのプロバイダー URL。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***」 をクリックします。
b) 「**参照**」セクションで、「**EJB 参照**」を選択します。
c) 定義した各ネーム・スペースに「**ターゲット・リソースの JNDI 名 (Target Resource JNDI Name)**」 フィールドが 1 つずつあります。ステップ 2c3bii (308 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。
d) 変更を保存して、同期化します。
e) クライアント・アプリケーションを再始動します。

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

445 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、 および Java Message Service (JMS) 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

238 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。

445 ページの『第 10 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。

448 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』
EJB クライアント・アプリケーションは、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。

関連情報

ソフトウェアのインストール

プロファイル管理ツールの開始

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**」 → 「**エンタープライズ・アプリケーション**」を選択し、`BPEContainer_`*`scope`* および `TaskContainer_`*`scope`* で始まる名前のアプリケーションが開始されていることを確認してください。

   アプリケーション・サーバーで 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。

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。 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。

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 Observer、Business Process Choreographer Explorer、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。このサーバーをネットワーク・デプロイメント環境に統合します。

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 Network Deployment インフォメーション・センターで、『addNode コマンド』を参照してください。 例えば、デプロイメント・マネージャーのホスト名が dmgr_host で、ポート dmgr_port を使用する場合、次のコマンドを入力します。

   ```
   addNode dmgr_host dmgr_port -includeapps -includebuses
   ```

   例えば、デプロイメント・マネージャーのホスト名が any.hostname.com で、ポート 9043 を使用し、プロファイル名が ProcSvr07、ユーザー ID が admin、およびパスワードが secret の場合、次のコマンドを入力します。

   ```
   addNode any.hostname.com 9043 -profileName ProcSvr07 -username admin
           -password secret -includeapps -includebuses
   ```

   前提条件のいずれかを満たしていない場合、エラー・メッセージが表示されます。満たしている場合は、サーバーは停止され、新規のデプロイメント・マネージャー・セルに統合されます。
3. 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。

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、Business Process Choreographer Observer、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・プロセスを含むエンタープライズ・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します。
3. ヒューマン・タスクまたはビジネス・プロセスを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します。
4. 以下のいずれかのアクションを実行します。
   - Business Process Choreographer 構成、Business Process Choreographer Explorer、Business Process Choreographer Observer、Event Collector、および関連するリソースを除去する場合は、『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、319 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。

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

このタスクを使用して、Business Flow Manager、Human Task Manager、Business Process Choreographer Explorer、および Business Process Choreographer Observer 構成と、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 構成を削除するには、その前にすべてのプロセス・テンプレートおよびタスク・テンプレートを停止し、すべてのプロセス・インスタンスおよびタスク・インスタンスを削除してから、ビジネス・プロセスまたはヒューマン・タスクを含むすべてのエンタープライズ・アプリケーションを停止して除去する必要があります。

手順

1. Business Process Choreographer `config` ディレクトリーに移動します。

   Windows プラットフォームの場合は、次のコマンドを入力します。

   ```
   cd install_root¥ProcessChoreographer¥config
   ```

Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

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

2. スクリプト bpeunconfig.jacl を実行します。 次のケースでは、該当するオプションも指定します。
   - スタンドアロン・サーバーの場合は、アプリケーション・サーバーを停止して -conntype NONE オプションを使用します。このステップでは、すべての Derby 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。
   - WebSphere 管理セキュリティーが有効になっている場合は、ユーザー ID およびパスワードも指定します。

     ```
     -user userID  -password password
     ```
   - デフォルト・プロファイルから構成を削除しない場合は、以下のようにしてプロファイル名も指定します。

     ```
     -profileName profileName
     ```

オプション	説明
Linux または UNIX 上の単一サーバーの場合	次のコマンドを入力します。 *install_root*/bin/wsadmin.sh -f bpeunconfig.jacl -server *Server* -node *Node* [-deleteDB *deleteDatabase*]
Windows 上の単一サーバーの場合	次のコマンドを入力します。 *install_root*¥bin¥wsadmin.bat -f bpeunconfig.jacl -server *Server* -node *Node* [-deleteDB *deleteDatabase*]
i5/OS 上の単一サーバーの場合	次のコマンドを入力します。 *install_root*/bin/wsadmin -f bpeunconfig.jacl -server *Server* -node *Node* [-deleteDB *deleteDatabase*]
Linux または UNIX 上のクラスターの場合	次のコマンドを入力します。 *install_root*/bin/wsadmin.sh -f bpeunconfig.jacl -cluster *Cluster*

オプション	説明
Windows 上のクラスターの場合	次のコマンドを入力します。 *install_root*¥bin¥wsadmin.bat -f bpeunconfig.jacl -cluster *Cluster*
i5/OS 上のクラスターの場合	次のコマンドを入力します。 *install_root*/bin/wsadmin -f bpeunconfig.jacl -cluster *Cluster*

各部の意味は、次のとおりです。

userID ユーザー ID。

password
ユーザー ID のパスワード。

profileName
構成中のプロファイルの名前です。デフォルト・プロファイルを構成する場合は、これはオプションです。

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。

deleteDatabase
Derby Embedded データベースおよび FILESTORE ディレクトリーを削除するかどうかを指定するブール値。

yes
no

このオプションを使用するには、サーバーが実行中であってはいけません。Derby Embedded 以外のデータベースがなく、このオプションを使用する場合は、スクリプトの実行後にステップ 4 (318 ページ) までスキップできます。

必須パラメーターを省くと、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。
- Derby データベースが Business Process Choreographer データベース用に使用されている場合は、実行中のアプリケーション・サーバーによってデータベースがロックされていない限り、オプションとして bpeunconfig.jacl スクリプ

トによりそのデータベースを削除することもできます。データベースがロックされている場合は、サーバーを停止してから -conntype NONE オプションを使用します。

- Derby データベースがメッセージング・データベースの場合は、実行中のアプリケーション・サーバーによってデータベースがロックされていない限り、オプションとして bpeunconfig.jacl スクリプトによりそのデータベースを削除することもできます。データベースがロックされている場合は、サーバーを停止してから -conntype NONE オプションを使用します。
- 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。
- Business Process Choreographer Observer データベースを除去するには、 299 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 Observer**」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。
   - 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

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、認証別名など) が除去されました。

ツールを使用した Business Process Choreographer Observer および Event Collector の除去

Business Process Choreographer Observer アプリケーションと Event Collector アプリケーション、および関連するリソースをサーバーまたはクラスターから除去します。

手順

1. Business Process Choreographer Observer を構成していた場合は、 302 ページの『setupObserver ツール』 を実行し、-remove オプションを指定します。
2. Business Process Choreographer Event Collector アプリケーションを構成していた場合は、 299 ページの『setupEventCollector ツール』 を実行して、-remove オプションを指定します。
3. オプション: Java ユーザー定義関数をインストールしていた場合は、それらを除去します。
4. オプション: データベースを削除します。

結果

Business Process Choreographer Observer アプリケーションおよび Event Collector アプリケーションが除去されました。

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、Business Process Choreographer Observer、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、すべてのプロセス・テンプレートおよびタスク・テンプレートを停止し、すべてのタスクおよびプロセス・インスタンスを削除してから、ビジネス・プロセスまたはヒューマン・タスクを含むすべてのエンタープライズ・アプリケーションを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**」 → 「**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。

      ここで、*scope* の値は構成によって異なります。

- Business Process Choreographer がアプリケーション・サーバー上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、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。 「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、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。

注: Business Process Choreographer Observer アプリケーションおよび Event Collector アプリケーションのアンインストールについては、ステップ 10 (325 ページ) で説明します。

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 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: 該当する接続ファクトリーとキューを除去します。
- デフォルトのメッセージングの場合は、接続ファクトリーを除去する前に、関連する認証データの別名をメモしておきます。その後、JMS 接続ファクトリーと JMS キューを除去します。

1) 「**リソース**」 → 「**JMS**」 → 「**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。

ビジネス・プロセス・コンテナーの場合は、通常、JNDI 名は次のようになります。

```
jms/BPECF
jms/BPECFC
jms/BFMJMSReplyCF
jms/BPEIntQueue
jms/BPERetQueue
jms/BPEHldQueue
jms/BFMJMSAPIQueue
jms/BFMJMSCallbackQueue
jms/BFMJMSReplyQueue
```

ヒューマン・タスク・コンテナーの場合は、通常、JNDI 名は次のようになります。

```
jms/HTMCF
jms/HTMIntQueue
jms/HTMHldQueue
```

d. オプション: WebSphere デフォルト・メッセージングを JMS プロバイダーとして使用している場合は、活動化仕様を削除します。

1) 「**リソース**」 → 「**JMS**」 → 「**活動化仕様**」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) 以下の活動化仕様を削除します。

```
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**」 → 「**アプリケーション・サーバー**」 → 「***serverName***」をクリックします。
2) 「通信」の下で「**メッセージング**」 → 「**メッセージ・リスナー・サービス**」 → 「**リスナー・ポート**」をクリックします。
3) 「アプリケーション・サーバー」ペインで、次のリスナー・ポートを削除します。

```
BPEInternalListenerPort
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**」 → 「**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**」をクリックし、「**認証**」セクションで「**Java 認証・承認サービス**」を展開し、「**J2C 認証データ**」をクリックします。
2) ステップ 2 (320 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.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 (320 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 からマイグレーション**しなかった**場合、その名前は `BPC_Auth_Alias` で、ネットワーク・デプロイメント環境内のすべての 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 からマイグレーションしていた場合は、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 (Work managers)**」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 (322 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**」 → 「**アプリケーション・サーバー**」 → 「***serverName***」をクリックします。

2) 「**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。

3) 「アプリケーション・サーバー」ペインで、作業域区画「`BPECompensation`」を選択して削除します。

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 (319 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. 「**トポロジー**」セクションの下で「**サービス統合**」 → 「**バス**」 → 「**BPC.*cellName*.Bus**」をクリックし、「**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。サーバー上では、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**」と進み、「**トポロジー**」セクションの下で、「**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 で書き留めておいたメッセージ・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. 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。
6. 「**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。
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: 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。

9. 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 (324 ページ) 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 (324 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 (324 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。 通常、Derby データベースのロケーションは以下のとおりです。
      - Linux、UNIX、および i5/OS プラットフォームの場合:

        profile_root/databases/com.ibm.ws.sib/
        nodeName.*serverName*-BPC.*cellName*.Bus
      - Windows プラットフォームの場合:

        profile_root¥databases¥com.ibm.ws.sib¥
        nodeName.*serverName*-BPC.*cellName*.Bus
10. オプション: Business Process Choreographer Observer を構成した場合、以下を実行します。
    a. Observer アプリケーション・インスタンスの場合は、327 ページの『管理コンソールを使用した Business Process Choreographer Observer の除去』。
    b. Event Collector アプリケーション・インスタンスの場合は、『管理コンソールを使用した Business Process Choreographer Event Collector の除去』。

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 Observer によって必要とされる関連するリソースを除去します。

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**」 → 「**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 (Service integration)**」 → 「**バス (Buses)**」 → **CommonEventInfrastructure_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. 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。
5. 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。
6. JMS 活動化仕様を削除します。
   a. 「**リソース**」 → 「**JMS**」 → 「**活動化仕様**」をクリックします。
   b. 以下の活動化仕様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。
7. 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。
8. 構成をバージョン 6.0.2 からマイグレーションした場合は、認証データ別名を削除します。

a. 「**セキュリティー**」 → 「**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**」 → 「**認証**」 → 「**Java 認証・承認サービス**」 → 「**J2C 認証データ**」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。

10. Windows プラットフォームの場合は *install_root*¥dbscripts¥ProcessChoreographer¥*database_type* ディレクトリーにある以下のスクリプトを実行し、Linux、UNIX、および i5/OS プラットフォームの場合は *install_root*/dbscripts/ProcessChoreographer/*database_type* ディレクトリーにある以下のスクリプトを実行して、Business Process Choreographer Observer によって使用されるデータベース、スキーマ、およびテーブル・スペースを除去します。

- dropSchema_Observer.sql
- dropTablespace_Observer.sql

結果

Business Process Choreographer Event Collector 構成が除去されました。

管理コンソールを使用した Business Process Choreographer Observer の除去

このタスクを使用して、Business Process Choreographer Observer 構成および関連するリソースを除去します。

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**」 → 「**エンタープライズ・アプリケーション**」を選択します。

2. Business Process Choreographer Observer インスタンスを見つけます。

   名前が `BPCObserver_scope` で始まるアプリケーションを探します。

   - Business Process Choreographer Observer がアプリケーション・サーバーにインストールされている場合は、*scope* の値は `nodeName_serverName` になります。
   - Business Process Choreographer Observer がクラスターにインストールされている場合は、*scope* の値は `clusterName` になります。

   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。

3. Business Process Choreographer Observer アプリケーションをアンインストールします。 削除するアプリケーション・インスタンスを選択して、「**アンインストール**」 → 「**OK**」 → 「**保管**」をクリックします。

結果

Business Process Choreographer Observer 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。

管理コンソールによる Business Process Choreographer の管理

管理コンソールを使用して実行可能な管理操作について説明します。

サーバーの補正サービスの管理

管理コンソールを使用して、アプリケーション・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。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。

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. 管理コンソールを表示します。
2. ナビゲーション・ペインで、「**サーバー**」 → 「**アプリケーション・サーバー**」 → ***server_name*** をクリックします。
3. 「構成」タブの「コンテナーの設定」の下で、「**コンテナー・サービス (Container services)**」 → 「**補正サービス (Compensation service)**」をクリックします。 このアクションにより、パネルに補正サービス・プロパティーが表示されます。「**サーバー始動時にサービスを使用可能に設定**」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。

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. 保留キューおよび保存キューにメッセージがいくつあるかを確認します。
   a. 「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスター上で構成されている場合は、「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックします。
   b. 「**構成**」タブの「ビジネス・インテグレーション」の下の「**Business Process Choreographer**」をクリックします。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。
2. 保留キューまたは保存キューにメッセージが含まれている場合は、内部作業キューへメッセージを移動できます。

   以下のいずれかのオプションをクリックします。
   - ビジネス・プロセスの場合: 「**保留キューの再生**」および「**保存キューの再生**」
   - ヒューマン・タスクの場合: 「**保留キューの再生**」

   注: WebSphere 管理セキュリティーが使用可能になっている場合は、再生ボタンはオペレーター権限を持つユーザーに対してのみ表示されます。

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

34 ページの『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。

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. 該当するアプリケーション・サーバーを選択します。

   「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスター上で構成されている場合は、「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックします。
2. メッセージ数を最新表示します。
   a. 「**構成**」タブの「ビジネス・インテグレーション」の下の「**Business Process Choreographer**」をクリックします。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。

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。

関連概念

34 ページの『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。

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**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスター上で構成されている場合は、「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックします。
2. 「**構成**」タブの「ビジネス・インテグレーション」の下で、「**Business Process Choreographer**」 → 「**Human Task Manager**」をクリックします。
3. 「**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使用可能になっている場合は、オペレーター権限を持つユーザーに対してのみ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。以下にリストされた代替方式を使用することを考慮してください。

結果

関連概念

102 ページの『担当者割り当て基準および担当者解決結果の管理』
タスク許可のロールに関連付けられている担当者割り当て基準は、デプロイ済みのタスク・テンプレートまたはタスク・インスタンスの存続期間中は常に有効です。

関連タスク

355 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。

337 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。

管理コンソールを使用した Common Base Event および監査証跡の使用可能化

このタスクを使用して、Business Process Choreographer イベントを Common Event Infrastructure に Common Base Event として放出するか、監査証跡に保管するか、またはその両方を行えるようにします。

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、監査ログの状態監視者ロギング、または構成の Common Event Infrastructure ロギングを変更します。

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**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスター上で構成されている場合は、「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックします。
   b. 「**構成**」タブの「ビジネス・インテグレーション」の下の「**Business Process Choreographer**」をクリックします。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。どちらかいずれか、またはその両方を使用可能にしたり、使用不可にすることができます。

   Common Event Infrastructure ロギングの使用可能化
   このチェック・ボックスを選択して、Common Event Infrastructure に基づくイベント放出を使用可能にします。

   監査ロギングの使用可能化
   このチェック・ボックスを選択して、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。
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。

結果

状態監視者は、必要に応じて設定します。変更は、サーバーの再始動後に有効になります。

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、監査ログの状態監視者ロギング、またはセッションの Common Event Infrastructure ロギングを変更します。

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。選択した設定は、次にサーバーを開始するまで有効です。

以下のように、セッション・インフラストラクチャーを変更します。

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**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスター上で構成されている場合は、「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックします。
   b. 「**構成**」タブの「ビジネス・インテグレーション」の下の「**Business Process Choreographer**」をクリックします。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。どちらかいずれか、またはその両方を使用可能にしたり、使用不可にすることができます。

   Common Event Infrastructure ロギングの使用可能化
   このチェック・ボックスを選択して、Common Event Infrastructure に基づくイベント放出を使用可能にします。

   監査ロギングの使用可能化
   このチェック・ボックスを選択して、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。
3. 「**OK**」をクリックして変更を受け入れます。

結果

状態監視者は、必要に応じて設定します。

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**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」をクリックするか、または Business Process Choreographer がクラスター上で構成されている場合は、「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」をクリックします。
   b. 「**構成**」タブの「ビジネス・インテグレーション」の下で、「**Business Process Choreographer**」 → 「**Human Task 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。
      - 設定を永続的に変更するには、「**構成**」タブをクリックします。変更は、アプリケーション・サーバーの再始動後に有効になります。
      - 設定を一時的に変更するには、「**ランタイム**」タブをクリックします。変更は直ちに有効になりますが、アプリケーション・サーバーの次回の再始動時にリセットされます。
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. 「**OK**」をクリックします。
5. 変更を保存します。「構成」タブで実行し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。

関連概念

102 ページの『担当者割り当て基準および担当者解決結果の管理』
タスク許可のロールに関連付けられている担当者割り当て基準は、デプロイ済みのタスク・テンプレートまたはタスク・インスタンスの存続期間中は常に有効です。

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。

関連資料

292 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』
ここでは、setStateObserver.py スクリプトを使用して、Business Process Choreographer で Common Event Infrastructure (CEI) イベントまたは監査イベントを使用可能または使用不可にする方法を説明します。

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。

- 監査ログ・エントリーの削除に使用するアプリケーション・サーバーが稼働している必要があります。つまり、サーバー接続が必要であるため、wsadmin の `-conntype none` オプションは使用できません。
- 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設定されている場合は、オペレーター権限を持っている必要があります。

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。

   Windows プラットフォームの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥admin
   ```

   Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

   ```
   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。

Windows プラットフォームの場合は、次のコマンドを 1 つ以上入力します。コマンド間の相違を強調して表示しています。

```
install_root¥bin¥wsadmin -f deleteAuditLog.py
                         -server server_name
                         [-profileName profileName]
                         [options]

install_root¥bin¥wsadmin -f deleteAuditLog.py
                         -node node_name
                         -server server_name
                         [-profileName profileName]
                         [options]

install_root¥bin¥wsadmin -f deleteAuditLog.py
                         -cluster cluster_name
                         [-profileName profileName]
                         [options]
```

Linux および UNIX プラットフォームの場合は、次のコマンドを 1 つ以上入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin.sh -f deleteAuditLog.py
                         -server server_name
                         [-profileName profileName]
                         [options]

install_root/bin/wsadmin.sh -f deleteAuditLog.py
                         -node node_name
                         -server server_name
                         [-profileName profileName]
                         [options]

install_root/bin/wsadmin.sh -f deleteAuditLog.py
                         -cluster cluster_name
                         [-profileName profileName]
                         [options]
```

i5/OS プラットフォームの場合は、次のコマンドを 1 つ以上入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin -f deleteAuditLog.py
                         -server server_name
                         [-profileName profileName]
                         [options]

install_root/bin/wsadmin -f deleteAuditLog.py
                         -node node_name
                         -server server_name
                         [-profileName profileName]
                         [options]

install_root/bin/wsadmin -f deleteAuditLog.py
                         -cluster cluster_name
                         [-profileName profileName]
                         [options]
```

各部の意味は、次のとおりです。

-cluster *cluster_name*
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 *server_name*
サーバーの名前。クラスター名が指定されていない場合は必須です。

-profileName *profileName*
ユーザー定義プロファイルの名前。デフォルト・プロファイルで作業していない場合、このオプションを指定します。

使用可能なオプションは以下のとおりです。

-all
データベース内の監査ログ・エントリーをすべて削除します。複数のトランザクションで削除が実行されます。各トランザクションでは、slice パラメーターで指定されたエントリー数またはデフォルト数が削除されます。

-time *timestamp*
timestamp に指定した時刻より古い監査ログ・エントリーすべてが削除されます。使用される時刻は、協定世界時 (UTC) です。その時刻形式は、`YYYY-MM-DD['T'HH:MM:SS]` となっている必要があります。年月日だけを指定する場合、時間と分と秒は `00:00:00` に設定されます。

-time および *-processtime* オプションは、相互に排他的です。

-*processtime* *timestamp*
timestamp で指定した時刻より前に完了したプロセスに属す監査ログ・エントリーがすべて削除されます。*-time* パラメーターの場合と同じ時刻形式を使用します。

-time および *-processtime* オプションは、相互に排他的です。

-slice *size*
-all パラメーターとともに使用され、*size* は各トランザクションに含まれるエントリー数を指定します。最適値は、データベース・システムで使用可能なログ・サイズによって決まります。値が高いほど、必要なトランザクションは少なくなりますが、データベース・ロギング・スペースを超過する可能性があります。値が低い場合は、スクリプトでの削除の完了に時間がかかる可能性があります。slice パラメーターのデフォルトのサイズは 250 です。

注: 未使用の担当者照会のクリーンアップのスクリプトの jacl バージョン `deleteAuditLog.jacl` は、推奨されていません。これは、`ProcessChoreographer` ディレクトリーの `util` サブディレクトリーで提供され、ここで説明しているのと同じパラメーターを使用しますが、`-lang jython` オプションは省略する必要があります。

未使用のプロセス・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

この手順を始める前に、削除されるテンプレートが置かれているアプリケーション・サーバーが稼働している必要があります。つまり、サーバー接続が必要であるため、wsadmin の -conntype none オプションは使用できません。WebSphere 管理セキュリティーが使用可能になっている場合でも、このコマンドを実行するための特殊権限は不要です。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。

   Windows プラットフォームの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥admin
   ```

   Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

   ```
   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。

   無効になったビジネス・プロセス・テンプレートを Windows システムで削除するには、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

   ```
   install_root¥bin¥wsadmin -f deleteInvalidProcessTemplate.py
                            -server server_name
                            -templateName templateName
                            -validFrom validFromString
                            [-profileName profileName]

   install_root¥bin¥wsadmin -f deleteInvalidProcessTemplate.py
                            -server server_name
                            -node node_name
                            -templateName templateName
                            -validFrom validFromString
                            [-profileName profileName]
   ```

```
install_root¥bin¥wsadmin -f deleteInvalidProcessTemplate.py
                         -cluster cluster_name
                         -templateName templateName
                         -validFrom validFromString
                         [-profileName profileName]
```

無効になったビジネス・プロセス・テンプレートを Linux および UNIX システムで削除するには、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin.sh -f deleteInvalidProcessTemplate.py
                         -server server_name
                         -templateName templateName
                         -validFrom validFromString
                         [-profileName profileName]

install_root/bin/wsadmin.sh -f deleteInvalidProcessTemplate.py
                         -server server_name
                         -node node_name
                         -templateName templateName
                         -validFrom validFromString
                         [-profileName profileName]

install_root/bin/wsadmin.sh -f deleteInvalidProcessTemplate.py
                         -cluster cluster_name
                         -templateName templateName
                         -validFrom validFromString
                         [-profileName profileName]
```

無効になったビジネス・プロセス・テンプレートを i5/OS システムで削除するには、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin -f deleteInvalidProcessTemplate.py
                         -server server_name
                         -templateName templateName
                         -validFrom validFromString
                         [-profileName profileName]

install_root/bin/wsadmin -f deleteInvalidProcessTemplate.py
                         -server server_name
                         -node node_name
                         -templateName templateName
                         -validFrom validFromString
                         [-profileName profileName]

install_root/bin/wsadmin -f deleteInvalidProcessTemplate.py
                         -cluster cluster_name
                         -templateName templateName
                         -validFrom validFromString
                         [-profileName profileName]
```

各部の意味は、次のとおりです。

-cluster *cluster_name*
クラスターの名前。Business Process Choreographer が WebSphere クラスター用に構成されている場合は必須です。クラスター名、またはサーバー名およびノード名を指定できます。

-node *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*
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。

-validFrom *validFromString*
管理コンソールの表示どおりにテンプレートが有効になった日付 (UTC 形式)。ストリングの形式は、「yyyy-MM-ddThh:mm:ss」(年、月、日、T、時、分、秒) 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。

-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。

次のタスク

注: 未使用プロセスのクリーンアップのスクリプトの jacl バージョン、`deleteInvalidProcessTemplate.jacl` は推奨されません。これは、`ProcessChoreographer` ディレクトリーの `util` サブディレクトリーで提供され、ここで説明しているのと同じパラメーターを使用しますが、`-lang jython` オプションは省略する必要があります。

未使用のヒューマン・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

この手順を始める前に、削除されるテンプレートが置かれているアプリケーション・サーバーが稼働している必要があります。つまり、サーバー接続が必要であるため、`wsadmin` の `-conntype none` オプションは使用できません。WebSphere 管理セキュリティーが使用可能になっている場合でも、このコマンドを実行するための特殊権限は不要です。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。

   Windows プラットフォームの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥admin
   ```

   Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

   ```
   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。

   無効になったヒューマン・タスク・テンプレートを Windows システムで削除するには、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

   ```
   install_root¥bin¥wsadmin -f deleteInvalidTaskTemplate.py
                            -server server_name
                            -templateName templateName
                            -validFrom validFromString
                            -nameSpace nameSpace
                             [-profileName profileName]

   install_root¥bin¥wsadmin -f deleteInvalidTaskTemplate.py
                            -server server_name
                            -node node_name
                            -templateName templateName
                            -validFrom validFromString
                            -nameSpace nameSpace
                            [-profileName profileName]

   install_root¥bin¥wsadmin -f deleteInvalidTaskTemplate.py
                            -cluster cluster_name
                            -templateName templateName
                            -validFrom validFromString
                            -nameSpace nameSpace
                            [-profileName profileName]
   ```

   無効になったヒューマン・タスク・テンプレートを UNIX および Linux システムで削除するには、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

   ```
   install_root/bin/wsadmin.sh -f deleteInvalidTaskTemplate.py
                            -server server_name
                            -templateName templateName
                            -validFrom validFromString
                            -nameSpace nameSpace
                            [-profileName profileName]

   install_root/bin/wsadmin.sh -f deleteInvalidTaskTemplate.py
                            -server server_name
                            -node node_name
                            -templateName templateName
                            -validFrom validFromString
                            -nameSpace nameSpace
   ```

```
                        [-profileName profileName]

install_root/bin/wsadmin.sh -f deleteInvalidTaskTemplate.py
                        -cluster cluster_name
                        -templateName templateName
                        -validFrom validFromString
                        -nameSpace nameSpace
                        [-profileName profileName]
```

無効になったヒューマン・タスク・テンプレートを i5/OS システムで削除するには、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin -f deleteInvalidTaskTemplate.py
                        -server server_name
                        -templateName templateName
                        -validFrom validFromString
                        -nameSpace nameSpace
                        [-profileName profileName]

install_root/bin/wsadmin -f deleteInvalidTaskTemplate.py
                        -server server_name
                        -node node_name
                        -templateName templateName
                        -validFrom validFromString
                        -nameSpace nameSpace
                        [-profileName profileName]

install_root/bin/wsadmin -f deleteInvalidTaskTemplate.py
                        -cluster cluster_name
                        -templateName templateName
                        -validFrom validFromString
                        -nameSpace nameSpace
                        [-profileName profileName]
```

各部の意味は、次のとおりです。

-cluster *cluster_name*
クラスターの名前。Business Process Choreographer が WebSphere クラスター用に構成されている場合は必須です。クラスター名、またはサーバー名およびノード名を指定できます。

-node *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*
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。

-validFrom *validFromString*
管理コンソールの表示どおりにテンプレートが有効になった日付 (UTC 形式)。ストリングの形式は、「yyyy-MM-ddThh:mm:ss」(年、月、日、T、時、分、秒) 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・ネーム・スペース。

-profileName *profileName*
ユーザー定義プロファイルの名前。デフォルト・プロファイルで作業していない場合、このオプションを指定します。

次のタスク

注: 未使用の担当者照会のクリーンアップのスクリプトの jacl バージョン `deleteInvalidTaskTemplate.jacl` は、推奨されていません。これは、`ProcessChoreographer` ディレクトリーの `util` サブディレクトリーで提供され、ここで説明しているのと同じパラメーターを使用しますが、`-lang jython` オプションは省略する必要があります。

完了したプロセス・インスタンスの削除

Business Process Choreographer データベースから、終了状態 (終了、強制終了、または失敗) になった最上位のプロセス・インスタンスを選択して削除するには、管理スクリプトを使用します。

始める前に

この手順を始める前に、削除されるプロセス・インスタンスが置かれているアプリケーション・サーバーが稼働している必要があります。つまり、サーバー接続が必要であるため、`wsadmin` の `-conntype none` オプションは使用できません。WebSphere 管理セキュリティーが使用可能になっている場合でも、このコマンドを実行するための特殊権限は不要です。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

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。

   Windows プラットフォームの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥admin
   ```

   Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

   ```
   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。

Windows システムの場合は、次のコマンドを入力します。

```
install_root¥bin¥wsadmin –f deleteCompletedProcessInstances.py
  [([-node nodeName] -server server_name) | (-cluster cluster_name)]
  (-all | -finished | -terminated | -failed )
  [-templateName templateName [-validFrom timestamp]]
  [-startedBy userID ]
  [-completedBefore timestamp]
  [-profileName profileName]
```

Linux および UNIX システムの場合、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 [-validFrom timestamp]]
  [-startedBy userID ]
  [-completedBefore timestamp]
  [-profileName profileName]
```

i5/OS システムの場合は、次のコマンドを入力します。

```
install_root/bin/wsadmin –f deleteCompletedProcessInstances.py
  [([-node nodeName] -server server_name) | (-cluster cluster_name)]
  (-all | -finished | -terminated | -failed )
  [-templateName templateName [-validFrom timestamp]]
  [-startedBy userID ]
  [-completedBefore timestamp]
  [-profileName profileName]
```

各部の意味は、次のとおりです。

-node *nodeName*
サーバー名を指定する場合のオプション。この名前は、ノードを示します。デフォルトはローカル・ノードです。サーバー名およびノード名、またはクラスター名を指定できます。

-server *server_name*
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*
クラスターの名前。Business Process Choreographer が WebSphere クラスター用に構成されている場合は必須です。クラスター名、またはサーバー名およびノード名を指定できます。

-all|-finished|-terminated|-failed
状態によってどのプロセス・インスタンスを削除するかを指定します。

-templateName *templateName*
(オプション) 削除するプロセス・テンプレートの名前を指定します。 このオプションを指定すると、validFrom パラメーターも使用できます。

-validFrom *timestamp*
管理コンソールの表示どおりにテンプレートが有効になった日付 (UTC 形式)。このオプションは、templateName オプションと一緒にしか使用できません。 *timestamp* ストリングの形式は、「yyyy-MM-ddThh:mm:ss」(年、月、日、T、時、分、秒) です。例えば、2006-11-20T12:00:00 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。

-completedBefore *timestamp*
(オプション) 指定の時刻より前に完了したプロセス・インスタンスを削除します。*timestamp* ストリングの形式は、「yyyy-MM-dd[Thh:mm:ss]」(年、月、日、T、時、分、秒) です。例えば、`2006-07-20T12:00:00` 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` に設定されます。

-profileName *profileName*
ユーザー定義プロファイルの名前。デフォルト・プロファイルで作業していない場合、このオプションを指定します。

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 –f deleteCompletedProcessInstances.py
        -node myNode -server myServer
        -finished
        -startedBy Antje
```

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。

Observer データベースからのデータの削除

Business Process Choreographer Observer データベースから、指定の条件に一致したプロセス・インスタンスのデータ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。

このタスクについて

プロセス・インスタンスの監視者情報を削除するには、3 つの方法があります。

- 指定の時刻より前に終了状態 deleted になったプロセス・インスタンスの監視者データを削除するには、パラメーター `-deletedBefore` *timestamp* を指定してください。
- 現在の状態にかかわらず、特定のテンプレート・バージョンのプロセス・インスタンスの監視者データを削除するには、パラメーター `-templateName` *templateName* `-validFrom` *timestamp* 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・インスタンスの監視者データを削除するには、パラメーター `-force -templateName` *template_name* `-validFrom` *timestamp* `-state` *state* `-reachedBefore` *timestamp* を指定してください。ここで、`-templateName` *template_name* および `-validFrom` *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. 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。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。
2. 管理スクリプトが置かれている Business Process Choreographer サブディレクトリーに移動します。

   Windows プラットフォームの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥admin
   ```

   Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

   ```
   cd install_root/ProcessChoreographer/admin
   ```
3. コマンドを入力して、データベースから、特定のプロセス・インスタンスの監視者データを削除します。

   Windows プラットフォームの場合、以下のコマンドを入力します。

   ```
   install_root¥bin¥wsadmin
        -f observerDeleteProcessInstanceData.py
        ( [-node node_name] -server server_name) | (-cluster cluster_name )
        [ -profileName profile_name ]
        [ -dataSource dataSource_JNDI_name ]
        [ -dbSchemaName dbSchemaName]
        (
          -deletedBefore timestamp
          | ( -templateName template_name  -validFrom timestamp )
          | ( -force [-templateName template_name  -validFrom timestamp]
              -state state -reachedBefore timestamp )
        )
   ```

   Linux および UNIX プラットフォームの場合、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 )
        [ -profileName profile_name ]
        [ -dataSource dataSource_JNDI_name ]
        [ -dbSchemaName dbSchemaName]
        (
          -deletedBefore timestamp
          | ( -templateName template_name  -validFrom timestamp )
          | ( -force [-templateName template_name  -validFrom timestamp]
              -state state -reachedBefore timestamp )
        )
   ```

   i5/OS プラットフォームの場合、以下のコマンドを入力します。

   ```
   install_root/bin/wsadmin
        -f observerDeleteProcessInstanceData.py
        ( [-node node_name] -server server_name) | (-cluster cluster_name )
        [ -profileName profile_name ]
        [ -dataSource dataSource_JNDI_name ]
        [ -dbSchemaName dbSchemaName]
        (
          -deletedBefore timestamp
          | ( -templateName template_name  -validFrom timestamp )
          | ( -force [-templateName template_name  -validFrom timestamp]
              -state state -reachedBefore timestamp )
        )
   ```

   各部の意味は、次のとおりです。

-node *node_name*
この名前は、ノードを示します。このパラメーターはオプションです。デフォルトはローカル・ノードです。

-server *server_name*
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*
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。

-profileName *profile_name*
ユーザー定義の WebSphere 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。

-dataSource *datasource_JNDI_name*
サーバーまたはクラスターには複数の Observer データベースを所有できるため、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、Observer 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。

-deletedBefore *timestamp*
指定の時刻より前に状態が削除になったプロセス・インスタンスの監視者データを削除する。

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*]」(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` 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監視者データをすべて削除します。

-validFrom *timestamp*
これは、templateName オプションを指定する場合に必要です。

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*」です (年、月、日、T、時、分、秒)。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・インスタンスの監視者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、オプション -state および -reachedBefore も指定する必要があります。オプション -templateName および -validFrom 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。

-reachedBefore *timestamp*
指定の状態になるべき時刻を指定します。

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*]」(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` に設定されます。

例えば、プロセス・テンプレート *my_template* のインスタンスの監視者データのうち、2007 年 1 月 2 日の正午から有効になり、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 –f observerDeleteProcessInstanceData.py
        -node my_node -server my_server
        -force -templateName my_template -validFrom 2007-01-02T12:00:00
               -state running -reachedBefore 2007-07-20T12:00:00
```

結果

正常に実行されれば、ツールから、監視者データが削除されたインスタンスの数と、データベースから削除された表項目の数が報告されます。そうでない場合は、エラー情報が報告され、データベースは変更されません。

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。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設定されている場合は、オペレーター権限を持っている必要があります。

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。

   Windows プラットフォームの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥admin
   ```

   Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

   ```
   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。

   Windows システムでは、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

   ```
   install_root¥bin¥wsadmin -f queryNumberOfFailedMessages.py
                            -cluster cluster_name
                            [ -bfm | -htm ]
                            [-profileName profileName]

   install_root¥bin¥wsadmin -f queryNumberOfFailedMessages.py
                            -node nodeName
                            -server server_name
                            [ -bfm | -htm ]
                            [-profileName profileName]
   ```

   Linux および UNIX システムでは、以下のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

   ```
   install_root/bin/wsadmin.sh -f queryNumberOfFailedMessages.py
                            -cluster cluster_name
                            [ -bfm | -htm ]
                            [-profileName profileName]

   install_root/bin/wsadmin.sh -f queryNumberOfFailedMessages.py
                            -node nodeName
                            -server server_name
                            [ -bfm | -htm ]
                            [-profileName profileName]
   ```

   i5/OS システムでは、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

   ```
   install_root/bin/wsadmin -f queryNumberOfFailedMessages.py
                            -cluster cluster_name
                            [ -bfm | -htm ]
                            [-profileName profileName]

   install_root/bin/wsadmin -f queryNumberOfFailedMessages.py
                            -node nodeName
                            -server server_name
                            [ -bfm | -htm ]
                            [-profileName profileName]
   ```

   各部の意味は、次のとおりです。

   -cluster *cluster_name*
   クラスターの名前。Business Process Choreographer が WebSphere クラスター用に構成されている場合は必須です。

-node *nodeName*
サーバー名を指定する場合のオプション。この名前は、ノードを示します。デフォルトはローカル・ノードです。

-server *server_name*
サーバーの名前。クラスター名が指定されていない場合は必須です。

-bfm|-htm
これらのキーワードはオプションであり、相互に排他的です。デフォルトでは、どちらのオプションも指定されていなければ、ビジネス・プロセスとヒューマン・タスク両方の失敗したメッセージ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。

-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。

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 server_name
```

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。

Windows システムでは、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

```
install_root¥bin¥wsadmin -f replayFailedMessages.py
                         -cluster cluster_name
                         -queue replayQueue
                         [ -bfm | -htm ]
                         [-profileName profileName]

install_root¥bin¥wsadmin -f replayFailedMessages.py
                         -node nodeName
                         -server server_name
                         -queue replayQueue
                         [ -bfm | -htm ]
                         [-profileName profileName]

install_root¥bin¥wsadmin -f replayFailedMessages.py
                         -server server_name
                         -queue replayQueue
                         [ -bfm | -htm ]
                         [-profileName profileName]
```

Linux および UNIX システムでは、以下のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin.sh -f replayFailedMessages.py
                         -cluster cluster_name
                         -queue replayQueue
                         [ -bfm | -htm ]
                         [-profileName profileName]

install_root/bin/wsadmin.sh -f replayFailedMessages.py
                         -node nodeName
                         -server server_name
                         -queue replayQueue
```

```
                        [ -bfm | -htm ]
                        [-profileName profileName]

install_root/bin/wsadmin.sh -f replayFailedMessages.py
                        -server server_name
                        -queue replayQueue
                        [ -bfm | -htm ]
                        [-profileName profileName]
```

i5/OS システムでは、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin -f replayFailedMessages.py
                        -cluster cluster_name
                        -queue replayQueue
                        [ -bfm | -htm ]
                        [-profileName profileName]

install_root/bin/wsadmin -f replayFailedMessages.py
                        -node nodeName
                        -server server_name
                        -queue replayQueue
                        [ -bfm | -htm ]
                        [-profileName profileName]

install_root/bin/wsadmin -f replayFailedMessages.py
                        -server server_name
                        -queue replayQueue
                        [ -bfm | -htm ]
                        [-profileName profileName]
```

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)

-cluster *cluster_name*
クラスターの名前。Business Process Choreographer が WebSphere クラスター用に構成されている場合は必須です。

-node *nodeName*
サーバー名を指定する場合のオプション。この名前は、ノードを示します。デフォルトはローカル・ノードです。

-server *server_name*
サーバーの名前。クラスター名が指定されていない場合は必須です。

-bfm|-htm
これらのキーワードはオプションであり、相互に排他的です。デフォルトでは、どちらのオプションも指定されていなければ、ビジネス・プロセスとヒューマン・タスク両方の失敗したメッセージが再生されます。ビジネス・プロセスのメッセージのみを再生する場合は、-bfm オプションを指定します。ヒューマン・タスクのメッセージのみを再生する場合は、-htm オプションを指定します。

-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。

次のタスク

注: 未使用担当者照会のクリーンアップのスクリプトの jacl バージョン、`replayFailedMessages.jacl` は推奨されません。これは、`ProcessChoreographer` ディレクトリーの `util` サブディレクトリーで提供され、ここで説明しているのと同じパラメーターを使用しますが、`-lang jython` オプションは省略する必要があります。

関連概念

34 ページの『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。

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

この手順を始める前に、次の条件が満たされている必要があります。

- メッセージが照会または再生されるアプリケーション・サーバーが稼働している必要があります。つまり、サーバー接続が必要であるため、wsadmin の `-conntype none` オプションは使用できません。
- 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設定されている場合は、オペレーター権限を持っている必要があります。

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。

   Windows プラットフォームの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥admin
   ```

   Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

   ```
   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。

Windows プラットフォームの場合は、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

```
install_root¥bin¥wsadmin -f refreshStaffQuery.py
        -server server_name
       [-processTemplate templateName |
       (-taskTemplate templateName [-nameSpace nameSpace]) |
        -userlist username{,username}...]
       [-profileName profileName]

install_root¥bin¥wsadmin -f refreshStaffQuery.py
        -node nodeName
        -server server_name
       [-processTemplate templateName |
       (-taskTemplate templateName [-nameSpace nameSpace]) |
        -userlist username{,username}...]
       [-profileName profileName]

install_root¥bin¥wsadmin -f refreshStaffQuery.py
        -cluster cluster_name
       [-processTemplate templateName |
       (-taskTemplate templateName [-nameSpace nameSpace]) |
        -userlist username{,username}...]
       [-profileName profileName]
```

Linux および UNIX プラットフォームでは、以下のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin.sh -f refreshStaffQuery.py
        -server server_name
       [-processTemplate templateName |
       (-taskTemplate templateName [-nameSpace nameSpace]) |
        -userlist username{,username}...]
       [-profileName profileName]

install_root/bin/wsadmin.sh -f refreshStaffQuery.py
        -node nodeName
        -server server_name
       [-processTemplate templateName |
       (-taskTemplate templateName [-nameSpace nameSpace]) |
        -userlist username{,username}...]
       [-profileName profileName]

install_root/bin/wsadmin.sh -f refreshStaffQuery.py
        -cluster cluster_name
       [-processTemplate templateName |
       (-taskTemplate templateName [-nameSpace nameSpace]) |
        -userlist username{,username}...]
       [-profileName profileName]
```

i5/OS プラットフォームでは、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin -f refreshStaffQuery.py
        -server server_name
       [-processTemplate templateName |
       (-taskTemplate templateName [-nameSpace nameSpace]) |
        -userlist username{,username}...]
       [-profileName profileName]

install_root/bin/wsadmin -f refreshStaffQuery.py
        -node nodeName
        -server server_name
       [-processTemplate templateName |
```

```
       (-taskTemplate templateName [-nameSpace nameSpace]) |
        -userlist username{,username}...]
       [-profileName profileName]

install_root/bin/wsadmin -f refreshStaffQuery.py
        -cluster cluster_name
       [-processTemplate templateName |
       (-taskTemplate templateName [-nameSpace nameSpace]) |
        -userlist username{,username}...]
       [-profileName profileName]
```

各部の意味は、次のとおりです。

-cluster *cluster_name*
クラスターの名前。Business Process Choreographer が WebSphere クラスター用に構成されている場合は必須です。

-node *nodeName*
サーバー名を指定する場合のオプション。この名前は、ノードを示します。デフォルトはローカル・ノードです。

-server *server_name*
サーバーの名前。クラスター名が指定されていない場合は必須です。

-processTemplate *templateName*
プロセス・テンプレートの名前。このプロセス・テンプレートに属す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。

-nameSpace *nameSpace*
タスク・テンプレートのネーム・スペース。

-userlist *userName*
コンマで区切られたユーザー名のリスト。指定された名前を含む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。

-profileName *profileName*
ユーザー定義プロファイルの名前。デフォルト・プロファイルで作業していない場合、このオプションを指定します。

注: *templateName* も *userlist* も指定しない場合は、データベースに保管されているすべての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。

注: 担当者照会の最新表示スクリプトの jacl バージョン `refreshStaffQuery.jacl` は、推奨されていません。これは、`ProcessChoreographer` ディレクトリーの `util` サブディレクトリーで提供され、ここで説明しているのと同じパラメーターを使用しますが、`-lang jython` オプションは省略する必要があります。

関連概念

102 ページの『担当者割り当て基準および担当者解決結果の管理』
タスク許可のロールに関連付けられている担当者割り当て基準は、デプロイ済みのタスク・テンプレートまたはタスク・インスタンスの存続期間中は常に有効です。

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

この手順を始める前に、次の条件が満たされている必要があります。

- 未使用の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。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設定されている場合は、オペレーター権限を持っている必要があります。

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。

   Windows プラットフォームの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥admin
   ```

   Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. 未使用の担当者リストを除去します。

   Windows プラットフォームの場合は、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

   ```
   install_root¥bin¥wsadmin -f cleanupUnusedStaffQueryInstances.py
                            -server server_name
                            [-profileName profileName]
   ```

```
install_root¥bin¥wsadmin -f cleanupUnusedStaffQueryInstances.py
                         -node nodeName
                         -server server_name
                         [-profileName profileName]

install_root¥bin¥wsadmin -f cleanupUnusedStaffQueryInstances.py
                         -cluster cluster_name
                         [-profileName profileName]
```

Linux および UNIX プラットフォームでは、以下のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin.sh -f cleanupUnusedStaffQueryInstances.py
                         -server server_name
                         [-profileName profileName]

install_root/bin/wsadmin.sh -f cleanupUnusedStaffQueryInstances.py
                         -node nodeName
                         -server server_name
                         [-profileName profileName]

install_root/bin/wsadmin.sh -f cleanupUnusedStaffQueryInstances.py
                         -cluster cluster_name
                         [-profileName profileName]
```

i5/OS プラットフォームでは、次のコマンドのいずれかを入力します。コマンド間の相違を強調して表示しています。

```
install_root/bin/wsadmin -f cleanupUnusedStaffQueryInstances.py
                         -server server_name
                         [-profileName profileName]

install_root/bin/wsadmin -f cleanupUnusedStaffQueryInstances.py
                         -node nodeName
                         -server server_name
                         [-profileName profileName]

install_root/bin/wsadmin -f cleanupUnusedStaffQueryInstances.py
                         -cluster cluster_name
                         [-profileName profileName]
```

各部の意味は、次のとおりです。

-cluster *cluster_name*
クラスターの名前。Business Process Choreographer が WebSphere クラスター用に構成されている場合は必須です。

-node *nodeName*
サーバー名を指定する場合のオプション。この名前は、ノードを示します。デフォルトはローカル・ノードです。

-server *server_name*
サーバーの名前。クラスター名が指定されていない場合は必須です。

-profileName *profileName*
ユーザー定義プロファイルの名前。デフォルト・プロファイルで作業していない場合、このオプションを指定します。

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。

次のタスク

注: 未使用の担当者照会をクリーンアップするスクリプトの jacl バージョン `cleanupUnusedStaffQueryInstances.jacl` は、推奨されません。これは、`ProcessChoreographer` ディレクトリーの `util` サブディレクトリーで提供され、ここで説明しているのと同じパラメーターを使用しますが、`-lang jython` オプションは省略する必要があります。

第 7 章 Business Process Choreographer Explorer の概要

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・タスクを管理したり、割り当てられたタスクを処理したりできます。

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。

- ビジネス管理者なら、ビジネス・プロセスのライフ・サイクルを管理したり、ビジネス・プロセスを修復したり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または代理人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。
- 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。

関連タスク

398 ページの『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。

403 ページの『作業割り当ての管理』
タスクの開始後に、そのタスクに対する作業割り当ての管理が必要になる場合があります (例えば、ワークグループのメンバー間でワークロードを効率よく分散するため)。

400 ページの『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。

401 ページの『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。

Business Process Choreographer Explorer ユーザー・インターフェース

Business Process Choreographer Explorer とは、ビジネス・プロセスおよびヒューマン・タスクを管理するための管理機能のセットを提供するスタンドアロン Web アプリケーションです。このインターフェースは、タスクバー、ナビゲーション・ペイン、およびワークスペースで構成されています。

以下の図は、Business Process Choreographer Explorer ユーザー・インターフェースのレイアウトを示しています。

ユーザー・インターフェースには以下のような主要な領域があります。

タスクバー

すべてのユーザーは、タスクバーを使用して、Business Process Choreographer Explorer からログアウトしたり、不在設定を指定したり、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。システム管理者権限を持っている場合、タスクバーには以下のオプションも含まれます。

カスタマイズ
このオプションは、Business Process Choreographer Explorer のこのインスタンスのナビゲーション・ペインに、ビューを追加したり除去したりするために選択します。また、ユーザーがログインしたときに表示されるビューを定義することもできます。

ビューの定義
このオプションは、ユーザー・グループのカスタマイズ・ビューを定義するために選択します。

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。

ナビゲーション・ペイン

ナビゲーション・ペインには、オブジェクト (例えば、開始済みのプロセス・インスタンス、または管理する許可が与えられているヒューマン・タスク) の管理に使用するビューへのリンクが含まれています。デフォルト・ユーザー・インターフェースには、ビジネス・プロセスおよびタスク用の事前定義ビューへのリンクが含まれています。

システム管理者は、ナビゲーション・ペインで事前定義ビューの追加や除去を行い、ナビゲーション・ペインに追加するためのカスタム・ビューを定義することにより、デフォルトのナビゲーション・ペインの内容をカスタマイズできます。すべてのユーザーは、ナビゲーション・ペインから個別設定ビューを定義できます。

ワークスペース

ワークスペースには、ビジネス・プロセスおよびヒューマン・タスク関連オブジェクトを表示および管理するために使用するページが含まれています。ナビゲーション・ペインのリンクをクリックするか、アクション・バーのアクションをクリックするか、またはワークスペース・ページ自体の内部のリンクをクリックすることによって、これらのページにアクセスします。

Business Process Choreographer Explorer ナビゲーション・ペイン

ナビゲーション・ペインは、ビジネス・プロセスおよびヒューマン・タスク・オブジェクト (プロセス・インスタンスおよび作業割り当てなど) を管理するために使用するビューへのアクセスに使用します。デフォルト・ユーザー・インターフェースには、ビジネス・プロセスおよびタスク用の事前定義ビューへのリンクが含まれています。また、独自の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。

- ビューへのナビゲート。

  ビュー名をクリックして、そのビューにナビゲートします。
- グループの縮小および展開。

ナビゲーション・ペインの項目の横にある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。

- 新規検索を定義します。

  「**新規検索**」アイコン（  ）をクリックして、オブジェクトを検索するか、個別設定ビューを定義します。

ビュー型によっては、ポップアップ・メニューから追加アクションを使用できます。アイコンによって、ポップアップ・メニューが使用可能であるかどうか示されます。

- ビューを削除するには、「**削除**」アイコン（  ）をクリックします。
- ビューを変更するには、「**編集**」アイコン（  ）をクリックします。
- ビューのコピーを作成して、そのコピーを変更するには、「**コピー**」アイコン（  ）をクリックします。
- ビューをリスト内で上下に移動させるには、「**上へ**」アイコン（  ）または「**下へ**」アイコン（  ）をクリックします。

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。ビューのグループに対してビューが定義されていない場合、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。

ユーザーのプロセス・テンプレート

このビューには、プロセス・テンプレートのリストが表示されます。このビューから、プロセス・テンプレートとその構造に関する情報を表示したり、テンプレートに関連したプロセス・インスタンスのリストを表示したり、プロセス・インスタンスを開始したりすることができます。

プロセス・インスタンス

プロセス・インスタンス・グループには、以下のビューが含まれます。

ユーザーが開始

このビューには、開始したプロセス・インスタンスが表示されます。このビューから、プロセス・インスタンスの進行をモニターし、それに関連したアクティビティー、プロセス、またはタスクをリストできます。

ユーザーが管理

このビューには、管理する許可が与えられているプロセス・インス

タンスが表示されます。このビューから、プロセス・インスタンスを操作することができます。例えば、プロセスの中断と再開、プロセス・インスタンス内のアクティビティーの進行のモニターを行えます。

重大なプロセス
このビューには、停止状態のアクティビティーを含む、実行状態のプロセス・インスタンスが表示されます。このビューから、プロセス・インスタンスを操作したり、アクティビティーをリストしてからそれらを操作することができます。

終了したプロセス
このビューには、終了状態のプロセス・インスタンスが表示されます。このビューから、それらのプロセス・インスタンスを操作できます。

失敗した補正
このビューには、microflow についての失敗した補正アクションが表示されます。

アクティビティー・インスタンス
デフォルトでは、アクティビティー・インスタンス・グループにビューは含まれません。そのため、このグループはデフォルト・ナビゲーション・ペインに表示されません。

タスク・テンプレート
タスク・テンプレート・グループには、以下のビューが含まれます。

ユーザーのタスク・テンプレート
このビューには、タスク・テンプレートのリストが表示されます。このビューから、タスク・インスタンスの作成と開始を行い、テンプレートに関連したタスク・インスタンスのリストを表示できます。

タスク・インスタンス
タスク・インスタンス・グループには、以下のビューが含まれます。

ユーザーの予定
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。

すべてのタスク
このビューには、ユーザーが所有者、潜在的所有者、または編集者となっているタスクがすべて表示されます。このビューから、タスク・インスタンスを処理したり、要求したタスク・インスタンスを解放したり、タスク・インスタンスを別のユーザーに転送することができます。

ユーザーが開始
このビューには、開始したタスク・インスタンスが表示されます。このビューから、タスク・インスタンスを処理したり、要求したタ

スク・インスタンスを解放したり、タスク・インスタンスを別のユーザーに転送することができます。

ユーザーが管理
このビューには、管理する許可が与えられているタスク・インスタンスが表示されます。このビューから、タスク・インスタンスを処理することができます。例えば、プロセスの中断と再開、タスク・インスタンスの作業項目の作成、タスク・インスタンスの現行作業項目のリストの表示を行えます。

ユーザーのエスカレーション
このビューには、ログオンしたユーザーのすべてのエスカレーションが表示されます。

ビュー型

ナビゲーション・ペインには、以下のタイプのビューが含まれます。ビューによっては、ポップアップ・メニューから追加アクションを使用可能です。

デフォルトのナビゲーション・ペインの事前定義ビュー。
これらのビュー・グループは、ナビゲーション・ペインが、「ナビゲーション・ツリーおよびログイン・ビューのカスタマイズ」ページにおいてシステム管理者によって変更されていない場合にのみ使用可能です。これらのビューでは、ポップアップ・メニューは使用不可です。

システム管理者によってナビゲーション・ペインに追加されたカスタマイズ・ビューと事前定義ビュー。
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。

- 事前定義ビューは「**事前定義ビュー (Predefined view)**」アイコン で示されます。システム管理者はポップアップ・メニューを使用して、ナビゲーション・ペインのビューの位置を変更できます。
- カスタマイズ・ビューは「**カスタム・ビュー (Custom view)**」アイコン で示されます。システム管理者は、これらのビューを削除、編集、コピー、および移動することができます。

個別設定ビュー。
これらのビューは「**カスタム・ビュー (Custom view)**」アイコン で示されます。これらのビューは、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。

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。

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。

   `http://app_server_host:port_no/context_root`

   各部の意味は、次のとおりです。

   app_server_host
   使用するビジネス・プロセス・アプリケーションを提供するアプリケーション・サーバーのホストのネットワーク名。

   port_no
   Business Process Choreographer Explorer が使用するポート番号。ポート番号はシステム構成によって異なります。デフォルト・ポート番号は `9080` です。

   context_root
   Business Process Choreographer Explorer アプリケーションの、アプリケーション・サーバー上のルート・ディレクトリー。デフォルトは `bpc` です。
2. セキュリティーが有効になっている場合は、ユーザー ID とパスワードを入力して、「**ログイン**」をクリックする必要があります。

結果

Business Process Choreographer Explorer の最初のページが表示されます。デフォルトでは、このページには「ユーザーの予定」ビューが表示されます。

Business Process Choreographer Explorer のカスタマイズ

Business Process Choreographer Explorer では、管理者がビジネス・プロセスやヒューマン・タスクを管理するため、およびビジネス・ユーザーが割り当てられたタスクを処理するためのユーザー・インターフェースが用意されています。これは汎用インターフェースなので、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。

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。Business Process Choreographer システム管理者ロールのいずれかを持っていると、ナビゲーション・ペインに表示されるリンク、ユーザーのログイン後に表示されるビュー、およびビューに表示される情報をカスタマイズできます。

始める前に

インターフェースをカスタマイズするには、BPCSystemAdministrator 権限が必要です。

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、ビジネス状態マシンを操作するためのビューが組み込まれていません。事前定義ビューを追加すると、ビジネス状態マシン用のプロセス・テンプレートとプロセス・インスタンスを操作できるようになります。

あるいは、顧客オーダーを担当するユーザーに、顧客サービス照会を担当するユーザーとは別のインターフェースを提供することもできます。Business Process Choreographer Explorer のインスタンスは、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。

Business Process Choreographer Explorer のデフォルトのユーザー・インターフェースをカスタマイズするには、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。

      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. ビューをカスタマイズします。

この Business Process Choreographer Explorer インスタンスのビューに表示される情報は、指定できます。

a. タスクバーの「**ビューの定義**」をクリックします。
b. 「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. ビューの「カスタマイズ・ビューの検索および定義」ページで、検索基準を指定します。
d. 「プロパティーの表示」タブを使用して、ビューに含めるプロパティーを選択し、リスト・プロパティーを指定します。

   これがタスク・インスタンス・ビューまたはプロセス・インスタンス・ビューである場合は、「**ビューの設定**」をクリックし、ビューのアクション・バーに追加する一連のアクションを選択して、さらにシステム管理者およびシステム・モニターのビューに組み込まれる項目を指定します。

   - 以下のようにして、ビュー型を選択します。
     - 管理アクションをビューのアクション・バーに追加するには、「**インスタンスの管理**」を選択します。
     - 一連のアクションをアクション・バーに追加してログオン・ユーザーがインスタンスを操作できるようにするには、「**インスタンスの処理**」を選択します。
   - 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。

e. 「**ビュー名**」フィールドにビューの表示名を入力し、「**保管**」をクリックします。

新しいビューがナビゲーション・ペインに表示されます。ユーザーが次に Business Process Choreographer Explorer にログインすると、新しいビューが表示されます。

ビジネス状態マシンのプロセス・テンプレート用ビューの定義

ビジネス状態マシンのプロセス・テンプレートには、事前定義ビューが用意されていますが、このタイプのテンプレート用に独自のビューを定義することもできます。

始める前に

カスタマイズ・ビューを作成するには、BPCSystemAdministrator 権限が必要です。

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、「**保管**」をクリックします。

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。

ビジネス状態マシンのプロセス・インスタンス用ビューの定義

ビジネス状態マシンのプロセス・インスタンスには、事前定義ビューが用意されていますが、このタイプのプロセス・インスタンス用に独自のビューを定義することもできます。

始める前に

カスタマイズ・ビューを作成するには、BPCSystemAdministrator 権限が必要です。

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. 「**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、「**ネーム・スペース**」フィールドに tns を入力します。ここで、tns は、*-process* のサフィックスが付けられたビジネス状態マシンのターゲット・ネーム・スペースです。また、「**表示名**」フィールドに列の表示名を指定し、「**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、「**ネーム・スペース**」フィールドに該当する情報を入力します。これらの値は、ビジネス状態マシンの定義から派生します。また、「**表示名**」フィールドで列の表示名を指定します。

  プロパティー名
  ビジネス状態マシンに定義した相関プロパティーの名前です。

  変数名 相関セットが着信パラメーターによって初期化される場合、変数名は以下の形式になります。

  *operation_name*_Input_*operation_parameter_name*

  ここで、*operation_name* は、初期状態からの移行操作の名前です。

  相関セットが発信パラメーターによって初期化される場合、変数名は以下の形式になります。

  *operation_name*_Output_*operation_parameter_name*

  ネーム・スペース
  照会プロパティーのネーム・スペース。ここで、tns は、*-process* のサフィックスが付けられたビジネス状態マシンのターゲット・ネーム・スペースです。

b. 他のカスタム・プロパティーまたは照会プロパティーを追加するか、選択した列のリストに列を追加するか、リストから列を除去します。

5. 「**ビュー名**」フィールドに照会の名前を入力し、「**保管**」をクリックします。

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、例えば特定のタスクまたはプロセスをモニターするために、独自のビューを追加することができます。

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. ナビゲーション・ツリーの新規ビューを定義するセクションで、「**新規検索**」アイコン () をクリックします。
2. ビューの「個別設定ビューの検索および定義 (Search and Define Personalized Views)」ページで、検索基準を指定します。
3. 「プロパティーの表示」タブを使用して、ビューに含めるプロパティーを選択し、リスト・プロパティーを指定します。

   これがタスク・インスタンス・ビューまたはプロセス・インスタンス・ビューである場合は、「**ビューの設定**」をクリックし、ビューのアクション・バーに追加する一連のアクションを選択します。システム管理者およびシステム・モニターであれば、ビューに含める項目を指定します。

   - 以下のようにして、ビュー型を選択します。
     - 管理アクションをビューのアクション・バーに追加するには、「**インスタンスの管理**」を選択します。
     - 一連のアクションをアクション・バーに追加してログオン・ユーザーがインスタンスを操作できるようにするには、「**インスタンスの処理**」を選択します。
   - 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。
4. 「**ビュー名**」フィールドにビューの表示名を入力し、「**保管**」をクリックします。

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

362 ページの『Business Process Choreographer Explorer ユーザー・インターフェース』

Business Process Choreographer Explorer とは、ビジネス・プロセスおよびヒューマン・タスクを管理するための管理機能のセットを提供するスタンドアロン Web アプリケーションです。このインターフェースは、タスクバー、ナビゲーション・ペイン、およびワークスペースで構成されています。

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』
- 374 ページの『フッター』
- 374 ページの『メニュー・バー』
- 374 ページの『ログイン・ページ』
- 374 ページの『ナビゲーター』
- 375 ページの『コンテンツ・パネル』
- 375 ページの『コマンド・バー』
- 375 ページの『リスト』
- 375 ページの『詳細パネル』
- 376 ページの『メッセージ・データ』
- 376 ページの『タブ付きペイン』
- 376 ページの『検索ページ』
- 377 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。

スタイル名	説明
.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。
.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。

スタイル名	説明
.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。
.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 Business Process Choreographer Observer の概要

ビジネス・プロセスおよびタスクが実行されている間、WebSphere® Process Server は、プロセス・インスタンスとそれに関連したアクティビティーの状態変更に関する情報を含むイベントを発行できます。 Business Process Choreographer Observer 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します。

このタスクについて

独自のレポートを定義するか、またはドリルダウン方法を使用して特定のプロセス・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。

Business Process Choreographer Observer は、IT レベルのモニターとビジネス・レベルのモニターの間の違いを埋めます。 Business Flow Manager コンポーネント内のイベントについて報告する手段を提供することにより、Business Process Choreographer で生じている事柄を理解することができます。

関連概念

380 ページの『Business Process Choreographer Observer ユーザー・インターフェース』
Business Process Choreographer Observer とは、プロセスおよびアクティビティー・イベントについてレポートするための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。

381 ページの『Business Process Choreographer Observer ナビゲーション・ペイン』
ナビゲーション・ペインは、作成するレポートの種類 (プロセス・レポートまたはアクティビティー・レポートなど) を選択するために使用します。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。

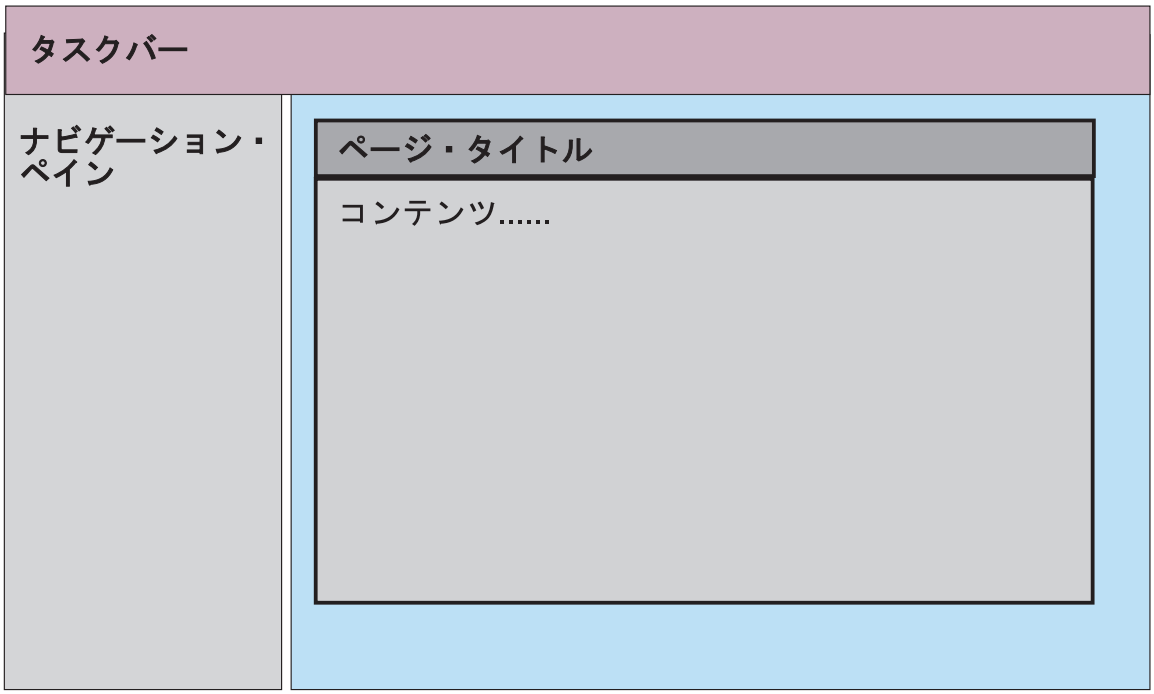
関連タスク

413 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』
ビジネス・プロセスおよびアクティビティーの処理中には、プロセス、アクティビティー、タスクのいずれかの状態が変更されると、イベントが生成できます。これらのイベントを保管し、Business Process Choreographer Observer によるレポート作成に利用できます。例えば、プロセス・ボトルネックの分析レポートや、アクティビティーから呼び出されるサービスの信頼性評価のレポートなどです。

Business Process Choreographer Observer ユーザー・インターフェース

Business Process Choreographer Observer とは、プロセスおよびアクティビティー・イベントについてレポートするための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 Observer 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

タスクバーを使用して、Business Process Choreographer Observer からログアウトしたり、一般の「ヘルプ」ページへのリンクを使用したりできます。

ナビゲーション・ペイン

ユーザー・インターフェースの左側のナビゲーション・ペインには、作成するレポートの種類 (例えば、グラフ内のアクティビティー・インスタンスのデータを表示できる) を選択するために使用するリンクが含まれています。

ワークスペース

ユーザー・インターフェースの右側のワークスペースには、レポート定義を指定し、レポートを表示するために使用するページが含まれています。これらのページにアクセスするには、ナビゲーション・ペインのリンクをクリックします。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。

関連タスク

379 ページの『第 8 章 Business Process Choreographer Observer の概要』
ビジネス・プロセスおよびタスクが実行されている間、WebSphere® Process Server は、プロセス・インスタンスとそれに関連したアクティビティーの状態変更に関する情報を含むイベントを発行できます。 Business Process Choreographer Observer 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します。

Business Process Choreographer Observer ナビゲーション・ペイン

ナビゲーション・ペインは、作成するレポートの種類 (プロセス・レポートまたはアクティビティー・レポートなど) を選択するために使用します。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。

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正符号 (+) をクリックして拡張するか、負符号 (-) をクリックしてその項目を縮小表示します。
- 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。

  「**新規レポート**」アイコン ( ) をクリックして、レポートのタイプ、レポートの内容、およびレポートのフィルター基準を指定します。
- 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。

  「**ポップアップ・メニューの表示 (Show pop-up menu)**」アイコン ( ) をクリックし、保管されたレポート定義で作業します。

  – 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。
– 検索を非同期に実行するには、「**非同期検索 (Asynchronous Search)**」アイコン（  ）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。

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。

重要: 従業員の作業効率に関するデータを収集することがプライバシーおよびデータ保護法に違反する国々では、個々のユーザーのアクティビティー・イベントを発行しないようにプロセス・モデルを定義する必要があります。

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。

関連タスク

379 ページの『第 8 章 Business Process Choreographer Observer の概要』
ビジネス・プロセスおよびタスクが実行されている間、WebSphere® Process Server は、プロセス・インスタンスとそれに関連したアクティビティーの状態変更に関する情報を含むイベントを発行できます。 Business Process Choreographer Observer 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します。

第 9 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・テンプレートおよびタスク・テンプレートを管理し、Business Process Choreographer Explorer を使用してプロセス・インスタンスおよびタスク・インスタンスを操作できます。 Business Process Choreographer Observer を使用して、ビジネス・プロセスおよびヒューマン・タスクについて報告します。

関連概念

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

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。

関連概念

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

386 ページの『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。

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?』
- 『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?』
- 387 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?』
- 387 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。

関連タスク

385 ページの『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。

388 ページの『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。

389 ページの『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・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、エンタープライズ・アプリケーション内ですべてのプロセス・テンプレートを停止します。

390 ページの『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。

390 ページの『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。

391 ページの『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。

392 ページの『プロセス・インスタンスの中断と再開』
長期間にわたって実行するトップレベルのプロセス・インスタンスを中断することができます。これは、例えば、プロセスの後半で使用するバックエンド・システムへのアクセスの構成、あるいはプロセス・インスタンスの失敗の原因となっている問題の修正を行えるよう実行することができます。プロセスの前提条件が満たされている場合は、プロセス・インスタンスの実行を再開できます。

393 ページの『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。

394 ページの『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。

395 ページの『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。

396 ページの『アクティビティーの再開』
アクティビティーを修復した場合は、新しい入力データでアクティビティーを再開することができます。

396 ページの『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。

397 ページの『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。

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

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。アプリケーションがインストールされているサーバーが稼働している必要があります。

このタスクについて

例えば、プロセス・テンプレートが属しているビジネス・プロセス・アプリケーションをアンインストールするには、まずそのビジネス・テンプレートを停止する必要があります。以下のステップでは、管理コンソールを使用して、プロセス・テンプレートを停止および開始する方法について説明します。

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 EJB モジュールの「構成」ページで、「**ビジネス・プロセス**」をクリックしてからプロセス・テンプレートをクリックします。
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。

関連概念

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

386 ページの『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。

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・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、エンタープライズ・アプリケーション内ですべてのプロセス・テンプレートを停止します。

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

例えば、プロセス・テンプレートが属しているビジネス・プロセス・アプリケーションをアンインストールするには、まずそのビジネス・テンプレートを停止する必要があります。以下のステップでは、管理スクリプトを使用して、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. 管理スクリプトを含む Business Process Choreographer サブディレクトリーに変更します。

   Windows システムの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥admin
   ```

   UNIX、Linux、および i5/OS システムの場合は、次のように入力します。

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. プロセス・テンプレートを停止します。

   Windows システムの場合は、次のように入力します。

   ```
   install_root¥bin¥wsadmin -f bpcTemplates.jacl
                                -stop application_name
   ```

   UNIX、Linux、および i5/OS システムの場合は、次のように入力します。

```
install_root/bin/wsadmin -f bpcTemplates.jacl
                              -stop application_name
```

ここで *application_name* は、そのテンプレートが所属するアプリケーションの名前です。

プロセス・テンプレートの既存のインスタンスは、正常に終了するまで稼働を続けます。アプリケーションが停止している場合、停止したテンプレートからプロセス・インスタンスを作成することはできません。

3. プロセス・テンプレートを開始します。

   Windows システムの場合は、次のように入力します。

```
install_root¥bin¥wsadmin -f bpcTemplates.jacl
                              -start application_name
```

   UNIX、Linux、および i5/OS システムの場合は、次のように入力します。

```
install_root/bin/wsadmin -f bpcTemplates.jacl
                              -start application_name
```

   プロセス・テンプレートが開始されます。Business Process Choreographer Explorer を使用して、プロセス・テンプレートからプロセス・インスタンスを開始することができます。

関連概念

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

386 ページの『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・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。

関連概念

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

386 ページの『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。

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

インストールされたプロセス・テンプレートは、すべて Business Process Choreographer Explorer でプロセス・テンプレートのリストに表示されます。新規プロセス・インスタンスを開始するには、次のステップを実行します。

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。

   ナビゲーション・ペインの「プロセス・テンプレート」の下で、「**自分のプロセス・テンプレート**」をクリックします。
2. プロセス・テンプレートの横にあるチェック・ボックスを選択し、「**インスタンスの開始**」をクリックします。

   このアクションにより、「プロセス入力メッセージ」ページが表示されます。

   プロセスに複数の操作が存在する場合、このアクションによって、すべての使用可能な操作を含むページが表示されます。プロセス・インスタンスを開始するための操作を選択します。
3. プロセス・インスタンスを開始するための入力データを提供します。

   プロセスが長期実行プロセスである場合は、プロセス・インスタンス名に入力できます。名前を指定しない場合、新規プロセス・インスタンスには、システムによって生成された名前が割り当てられます。

   プロセス入力メッセージに対する入力を完了します。
4. プロセスを開始するには、「**実行依頼**」をクリックします。

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、そのプロセスが終了すると即時にプロセス出力メッセージが表示されます。長期間にわたって実行するプロセスの場合、プロセス・インスタンスがプロセスの完了後に自動的に削除されないようにしてください。その場合、プロセス出力メッセージがプロセス・インスタンス・ビューに表示されます。すべてのプロセスに出力メッセージがあるわけではありません。例えば、プロセスが片方向操作をインプリメントしている場合、出力メッセージは表示されません。

関連概念

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

386 ページの『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・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、ナビゲーション・ペインの「プロセス・インスタンス」の下で、「**自分で管理**」をクリックします。
2. プロセス・インスタンスの横にあるチェック・ボックスを選択し、「**プロセス状態の表示**」をクリックします。

   「プロセス状態」ページが表示されます。このページには、アクティビティー、遷移を含むリンクとそのリンクの結合条件、障害ハンドラー、補正ハンドラー、およびプロセスに定義されるイベント・ハンドラーが表示されます。太字で表示されるアクティビティーは、プロセス・モデル内でビジネス関連として定義されています。これらのアクティビティーに対しては、状態情報が表示されます。
3. アクティビティーを操作するには、そのアクティビティーをクリックします。

   「アクティビティー」ページが表示されます。ここで、プロセスが最後まで実行できるようにアクションを起こすことができます。

関連概念

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

386 ページの『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・インスタンスの中断と再開

長期間にわたって実行するトップレベルのプロセス・インスタンスを中断することができます。これは、例えば、プロセスの後半で使用するバックエンド・システムへのアクセスの構成、あるいはプロセス・インスタンスの失敗の原因となっている問題の修正を行えるよう実行することができます。プロセスの前提条件が満たされている場合は、プロセス・インスタンスの実行を再開できます。

始める前に

プロセス・インスタンスを中断して再開するには、プロセス管理者権限が必要です。

プロセス・インスタンスを中断するには、プロセス・インスタンスが実行中または失敗している状態である必要があります。プロセスを再開するには、プロセス・インスタンスが中断された状態である必要があります。

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、ナビゲーション・ペインの「プロセス・インスタンス」の下で、「**自分で管理**」をクリックします。
2. プロセスを中断します。

   プロセス・インスタンスの横にあるチェック・ボックスを選択し、「**中断**」をクリックします。
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、指定されたトップレベルのプロセス・インスタンスが中断されます。プロセス・インスタンスは中断状態になります。autonomy 属性が `child` に設定されたサブプロセスも、状態が実行中、失敗、終了、または補正になっていれば中断されます。ただし、その場合も、プロセス・インスタンスに属するアクティブなアクティビティーおよびタスクは完了できます。

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

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

386 ページの『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・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、プロセス管理者権限が必要です。

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。

   ナビゲーション・ペインの「プロセス・インスタンス」の下で、「**自分で管理**」をクリックします。
2. 停止するプロセス・インスタンスの横にあるチェック・ボックスを選択します。
   - 補正を持つプロセス・インスタンスを終了させるには、「**補正**」をクリックします。

     このアクションで、プロセス・インスタンスは終了し、補正処理が開始します。
   - 補正を持たないプロセス・インスタンスを終了させるには、「**終了**」をクリックします。

     このアクションにより、プロセス・インスタンスは未解決のアクティビティーまたはタスクを待たず、即時に停止します。終了したプロセス・インスタンスは補正されません。

関連概念

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

386 ページの『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・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、プロセス管理者権限が必要です。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。

   ナビゲーション・ペインの「プロセス・インスタンス」の下で、「**自分で管理**」をクリックします。
2. 削除するプロセス・インスタンスを選択し、「**削除**」をクリックします。

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

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

386 ページの『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。

- 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。

関連概念

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

386 ページの『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。

アクティビティーの再開

アクティビティーを修復した場合は、新しい入力データでアクティビティーを再開することができます。

始める前に

アクティビティーは停止状態、関連付けられたプロセス・インスタンスは実行状態である必要があります。

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. 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. アクティビティーの再開に必要な入力データを指定します。

   アクティビティーの再開時にエラーが発生してもプロセスを続行する場合は、「**エラーの継続**」を選択します。
3. 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。

関連概念

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

386 ページの『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。

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。

始める前に

一般的に、アクティビティーは停止状態である必要があります。ただし、アクティビティーがスタッフ・アクティビティーである場合、アクティビティーは作動可能状態または要求済み状態のどちらかになることもあります。関連付けられたプロセス・インスタンスは実行状態である必要があります。

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. アクティビティーの「アクティビティー」ページに移動して、「**強制完了**」をクリックします。
2. アクティビティーの完了に必要なデータを指定します。
3. もう一度「**強制完了**」をクリックします。

関連概念

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

386 ページの『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。

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。

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。

   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。

関連概念

3 ページの『第 1 章 ビジネス・プロセスについて』
ビジネス・プロセスは、ビジネス・ゴールを達成するために呼び出される、ビジネス関連の一連のアクティビティーです。

386 ページの『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。

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。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

関連タスク

361 ページの『第 7 章 Business Process Choreographer Explorer の概要』
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・タスクを管理したり、割り当てられたタスクを処理したりできます。

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 EJB モジュールの「構成」ページで、「**ヒューマン・タスク**」をクリックしてからプロセス・テンプレートをクリックします。
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。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

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・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、エンタープライズ・アプリケーション内ですべてのタスク・テンプレートを停止します。

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、オペレーター権限を持つユーザー ID でログインしていることを確認します。

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. 管理スクリプトを含む Business Process Choreographer サブディレクトリーに変更します。

   Windows システムの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥admin
   ```

   UNIX、Linux、および i5/OS システムの場合は、次のように入力します。

   ```
   cd install_root/ProcessChoreographer/admin
   ```
2. タスク・テンプレートを停止します。

   Windows システムの場合は、次のように入力します。

   ```
   install_root¥bin¥wsadmin -f bpcTemplates.jacl
                            -stop application_name
   ```

UNIX、Linux、および i5/OS システムの場合は、次のように入力します。

```
install_root/bin/wsadmin -f bpcTemplates.jacl
                         -stop application_name
```

ここで *application_name* は、そのテンプレートが所属するアプリケーションの名前です。 タスク・テンプレートの既存のインスタンスは、正常に終了するまで稼働を続けます。

3. タスク・テンプレートを開始します。

   Windows システムの場合は、次のように入力します。

```
install_root¥bin¥wsadmin -f bpcTemplates.jacl
                         -start application_name
```

   UNIX、Linux、および i5/OS システムの場合は、次のように入力します。

```
install_root/bin/wsadmin -f bpcTemplates.jacl
                         -start application_name
```

   タスク・テンプレートが開始されます。Business Process Choreographer Explorer を使用して、タスク・テンプレートに関連付けられているタスク・インスタンスを処理できます。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

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

インストールされたタスク・テンプレートは、すべて Business Process Choreographer Explorer でタスク・テンプレートのリストに表示されます。タスク・テンプレートからタスク・インスタンスを作成して開始するには、次のステップを実行します。

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。

   ナビゲーション・ペインの「タスク・テンプレート」の下で、**「自分のタスク・テンプレート」**をクリックします。
2. タスク・テンプレートの横にあるチェック・ボックスを選択し、**「インスタンスの開始」**をクリックします。

   このアクションにより、「タスク入力メッセージ」ページが表示されます。
3. タスク・インスタンスを開始するための入力データを提供します。
4. タスク・インスタンスを開始するには、**「実行依頼」**をクリックします。

結果

タスク・インスタンスで作業する準備が整いました。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

関連タスク

361 ページの『第 7 章 Business Process Choreographer Explorer の概要』
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・タスクを管理したり、割り当てられたタスクを処理したりできます。

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。

   「**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。

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。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

関連タスク

361 ページの『第 7 章 Business Process Choreographer Explorer の概要』
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・タスクを管理したり、割り当てられたタスクを処理したりできます。

タスク・インスタンスの中断と再開

タスク・インスタンスを中断できます。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。

   ナビゲーション・ペインの「タスク・インスタンス」の下で、「**自分で管理**」をクリックします。
2. 「タスク・インスタンス」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。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

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、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。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

作業割り当ての管理

タスクの開始後に、そのタスクに対する作業割り当ての管理が必要になる場合があります (例えば、ワークグループのメンバー間でワークロードを効率よく分散するため)。

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・インスタンスには幾つかの作業項目を関連付けることができます。例えば、John、Sarah、Mike はすべてタスク・インスタンスの潜在的な所有者であり、Anne は管理者です。作業項目は、この 4 人全員に対して生成されます。 John、Sarah、Mike のタスク・リストには、それぞれ自分の作業項目だけがタスクとして表示されます。Anne

は管理者なので、自分のタスクの作業項目を取得し、さらに John、Sarah、Mike に対して生成された作業項目を管理できます。

場合によっては、タスクの開始後にタスク割り当てを変更する必要があります。例えば、元の所有者から別のユーザーへ作業項目を転送したり、不在時間中の不在設定を指定したりします。 追加の作業項目の作成や、必要なくなった作業項目の削除を実行しなければならない場合もあります。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

関連タスク

361 ページの『第 7 章 Business Process Choreographer Explorer の概要』
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・タスクを管理したり、割り当てられたタスクを処理したりできます。

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。

   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。

   「**新規所有者**」フィールドで、新規タスク所有者のユーザー ID を指定し、「**転送**」をクリックします。タスクは、別の潜在的所有者またはタスク管理者にのみ転送できます。

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。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

関連タスク

406 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。

408 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。

自分がスターター、オリジネーター、またはタスクの管理者である作業項目の転送

タスクで作業を始めた後に作業割り当てを変更する必要がでてくる場合があります。例えば、タスクの所有者が休暇中で、その所有者が出社する前にタスクを完了しなければならない場合、作業項目を別のユーザーに転送する必要があります。所有しているロールとタスクの状態によって作業項目の転送方法は変わります。

始める前に

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	割り当て理由	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。
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	リーダーまたは管理者	非アクティブ状態以外のすべての状態	リーダー、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。

ナビゲーション・ペインの「タスク・インスタンス」の下で、「**自分で管理**」をクリックします。

2. タスク・インスタンスの作業項目を表示します。

   「自分で管理するタスク・インスタンス」ページで、タスク・インスタンスの横にあるチェック・ボックスを選択して「**作業項目**」をクリックします。

3. 作業項目を転送します。

   a. 「**新規所有者**」フィールドで、新規作業項目所有者のユーザー ID を指定します。

   b. 1 つ以上の作業項目を選択し、「**転送**」をクリックします。

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。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

関連タスク

408 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。

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には、置換用の Virtual Member Manager (VMM) 担当者ディレクトリー・プロバイダーが必要です。

このタスクについて

適用された置換ポリシーに従って、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。置換ポリシーはタスク・テンプレートごとに異なってもかまいません。

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。 適用された置換ポリシーに従って、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。置換ポリシーはタスク・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. BPESystemAdministrator に担当者の照会結果を最新表示するように依頼します。

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。

関連概念

99 ページの『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。

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

関連タスク

404 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)。

225 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』
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担当者割り当てを実行するための、Business Process Choreographer 用の 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。デフォルトの担当者ディレクトリー・プロバイダーはいつでも使用できるので、カスタムの担当者割り当て基準を導入する場合に必要なのは、単に構成することだけです。

333 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。

355 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。

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。このタスクを実行するには、置換用の Virtual Member Manager (VMM) 担当者ディレクトリー・プロバイダーが必要です。

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。 適用された置換ポリシーに従って、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。置換ポリシーはタスク・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。

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。

関連概念

99 ページの『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。

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

関連タスク

405 ページの『自分がスターター、オリジネーター、またはタスクの管理者である作業項目の転送』
タスクで作業を始めた後に作業割り当てを変更する必要がでてくる場合がありま

す。例えば、タスクの所有者が休暇中で、その所有者が出社する前にタスクを完了しなければならない場合、作業項目を別のユーザーに転送する必要があります。所有しているロールとタスクの状態によって作業項目の転送方法は変わります。

404 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)。

225 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』
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担当者割り当てを実行するための、Business Process Choreographer 用の 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。デフォルトの担当者ディレクトリー・プロバイダーはいつでも使用できるので、カスタムの担当者割り当て基準を導入する場合に必要なのは、単に構成することだけです。

333 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。

355 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。

作業項目の作成

例えば現在の潜在的な所有者が追加作業を受け入れられない場合などに、新規の潜在的な所有者の作業項目を作成できます。また、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、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。

   ナビゲーション・ペインの「タスク・インスタンス」の下で、**「自分で管理」**をクリックします。

2. 作業項目を作成する対象のタスク・インスタンスの横にあるチェック・ボックスを選択し、「**作業項目の作成**」をクリックします。 「作業項目の作成」ページが表示されます。
3. 作業項目を作成します。
   a. 「**新規所有者**」フィールドで、新規作業項目所有者のユーザー ID を指定します。
   b. 「**理由**」リストから 1 つ以上のロールを選択します。

      これらのロールにより、割り当てられたユーザーが新規作業項目で実行できるアクションが決まります。
   c. 「**作成**」をクリックします。

結果

作業項目は、新規所有者に対して指定する各ロールに対して作成されます。新規タスクは、このユーザーに割り当てられるタスクのリストに表示されます。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

作業項目の削除

例えば、エラーの作業項目を作成した場合や、もう会社で働いていない人に対して作業項目が生成されている場合は、作業項目を削除することができます。

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

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。

   ナビゲーション・ペインの「タスク・インスタンス」の下で、「**自分で管理**」をクリックします。
2. タスク・インスタンスの作業項目を表示します。

   「自分で管理するタスク・インスタンス」ページで、タスク・インスタンスを選択して「**作業項目**」をクリックします。
3. 作業項目を削除します。

1 つ以上の作業項目を選択し、「**削除**」をクリックします。

結果

作業項目が削除されます。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

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。

手順

エスカレーションを表示するには、「タスク・インスタンス」の下にある「**自分のエスカレーション**」をクリックします。

- エスカレーションに関する情報を表示するには、エスカレーション ID をクリックします。
- エスカレートされたタスクに関する情報を表示するには、タスク名をクリックします。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

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「**担当者ディレクトリー (JNDI 名) (People directory (JNDI name))**」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 (Notification type)**」フィールドの値を `E-mail` に設定します。
   c. エスカレーション通知のために送信される E メールの本文用テキストを指定します。

      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 (Add Variable)**」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。

関連概念

43 ページの『第 2 章 ヒューマン・タスクについて』
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。

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、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は、プロセス、アクティビティー、タスクのいずれかの状態が変更されると、イベントが生成できます。これらのイベントを保管し、Business Process Choreographer Observer によるレポート作成に利用できます。例えば、プロセス・ボトルネックの分析レポートや、アクティビティーから呼び出されるサービスの信頼性評価のレポートなどです。

このタスクについて

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。

関連概念

『スナップショット・レポート』
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。

416 ページの『期間レポート』
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。

418 ページの『時間処理』
作成するレポートでは、Business Process Choreographer Observer が処理するタイム・スタンプと期間を考慮します。

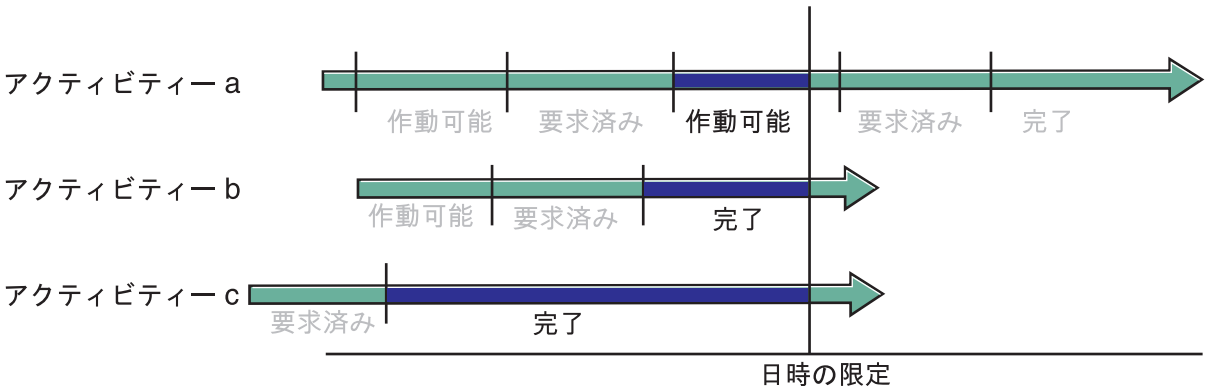
関連タスク

379 ページの『第 8 章 Business Process Choreographer Observer の概要』
ビジネス・プロセスおよびタスクが実行されている間、WebSphere® Process Server は、プロセス・インスタンスとそれに関連したアクティビティーの状態変更に関する情報を含むイベントを発行できます。 Business Process Choreographer Observer 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します。

スナップショット・レポート

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。

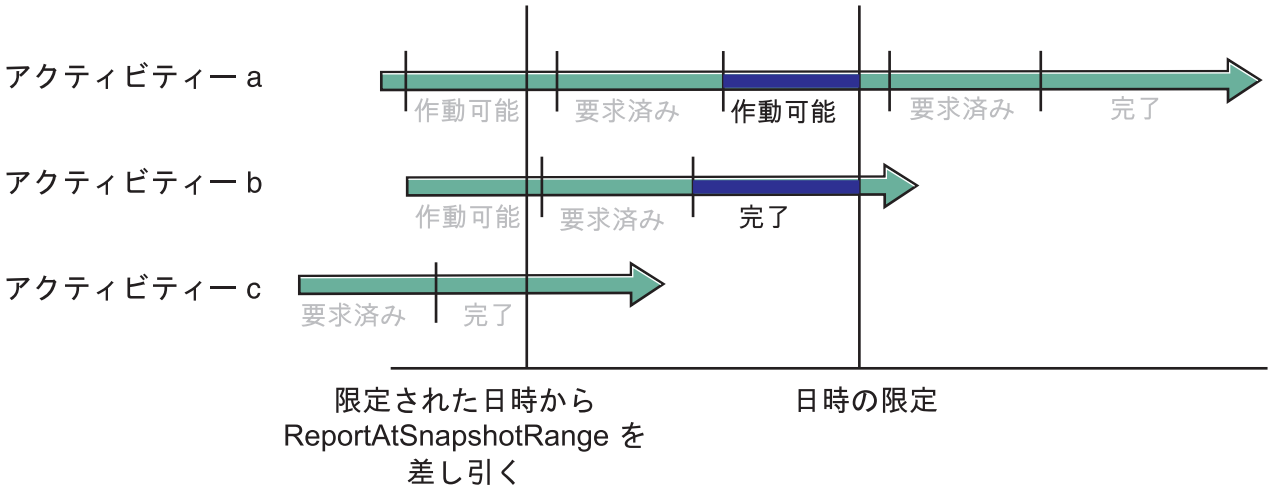
例えば、深夜に実行されているプロセス・インスタンスの数を知りたいとします。Business Process Choreographer Observer は、プロセスまたはアクティビティーのインスタンスごとに、指定された日時より前の最後のイベントを検出して結果の状態を評価します。以下の状態遷移は、スナップショット・レポートの対象となるイベントをどのように絞り込むかを示しています。



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、完了状態の 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

Observer 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、完了状態の 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

421 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・インスタンス状態の分布を表示させるには、事前定義のスナップショット・グラフを使用します。

425 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。

413 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』
ビジネス・プロセスおよびアクティビティーの処理中には、プロセス、アクティビティー、タスクのいずれかの状態が変更されると、イベントが生成できます。これらのイベントを保管し、Business Process Choreographer Observer によるレポート作成に利用できます。例えば、プロセス・ボトルネックの分析レポートや、アクティビティーから呼び出されるサービスの信頼性評価のレポートなどです。

419 ページの『事前定義のリストおよび図表の使用』
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・アプローチ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。

424 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、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。

関連資料

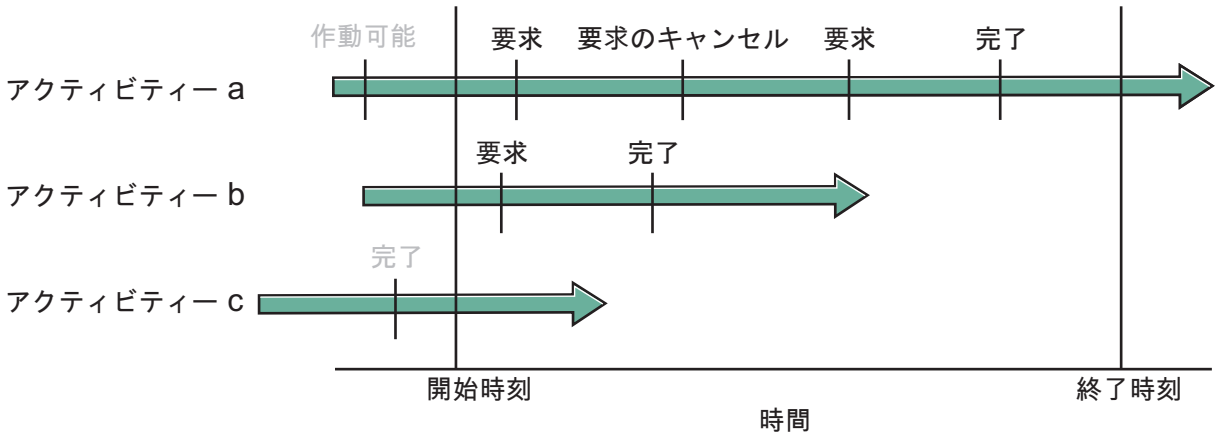
294 ページの『Business Process Choreographer Observer の構成パラメーターの変更』
Business Process Choreographer Observer アプリケーションと Event Collector アプリケーションの構成パラメーターを調整することは、検査を使用可能にし、パフォーマンスを向上させるために重要です。

期間レポート

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。



これは、この期間内に完了したイベントの数を照会すると、結果が 2 になることを意味します。

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

422 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、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。

428 ページの『ユーザー定義期間レポートの作成』
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。

413 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』
ビジネス・プロセスおよびアクティビティーの処理中には、プロセス、アクティビティー、タスクのいずれかの状態が変更されると、イベントが生成できます。これらのイベントを保管し、Business Process Choreographer Observer によるレポート作成に利用できます。例えば、プロセス・ボトルネックの分析レポートや、アクティビティーから呼び出されるサービスの信頼性評価のレポートなどです。

419 ページの『事前定義のリストおよび図表の使用』
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・アプローチ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。

424 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、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。

時間処理

作成するレポートでは、Business Process Choreographer Observer が処理するタイム・スタンプと期間を考慮します。

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

413 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』
ビジネス・プロセスおよびアクティビティーの処理中には、プロセス、アクティビティー、タスクのいずれかの状態が変更されると、イベントが生成できます。これらのイベントを保管し、Business Process Choreographer Observer によるレポート作成に利用できます。例えば、プロセス・ボトルネックの分析レポートや、アクティビティーから呼び出されるサービスの信頼性評価のレポートなどです。

419 ページの『事前定義のリストおよび図表の使用』
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・アプローチ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。

424 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、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。

事前定義のリストおよび図表の使用

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・アプローチ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

414 ページの『スナップショット・レポート』
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。

416 ページの『期間レポート』
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。

418 ページの『時間処理』
作成するレポートでは、Business Process Choreographer Observer が処理するタイム・スタンプと期間を考慮します。

関連タスク

423 ページの『例: 事前定義のグラフの使用』
このシナリオでは、事前定義のグラフの使用例を示します。

420 ページの『例: 事前定義のリストの使用』
このシナリオでは、事前定義のリスト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、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。

手順

1. リスト・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

418 ページの『時間処理』
作成するレポートでは、Business Process Choreographer Observer が処理するタイム・スタンプと期間を考慮します。

関連タスク

『例: 事前定義のリストの使用』
このシナリオでは、事前定義のリスト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、事前定義のリストの使用例を示します。

始める前に

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。

このタスクについて

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、以下のように事前定義のリストを使用します。

手順

1. ナビゲーション・バーの「**リスト**」セクションで、「**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。

関連タスク

419 ページの『事前定義のリストおよび図表の使用』
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・アプローチ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。

419 ページの『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、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。

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・インスタンス状態の分布を表示させるには、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

グラフを表示させるには、ブラウザーで Macromedia Flash Player が実行可能でなければなりません。

手順

1. 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

414 ページの『スナップショット・レポート』
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。

418 ページの『時間処理』
作成するレポートでは、Business Process Choreographer Observer が処理するタイム・スタンプと期間を考慮します。

関連タスク

423 ページの『例: 事前定義のグラフの使用』
このシナリオでは、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、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

グラフを表示させるには、ブラウザーで Macromedia Flash Player が実行可能でなければなりません。

このタスクについて

例えば、定義済みのグラフを使用すると、直近 12 カ月に完了したプロセス・インスタンスの分布が表示されます。

手順

1. 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

416 ページの『期間レポート』
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。

418 ページの『時間処理』
作成するレポートでは、Business Process Choreographer Observer が処理するタイム・スタンプと期間を考慮します。

関連タスク

『例: 事前定義のグラフの使用』
このシナリオでは、事前定義のグラフの使用例を示します。

例: 事前定義のグラフの使用

このシナリオでは、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、以下のようにして、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。

手順

1. ナビゲーション・バーの「**グラフ**」セクションで、「期間別プロセス (Process by period)」を選択し、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。

関連タスク

419 ページの『事前定義のリストおよび図表の使用』
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・アプローチ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。

421 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・インスタンス状態の分布を表示させるには、事前定義のスナップショット・グラフを使用します。

422 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、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、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

414 ページの『スナップショット・レポート』
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。

416 ページの『期間レポート』
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。

418 ページの『時間処理』
作成するレポートでは、Business Process Choreographer Observer が処理するタイム・スタンプと期間を考慮します。

関連タスク

431 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、ユーザー定義レポートの使用例を示します。

関連資料

433 ページの『レポート属性』
属性を使用して、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。

435 ページの『Business Process Choreographer Observer 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 Observer で使用可能です。

437 ページの『パフォーマンス関連の属性』
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。

このタスクについて

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. 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

414 ページの『スナップショット・レポート』
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。

418 ページの『時間処理』
作成するレポートでは、Business Process Choreographer Observer が処理するタイム・スタンプと期間を考慮します。

関連タスク

431 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、ユーザー定義レポートの使用例を示します。

関連資料

433 ページの『レポート属性』
属性を使用して、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。

435 ページの『Business Process Choreographer Observer 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 Observer で使用可能です。

437 ページの『パフォーマンス関連の属性』
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

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。

このタスクについて

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. 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

416 ページの『期間レポート』
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。

418 ページの『時間処理』
作成するレポートでは、Business Process Choreographer Observer が処理するタイム・スタンプと期間を考慮します。

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、ユーザー定義レポートの使用例を示します。

関連資料

433 ページの『レポート属性』
属性を使用して、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。

435 ページの『Business Process Choreographer Observer 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 Observer で使用可能です。

437 ページの『パフォーマンス関連の属性』
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、ユーザー定義レポートの使用例を示します。

始める前に

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 人は、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。

このタスクについて

「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。

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 Observer ではビジネス・オブジェクトの内容を利用できません。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. ナビゲーション・バーの「**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. 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」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. 「レポートの結果」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、「**編集**」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・バーの「**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」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。

関連タスク

424 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、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。

425 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。

428 ページの『ユーザー定義期間レポートの作成』
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。

レポート属性

属性を使用して、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

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
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

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・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
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 Observer 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 Observer 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。

- 436 ページの『プロセス・イベント』
- 436 ページの『アクティビティー・イベント』

WebSphere Integration Developer の設定に応じて、6.0.2 イベントと 6.1 イベントが発生します。

Business Process Choreographer Observer では、イベントのビジネス・データは不要です。

プロセス・イベント

以下の表に、Business Process Choreographer Observer 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
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
42046	プロセス補正が失敗した
42009	プロセスが強制終了中
42010	プロセスが失敗する

アクティビティー・イベント

以下の表に、Business Process Choreographer Observer 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	要求がキャンセルされた
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

関連資料

656 ページの『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。

パフォーマンス関連の属性

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

Observer 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。

関連資料

294 ページの『Business Process Choreographer Observer の構成パラメーターの変更』
Business Process Choreographer Observer 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。

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。

手順

1. 保存したレポート定義を実行するには、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

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。

このタスクについて

保存したレポート定義を非同期に実行するには、「**ポップアップ・メニューの表示)**」アイコン () をクリックし、「非同期検索 (Asynchronous Search)」アイコン () をクリックします。

手順

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

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。

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。

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、「**ポップアップ・メニューの表示 (Show pop-up menu)**」 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。

Windows プラットフォームの場合は、java -jar bpcobserverexporter.jar *options* と入力します。

Linux、UNIX、および i5/OS プラットフォームの場合は、java -jar bpcobserverexporter.jar *options* と入力します。

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 16. 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 Observer 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。

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

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。

手順

1. 「**ポップアップ・メニューの表示 (Show pop-up menu)**」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

418 ページの『時間処理』
作成するレポートでは、Business Process Choreographer Observer が処理するタイム・スタンプと期間を考慮します。

関連資料

433 ページの『レポート属性』
属性を使用して、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。

435 ページの『Business Process Choreographer Observer 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 Observer で使用可能です。

437 ページの『パフォーマンス関連の属性』
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するには、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

削除したレポート定義を復元することはできません。

手順

レポート定義を削除するには、「**ポップアップ・メニューの表示 (Show pop-up menu)**」アイコン () をクリックし、「**削除**」アイコン () をクリックします。

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 10 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。

関連概念

18 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、SCA (Service Component Architecture) コンポーネントの実装タイプです。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。

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、および Java Message Service (JMS) 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、および JMS 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
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。
データ処理における考慮事項	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。
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。
セキュリティー	Java 2, Enterprise Edition (J2EE) のセキュリティー。	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 2, Enterprise Edition (J2EE) 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。

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インプリメントするシナリオに応じて、2 つのセッション Bean のうちの 1 つ、または両方を使用することができます。
3. アプリケーションのユーザーが必要とする許可権限を判別します。

   アプリケーションのユーザーは、アプリケーションに組み込まれたメソッドを呼び出すこと、およびそれらのメソッドが戻すオブジェクトとそのオブジェクトの属性を表示するのに適した権限ロールを割り当てられていなければなりません。該当するセッション Bean のインスタンスを作成するときに、WebSphere Application Server がコンテキストとそのインスタンスを関連付けます。コンテキストには呼び出し元のプリンシパル ID、グループ・メンバーシップ・リスト、およびロールに関する情報が含まれています。この情報は、それぞれの呼び出しごとに、呼び出し元の権限を確認するために使用されます。

   Javadoc には、各メソッドの許可情報が含まれています。

4. アプリケーションをレンダリングする方法を決めます。

   EJB API は、ローカル側でもリモート側でも呼び出すことができます。
5. アプリケーションを開発します。
   a. EJB API にアクセスします。
   b. EJB API を使用して、プロセスまたはタスクと対話します。
      - データを照会します。
      - データで作業を行います。

EJB API へのアクセス

Enterprise JavaBeans (EJB) API は、次の 2 種類のステートレス・セッション・エンタープライズ Bean として提供されます。ビジネス・プロセス・アプリケーションおよびタスク・アプリケーションは、Bean のホーム・インターフェースを介して、適切なセッション・エンタープライズ Bean にアクセスします。

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

EJB クライアント・アプリケーションは、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 2 Platform Enterprise Edition (J2EE) クライアント・アプリケーションの場合は、application-client.xml ファイル
   - Web アプリケーションの場合は、web.xml ファイル
   - Enterprise JavaBeans (EJB) アプリケーションの場合は、ejb-jar.xml ファイル

   プロセス・アプリケーションの場合のリモート・ホーム・インターフェースへの参照は、以下の例で示されます。

```
<ejb-ref>
 <ejb-ref-name>ejb/BusinessFlowManagerHome</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <home>com.ibm.bpe.api.BusinessFlowManagerHome</home>
 <remote>com.ibm.bpe.api.BusinessFlowManager</remote>
</ejb-ref>
```

タスク・アプリケーションの場合のリモート・ホーム・インターフェースへの参照は、以下の例で示されます。

```
<ejb-ref>
 <ejb-ref-name>ejb/HumanTaskManagerHome</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <home>com.ibm.task.api.HumanTaskManagerHome</home>
 <remote>com.ibm.task.api.HumanTaskManager</remote>
</ejb-ref>
```

WebSphere Integration Developer を使用して EJB 参照をデプロイメント記述子に追加する場合、EJB 参照のバインディングが、アプリケーションのデプロイ時に自動的に作成されます。EJB 参照の追加について詳しくは、WebSphere Integration Developer の文書を参照してください。

2. 生成されたスタブをアプリケーションにパッケージします。

   ご使用のアプリケーションが、BPEContainer アプリケーションまたは TaskContainer アプリケーションを実行しているのと異なる Java 仮想マシン (JVM) で実行されている場合、以下のアクションを完了します。

   a. プロセス・アプリケーションの場合、<*install_root*>/ProcessChoreographer/client/bpe137650.jar ファイルを、ご使用のアプリケーションのエンタープライズ・アーカイブ (EAR) ファイルにパッケージします。
   b. タスク・アプリケーションの場合、<*install_root*>/ProcessChoreographer/client/task137650.jar ファイルを、ご使用のアプリケーションの EAR ファイルにパッケージします。
   c. アプリケーション・モジュールのマニフェスト・ファイル内の **Classpath** パラメーターを、JAR ファイルを含めるように設定します。

      アプリケーション・モジュールは、J2EE アプリケーション、Web アプリケーション、または EJB アプリケーションの可能性があります。

3. 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 J2EE 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 J2EE 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 を使用して入力されるビジネス・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();
   }
```

4. 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。

5. セッション Bean のリモート・インターフェースにアクセスします。

   以下の例では、プロセス・アプリケーションでのこのステップを示します。

   ```
   BusinessFlowManager process = processHome.create();
   ```

   セッション Bean へのアクセス権は、呼び出し元が Bean が提供するすべてのアクションを実行できることを保証するものではありません。呼び出し元には、そのアクションに対する許可も必要になります。セッション Bean のインスタンスを作成すると、コンテキストとそのセッション Bean のインスタンスが関連付けられます。コンテキストには呼び出し元のプリンシパル ID、グループ・メンバーシップ・リストが含まれ、その呼び出し元に Business Process Choreographer の J2EE ロールのいずれかがあるかどうかを示します。コンテキストは、グローバル・セキュリティーが設定されていなくても、それぞれの呼び出しごとに、呼び出し元の権限を確認するために使用されます。グローバル・セキュリティーが設定されていない場合、呼び出し元のプリンシパル ID の値は UNAUTHENTICATED になります。

6. 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

EJB クライアント・アプリケーションは、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 2 Platform Enterprise Edition (J2EE) クライアント・アプリケーションの場合は、application-client.xml ファイル
- Web アプリケーションの場合は、web.xml ファイル
- Enterprise JavaBeans (EJB) アプリケーションの場合は、ejb-jar.xml ファイル

プロセス・アプリケーションの場合のローカル・ホーム・インターフェースへの参照は、以下の例で示されます。

```
<ejb-local-ref>
 <ejb-ref-name>ejb/LocalBusinessFlowManagerHome</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <local-home>com.ibm.bpe.api.LocalBusinessFlowManagerHome</local-home>
 <local>com.ibm.bpe.api.LocalBusinessFlowManager</local>
</ejb-local-ref>
```

タスク・アプリケーションの場合のローカル・ホーム・インターフェースへの参照は、以下の例で示されます。

```
<ejb-local-ref>
 <ejb-ref-name>ejb/LocalHumanTaskManagerHome</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <local-home>com.ibm.task.api.LocalHumanTaskManagerHome</local-home>
 <local>com.ibm.task.api.LocalHumanTaskManager</local>
</ejb-local-ref>
```

WebSphere Integration Developer を使用して EJB 参照をデプロイメント記述子に追加する場合、EJB 参照のバインディングが、アプリケーションのデプロイ時に自動的に作成されます。EJB 参照の追加について詳しくは、WebSphere Integration Developer の文書を参照してください。

2. Java Naming and Directory Interface (JNDI) からセッション 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 J2EE ロールのいずれかがあるかどうかを示します。コンテキストは、グローバル・セキュリティーが設定されていなくても、それぞれの呼び出しごとに、呼び出し元の権限を確認するために使用されます。グローバル・セキュリティーが設定されていない場合、呼び出し元のプリンシパル ID の値は UNAUTHENTICATED になります。

4. サービス・インターフェースによって公開されたビジネス関数を呼び出します。

   以下の例では、プロセス・アプリケーションでのこのステップを示します。

   ```
   process.initiate("MyProcessModel",input);
   ```

   アプリケーションからの呼び出しは、トランザクションとして実行されます。トランザクションは、以下のいずれかの方法で確立されて終了します。
   - WebSphere Application Server から自動的に (デプロイメント記述子が TX_REQUIRED を指定)。
   - アプリケーションから明示的に。アプリケーションの呼び出しを 1 つのトランザクションにバンドルすることができます。

     ```
     // Obtain user transaction interface
     UserTransaction transaction=
            (UserTransaction)initialContext.lookup("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・プロセスおよびタスク関連のオブジェクトの照会

クライアント・アプリケーションは、ビジネス・プロセスとタスク関連オブジェクトを操作します。データベース内のビジネス・プロセス・オブジェクトおよびタスク関連のオブジェクトを照会して、これらのオブジェクトの特定のプロパティーを取得することができます。

このタスクについて

Business Process Choreographer を構成すると、リレーショナル・データベースは、ビジネス・プロセス・コンテナーおよびタスク・コンテナーの両方に関連付けられます。このデータベースは、ビジネス・プロセスとタスクの管理用のすべてのテンプレート (モデル) とインスタンス (ランタイム) のデータを保管します。そのデータを照会するには、SQL 形式の構文を使用します。

単発の照会を実行して、オブジェクトの特定のプロパティーを取得することができます。また、頻繁に使用する照会を保管しておいて、この保管照会文をアプリケーションに組み込むこともできます。

ビジネス・プロセスおよびタスク関連オブジェクトに対する照会

サービス API の query メソッドまたは queryAll メソッドを使用して、ビジネス・プロセスおよびタスクに関する保管情報を取得します。

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、J2EE 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、J2EE ロール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 17.

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);`
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。select 文節に含まれている列を指定することをお勧めします。

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インプリメントします。

このパラメーターを null に設定したときに、しきい値パラメーターを設定していないと、すべての適格なタプルが戻されます。

スキップ・タプル・パラメーターの例

- new Integer(5)

  最初の 5 つの適格なタプルを戻さないように指定します。

しきい値パラメーター:

照会関数のしきい値パラメーターは、照会の結果セットとしてサーバーからクライアントに戻されるオブジェクトの数を制限します。

実動シナリオにおける照会結果セットは数千から数百万の項目を含む場合さえあるため、常にしきい値を指定することがベスト・プラクティスとなります。しきい値パラメーターは、例えば、同時に少数の項目のみが表示されるグラフィカル・ユーザー・インターフェースなどで有用な場合があります。しきい値パラメーターを適宜設定すると、データベース照会が高速化し、サーバーからクライアントへ転送する必要のあるデータが少なくなります。

このパラメーターを null に設定したときに、スキップ・タプル・パラメーターを設定していないと、適格なオブジェクトがすべて戻されます。

しきい値パラメーターの例

- new Integer(50)

  50 個の適格なタプルを戻すように指定します。

時間帯パラメーター:

照会関数の時間帯パラメーターは、照会内のタイム・スタンプ定数の時間帯を定義します。

照会を開始するクライアントと照会を処理するサーバーの間で、時間帯が異なることがあります。時間帯パラメーターを使用して、where 文節で使用されるタイム・スタンプ定数の時間帯を、例えば地方時を指定するように指定します。照会の結果セットで戻される日付には、照会で指定したものと同じ時間帯が設定されます。

このパラメーターを null に設定すると、タイム・スタンプ定数は協定世界時 (UTC) と想定されます。

時間帯パラメーターの例

- ```
  process.query("ACTIVITY.AIID",
                "ACTIVITY.STARTED > TS('2005-01-01T17:40')",
                 (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。

保管照会文のパラメーター:

保管照会文は、データベースに保管され、名前で識別される照会のことです。照会が実行されると、修飾するタプルが動的にアセンブルされます。保管照会文を再使用可能にするために、実行時に解決されるパラメーターを照会定義で使用できます。

例えば、顧客名を保管するカスタム・プロパティーを定義したとします。特定の顧客 ACME Co. に関連したタスクを戻すように、照会を定義することができます。この情報を照会する場合、照会内の where 文節は以下の例のようになります。

```
String whereClause =
  "TASK.STATE = TASK.STATE.STATE_READY
  AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER
  AND TASK_CPROP.NAME = 'company' AND TASK_CPROP.STRING_VALUE = 'ACME Co.'";
```

顧客 BCME Ltd. も検索できるようにこの照会を再使用可能にするには、カスタム・プロパティーの値に対してパラメーターを使用できます。パラメーターをタスク照会に追加する場合、以下の例のようになります。

```
String whereClause =
  "TASK.STATE = TASK.STATE.STATE_READY
  AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER
  AND TASK_CPROP.NAME = 'company' AND TASK_CPROP.STRING_VALUE = '@param1'";
```

`@param1` パラメーターは、query メソッドに受け渡されるパラメーターのリストから、実行時に解決されます。以下の規則は、照会内でのパラメーターの使用に適用されます。

- パラメーターは where 文節でのみ使用できる。
- パラメーターはストリングである。
- パラメーターは実行時にストリングの置換を使用して置き換えられる。特殊文字が必要な場合、where 文節に指定するか、実行時にパラメーターの一部として受け渡す必要があります。
- パラメーター名は、ストリング `@param` を整数と連結したもので構成される。最小番号は 1 で、実行時に照会 API に受け渡されるパラメーターのリスト内の最初の項目を指します。
- パラメーターは where 文節内で複数回使用できます。出現するパラメーターはすべて同じ値で置き換えられます。

照会結果:

照会結果セットには、照会の結果が入ります。

結果セットのエレメントは、呼び出し元により指定された 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。

ビジネス・プロセス・オブジェクトおよびヒューマン・タスク・オブジェクトの照会のための事前定義ビュー

ビジネス・プロセス・オブジェクトおよびヒューマン・タスク・オブジェクト用に、事前定義データベース・ビューが提供されています。これらのオブジェクトの参照データを照会する場合は、これらのビューを使用します。

定義済みビューを使用する場合は、ビューの列用に明示的に結合述部を追加する必要はありません。これらの構成要素は自動的に追加されます。このデータを照会す

るには、サービス API (BusinessFlowManagerService または HumanTaskManagerService) の汎用照会関数を使用することができます。HumanTaskManagerDelegate API の対応するメソッド、または ExecutableQuery インターフェースのインプリメンテーションによって提供される定義済み照会を使用することもできます。

注: 説明されていない列がビューに含まれていることがあります。このような列は内部でのみ使用される列です。

ACTIVITY ビュー:

この定義済みデータベース・ビューは、アクティビティーの照会に使用します。

表 18. 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。
KIND	整数	アクティビティーの種類。指定可能な値は、以下のとおりです。 KIND_INVOKE (21) KIND_RECEIVE (23) KIND_REPLY (24) KIND_THROW (25) KIND_RETHROW (46) KIND_TERMINATE (26) KIND_WAIT (27) KIND_COMPENSATE (29) KIND_SEQUENCE (30) KIND_EMPTY (3) KIND_SWITCH (32) KIND_WHILE (34) KIND_PICK (36) KIND_FLOW (38) KIND_SCOPE (40) KIND_SCRIPT (42) KIND_STAFF (43) KIND_ASSIGN (44) KIND_CUSTOM (45) KIND_FOR_EACH_PARALLEL (49) KIND_FOR_EACH_SERIAL (47)
COMPLETED	タイム・スタンプ	アクティビティーの完了時刻。
ACTIVATED	タイム・スタンプ	アクティビティーが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。

表 18. ACTIVITY ビュー内の列 (続き)

列名	タイプ	コメント
STARTED	タイム・スタンプ	アクティビティーの開始時刻。
STATE	整数	アクティビティーの状態。指定可能な値は、以下のとおりです。 STATE_INACTIVE (1) STATE_READY (2) STATE_RUNNING (3) STATE_PROCESSING_UNDO (14) STATE_SKIPPED (4) STATE_FINISHED (5) STATE_FAILED (6) STATE_TERMINATED (7) STATE_CLAIMED (8) STATE_TERMINATING (9) STATE_FAILING (10) STATE_WAITING (11) STATE_EXPIRED (12) STATE_STOPPED (13)
OWNER	ストリング	所有者のプリンシパル ID。
DESCRIPTION	ストリング	アクティビティー・テンプレートの説明にプレースホルダーが含まれている場合、この列には、解決済みのプレースホルダーを所有するアクティビティー・インスタンスの説明が入ります。
TEMPLATE_NAME	ストリング	関連するアクティビティー・テンプレートの名前。
TEMPLATE_DESCR	ストリング	関連するアクティビティー・テンプレートの説明。
BUSINESS_RELEVANCE	ブール	アクティビティーがビジネスと関係があるかどうかを指定します。指定可能な値は、以下のとおりです。 **TRUE** アクティビティーはビジネスに関係があります。Business Process Choreographer Explorer でアクティビティー状況を表示できます。 **FALSE** アクティビティーはビジネスに関係がありません。
EXPIRES	タイム・スタンプ	アクティビティーの期限が切れる日時。アクティビティーの期限が切れている場合は、このイベントが発生したときの日時。

ACTIVITY_ATTRIBUTE ビュー:

この定義済みデータベース・ビューは、アクティビティーのカスタム・プロパティーの照会に使用します。

表 19. ACTIVITY_ATTRIBUTE ビュー内の列

列名	タイプ	コメント
AIID	ID	カスタム・プロパティーを所有するアクティビティー・インスタンスの ID。
NAME	ストリング	カスタム・プロパティーの名前。
VALUE	ストリング	カスタム・プロパティーの値。

ACTIVITY_SERVICE ビュー:

この定義済みデータベース・ビューは、アクティビティー・サービスの照会に使用します。

表 20. 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	ネーム・スペースの URI。
OPERATION	ストリング	サービスのオペレーション名。

APPLICATION_COMP ビュー:

この定義済みデータベース・ビューは、アプリケーション・コンポーネント ID、およびタスクのデフォルト設定の照会に使用します。

表 21. APPLICATION_COMP ビュー内の列

列名	タイプ	コメント
ACOID	ストリング	アプリケーション・コンポーネントの ID。
BUSINESS_ RELEVANCE	ブール	コンポーネントのデフォルトのタスク・ビジネス関連ポリシー。この値は、タスク・テンプレートまたはタスクの定義によって上書きされる場合があります。属性は、監査証跡へのロギングに影響します。指定可能な値は、以下のとおりです。 **TRUE** タスクはビジネスと関係があり、監査されます。 **FALSE** タスクはビジネスとの関係がなく、監査されません。
NAME	ストリング	アプリケーション・コンポーネントの名前。

表 21. APPLICATION_COMP ビュー内の列 (続き)

列名	タイプ	コメント
SUPPORT_ AUTOCLAIM	ブール	コンポーネントのデフォルトの自動要求ポリシー。この属性が TRUE に設定されている場合、潜在的な所有者が単一のユーザーであれば、タスクを自動的に要求することができます。この値は、タスク・テンプレートまたはタスクの定義によって上書きされる場合があります。
SUPPORT_CLAIM_ SUSP	ブール	中断されているタスクを要求できるかどうかを判断するコンポーネントのデフォルト設定。この属性が TRUE に設定されている場合は、中断されているタスクを要求することができます。この値は、タスク・テンプレートまたはタスクの定義によって上書きされる場合があります。
SUPPORT_ DELEGATION	ブール	コンポーネントのデフォルトのタスク代行ポリシー。この属性が TRUE に設定されている場合は、タスクの作業項目割り当てを変更することができます。すなわち、作業項目の作成、削除、または転送が可能です。
SUPPORT_ FOLLOW_ON	ブール	コンポーネントのデフォルトの後続タスク・ポリシー。この属性が TRUE に設定されている場合は、タスクの後続タスクを作成できます。この値は、タスク・テンプレートまたはタスクの定義によって上書きされる場合があります。
SUPPORT_ SUB_TASK	ブール	コンポーネントのデフォルトのサブタスク・ポリシー。この属性が TRUE に設定されている場合は、タスクのサブタスクを作成できます。この値は、タスク・テンプレートまたはタスクの定義によって上書きされる場合があります。

ESCALATION ビュー:

この定義済みデータベース・ビューは、エスカレーションのデータの照会に使用します。

表 22. ESCALATION ビュー内の列

列名	タイプ	コメント
ESIID	ストリング	エスカレーション・インスタンスの ID。
ACTION	整数	エスカレーションによって起動されるアクション。指定可能な値は、以下のとおりです。 **ACTION_CREATE_WORK_ITEM (1)** それぞれのエスカレーションに対する受信側の作業項目を作成します。 **ACTION_SEND_EMAIL (2)** それぞれのエスカレーションの受信側に E メールを送信します。 **ACTION_CREATE_EVENT (3)** イベントを作成および公表します。

表 22. ESCALATION ビュー内の列 (続き)

列名	タイプ	コメント
ACTIVATION_STATE	整数	対応するタスクが以下のいずれかの状態になると、エスカレーション・インスタンスが作成されます。 **ACTIVATION_STATE_READY (2)** ヒューマン・タスクまたは参加タスクは、要求を受ける準備ができていることを示します。 **ACTIVATION_STATE_RUNNING (3)** 親タスクが開始され、実行中であることを示します。 **ACTIVATION_STATE_CLAIMED (8)** タスクが要求されることを明示します。 **ACTIVATION_STATE_WAITING_ FOR_SUBTASK (20)** タスクがサブタスクの完了を待つことを示します。
ACTIVATION_TIME	タイム・スタンプ	エスカレーションがアクティブ化される時刻。
AT_LEAST_ EXP_STATE	整数	エスカレーションによって予期されるタスクの状態。タイムアウトが発生した場合、タスク状態がこの属性の値と比較されます。指定可能な値は、以下のとおりです。 **AT_LEAST_EXPECTED_STATE_CLAIMED (8)** タスクが要求されることを明示します。 **AT_LEAST_EXPECTED_STATE_ENDED (20)** タスクが最終状態 (FINISHED、FAILED、TERMINATED、または EXPIRED) にあることを明示します。 **AT_LEAST_EXPECTED_STATE_ SUBTASKS_COMPLETED (21)** タスクのすべてのサブタスクが完了していることを示します。
ESTID	ストリング	対応するエスカレーション・テンプレートの ID。
FIRST_ESIID	ストリング	チェーン内の最初のエスカレーションの ID。
INCREASE_PRIORITY	整数	タスクの優先度を増やす方法を示します。指定可能な値は、以下のとおりです。 **INCREASE_PRIORITY_NO (1)** タスクの優先度は増えません。 **INCREASE_PRIORITY_ONCE (2)** タスクの優先度は一度に 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。

表 22. ESCALATION ビュー内の列 (続き)

列名	タイプ	コメント
STATE	整数	エスカレーションの状態。指定可能な値は、以下のとおりです。 STATE_INACTIVE (1) STATE_WAITING (2) STATE_ESCALATED (3) STATE_SUPERFLUOUS (4)
TKIID	ストリング	エスカレーションが所属するタスク・インスタンス ID。

ESCALATION_CPROP ビュー:

この定義済みデータベース・ビューは、エスカレーションのカスタム・プロパティーを照会するために使用します。

表 23. ESCALATION_CPROP ビュー内の列

列名	タイプ	コメント
ESIID	ストリング	エスカレーション ID。
NAME	ストリング	プロパティーの名前。
DATA_TYPE	ストリング	非ストリング・カスタム・プロパティーのクラスのタイプ。
STRING_VALUE	ストリング	String 型のカスタム・プロパティーの値。

ESCALATION_DESC ビュー:

この定義済みデータベース・ビューは、エスカレーションのマルチリンガル記述データを照会するために使用します。

表 24. ESCALATION_DESC ビュー内の列

列名	タイプ	コメント
ESIID	ストリング	エスカレーション ID。
LOCALE	ストリング	説明または表示名に関連付けられているロケールの名前。
DESCRIPTION	ストリング	タスク・テンプレートの説明。
DISPLAY_NAME	ストリング	エスカレーションの記述名。

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 25. ESC_TEMPL ビュー内の列

列名	タイプ	コメント
ESTID	ストリング	エスカレーション・テンプレートの ID。

表 25. ESC_TEMPL ビュー内の列 (続き)

列名	タイプ	コメント
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。
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。

表 25. ESC_TEMPL ビュー内の列 (続き)

列名	タイプ	コメント
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	ストリング	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。
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	ストリング	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	ストリング	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 26. ESC_TEMPL_CPROP ビュー内の列

列名	タイプ	コメント
ESTID	ストリング	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	ストリング	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 27. ESC_TEMPL_DESC ビュー内の列

列名	タイプ	コメント
ESTID	ストリング	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	ストリング	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。
DISPLAY_NAME	ストリング	エスカレーションの記述名。

PROCESS_ATTRIBUTE ビュー:

この定義済みデータベース・ビューは、プロセスのカスタム・プロパティーの照会に使用します。

表 28. 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。
VALUE	ストリング	カスタム・プロパティーの値。

PROCESS_INSTANCE ビュー:

この定義済みデータベース・ビューは、プロセス・インスタンスの照会に使用します。

表 29. 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)

表 29. *PROCESS_INSTANCE ビュー内の列 (続き)*

列名	タイプ	コメント
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。
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。

PROCESS_TEMPLATE ビュー:

この定義済みデータベース・ビューは、プロセス・テンプレートの照会に使用します。

表 30. *PROCESS_TEMPLATE ビュー内の列*

列名	タイプ	コメント
PTID	ID	プロセス・テンプレート ID。
NAME	ストリング	プロセス・テンプレートの名前。
VALID_FROM	タイム・スタンプ	プロセス・テンプレートのインスタンス化が可能になる時刻。
TARGET_NAMESPACE	ストリング	プロセス・テンプレートのターゲット・ネーム・スペース。
APPLICATION_NAME	ストリング	プロセス・テンプレートが所属するエンタープライズ・アプリケーションの名前。
VERSION	ストリング	ユーザー定義のバージョン。

表 30. PROCESS_TEMPLATE ビュー内の列 (続き)

列名	タイプ	コメント
CREATED	タイム・スタンプ	プロセス・テンプレートがデータベース内に作成される時刻。
STATE	整数	プロセス・インスタンスの作成にプロセス・テンプレートを使用できるかどうかを指定します。指定可能な値は、以下のとおりです。 STATE_STARTED (1) STATE_STOPPED (2)
EXECUTION_MODE	整数	このプロセス・テンプレートから派生したプロセス・インスタンスの実行方法を指定します。指定可能な値は、以下のとおりです。 EXECUTION_MODE_MICROFLOW (1) EXECUTION_MODE_LONG_RUNNING (2)
DESCRIPTION	ストリング	プロセス・テンプレートの説明。
COMP_SPHERE	整数	プロセス・テンプレート内の microflow のインスタンスの補正の振る舞いを指定します。既存の補正範囲を結合するか、補正範囲を作成するかのいずれかです。 指定可能な値は、以下のとおりです。 COMP_SPHERE_REQUIRED (2) COMP_SPHERE_SUPPORTS (4)
DISPLAY_NAME	ストリング	プロセスの記述名。

QUERY_PROPERTY ビュー:

この定義済みデータベース・ビューは、プロセス・レベル変数の照会に使用します。

表 31. QUERY_PROPERTY ビュー内の列

列名	タイプ	コメント
PIID	ID	プロセス・インスタンス ID。
VARIABLE_NAME	ストリング	プロセス・レベル変数の名前。
NAME	ストリング	照会プロパティーの名前。
NAMESPACE	ストリング	照会プロパティーのネームスペース。
GENERIC_VALUE	ストリング	次のいずれかの定義済みタイプにマップしないプロパティー・タイプのストリング表記。 STRING_VALUE、 NUMBER_VALUE、 DECIMAL_VALUE、または TIMESTAMP_VALUE。

表 31. QUERY_PROPERTY ビュー内の列 (続き)

列名	タイプ	コメント
STRING_VALUE	ストリング	プロパティー・タイプがストリング・タイプにマップされる場合、これがストリングの値です。
NUMBER_VALUE	整数	プロパティー・タイプが整数タイプにマップされる場合、これが整数の値です。
DECIMAL_VALUE	10 進数	プロパティー・タイプが浮動小数点タイプにマップされる場合、これが 10 進数の値です。
TIMESTAMP_VALUE	タイム・スタンプ	プロパティー・タイプがタイム・スタンプ・タイプにマップされる場合、これがタイム・スタンプの値です。

TASK ビュー:

この定義済みデータベース・ビューは、タスク・オブジェクトの照会に使用します。

表 32. 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。
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。

表 32. TASK ビュー内の列 (続き)

列名	タイプ	コメント
CTX_ AUTHORIZATION	整数	タスクの所有者がタスク・コンテキストにアクセスできるようにします。指定可能な値は、以下のとおりです。 **AUTH_NONE** 関連コンテキスト・オブジェクトに対する許可権限はありません。 **AUTH_READER** 関連コンテキスト・オブジェクトに関する操作に、プロセス・インスタンスのプロパティーの読み取りなどのリーダー権限が必要です。
DUE	タイム・スタンプ	タスクの期限時刻。
EXPIRES	タイム・スタンプ	タスクの有効期限が切れる日付。
FIRST_ACTIVATED	タイム・スタンプ	タスクが初めて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。
IS_AD_HOC	ブール	このタスクが実行時に動的に作成されたのか、またはタスク・テンプレートから作成されたのかを示します。
IS_ESCALATED	ブール	このタスクのエスカレーションが発生済みかどうかを示します。
IS_INLINE	ブール	タスクがビジネス・プロセス内のインラインのタスクであるかどうかを示します。
IS_WAIT_FOR_ SUB_TK	ブール	親タスクが、サブタスクが終了状態になるのを待機しているかどうかを示します。

表 32. TASK ビュー内の列 (続き)

列名	タイプ	コメント
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_WPC_STAFF_ACTIVITY (102)** タスクが、WebSphere Business Integration Server Foundation バージョン 5 ビジネス・プロセスのスタッフ・アクティビティーであるヒューマン・タスクであることを示します。 **KIND_ORIGINATING (103)** 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。
NAME_SPACE	ストリング	タスクのカテゴリー化に使用されるネーム・スペース。
ORIGINATOR	ストリング	タスク・オリジネーターのプリンシパル ID。
OWNER	ストリング	タスクの所有者のプリンシパル ID。
PARENT_CONTEXT_ID	ストリング	このタスクの親コンテキスト。この属性は、呼び出し側アプリケーション・コンポーネント内の対応するコンテキストに対するキーを提供します。親コンテキストは、そのタスクを作成するアプリケーション・コンポーネントによって設定されます。
PRIORITY	整数	タスクの優先順位。
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。

表 32. TASK ビュー内の列 (続き)

列名	タイプ	コメント
STATE	整数	タスクの状態。指定可能な値は、以下のとおりです。 **STATE_READY (2)** そのタスクは要求を受ける準備ができたことを示します。 **STATE_RUNNING (3)** タスクが開始され、実行中であることを示します。 **STATE_FINISHED (5)** タスクが正常に完了したことを示します。 **STATE_FAILED (6)** タスクが正常に完了しなかったことを示します。 **STATE_TERMINATED (7)** 外部要求または内部要求が原因で、タスクが終了したことを示します。 **STATE_CLAIMED (8)** タスクが要求されることを示します。 **STATE_EXPIRED (12)** 指定された期間を超えたため、タスクが終了したことを示します。 **STATE_FORWARDED (101)** タスクが完了して、後続タスクがあることを示します。
SUPPORT_AUTOCLAIM	ブール	このタスクが単一ユーザーに割り当てられている場合に、自動的に要求されるかどうかを示します。
SUPPORT_CLAIM_SUSP	ブール	このタスクが中断されている場合に、要求可能かどうかを示します。
SUPPORT_DELEGATION	ブール	このタスクが、作業項目の作成、削除、または転送によって、作業代行をサポートするかどうかを示します。
SUPPORT_FOLLOW_ON	ブール	このタスクが後続タスクの作成をサポートするかどうかを示します。
SUPPORT_SUB_TASK	ブール	このタスクがサブタスクの作成をサポートするかどうかを示します。
SUSPENDED	ブール	タスクが中断されているかどうかを示します。
TKTID	ID	タスク・テンプレート ID。
TOP_TKIID	ID	これがサブタスクの場合、親タスク上位インスタンス ID。
TYPE	ストリング	タスクのカテゴリー化に使用されるタイプ。

TASK_CPROP ビュー:

この定義済みデータベース・ビューは、タスク・オブジェクトのカスタム・プロパティーを照会するために使用します。

表 33. TASK_CPROP ビュー内の列

列名	タイプ	コメント
TKIID	ストリング	タスク・インスタンス ID。
NAME	ストリング	プロパティーの名前。
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 34. TASK_DESC ビュー内の列

列名	タイプ	コメント
TKIID	ストリング	タスク・インスタンス ID。
LOCALE	ストリング	説明または表示名に関連付けられているロケールの名前。
DESCRIPTION	ストリング	タスクの説明。
DISPLAY_NAME	ストリング	タスクの記述名。

TASK_TEMPL ビュー:

この定義済みデータベース・ビューは、タスクのインスタンスを生成するために使用できるデータを保持します。

表 35. TASK_TEMPL ビュー内の列

列名	タイプ	コメント
TKTID	ストリング	タスク・テンプレート ID。
VALID_FROM	タイム・スタンプ	インスタンス化にタスク・テンプレートが使用できるようになる時刻。
APPLIC_ DEFAULTS_ID	ストリング	タスク・テンプレートのデフォルト値を指定するアプリケーション・コンポーネントの ID。
APPLIC_NAME	ストリング	タスク・テンプレートが所属するエンタープライズ・アプリケーションの名前。
BUSINESS_ RELEVANCE	ブール	タスク・テンプレートがビジネスと関係があるかどうかを指定します。 属性は、監査証跡へのロギングに影響します。指定可能な値は、以下のとおりです。 **TRUE** タスクはビジネスと関係があり、監査されます。 **FALSE** タスクはビジネスとの関係がなく、監査されません。

表 35. TASK_TEMPL ビュー内の列 (続き)

列名	タイプ	コメント
CONTAINMENT_CTX_ID	ID	このタスク・テンプレートの包含コンテキスト。この属性は、タスク・テンプレートのライフ・サイクルを決定します。包含コンテキストを削除すると、タスク・テンプレートも削除されます。
CTX_AUTHORIZATION	整数	タスクの所有者がタスク・コンテキストにアクセスできるようにします。指定可能な値は、以下のとおりです。 **AUTH_NONE** 関連コンテキスト・オブジェクトに対する許可権限はありません。 **AUTH_READER** 関連コンテキスト・オブジェクトに関する操作に、プロセス・インスタンスのプロパティーの読み取りなどのリーダー権限が必要です。
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ネーム・スペース。
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。

表 35. TASK_TEMPL ビュー内の列 (続き)

列名	タイプ	コメント
NAMESPACE	ストリング	タスク・テンプレートのカテゴリー化に使用されるネーム・スペース。
PRIORITY	整数	タスク・テンプレートの優先順位。
STATE	整数	タスク・テンプレートの状態。指定可能な値は、以下のとおりです。 **STATE_STARTED (1)** タスク・テンプレートをタスク・インスタンスの作成に使用できることを明示します。 **STATE_STOPPED (2)** タスク・テンプレートが停止されたことを明示します。この状態のタスク・テンプレートからタスク・インスタンスを作成することはできません。
SUPPORT_AUTOCLAIM	ブール	このタスク・テンプレートから派生したタスクが 1 人のユーザーに割り当てられた場合、タスクを自動的に要求できるかどうかを示します。
SUPPORT_CLAIM_SUSP	ブール	このタスク・テンプレートから派生したタスクが中断された場合、タスクを自動的に要求できるかどうかを示します。
SUPPORT_DELEGATION	ブール	このタスク・テンプレートから派生したタスクが、作業項目の作成、削除、または転送を使用して作業代行をサポートするかどうかを示します。
SUPPORT_FOLLOW_ON	ブール	タスク・テンプレートが後続タスクの作成をサポートするかどうかを示します。
SUPPORT_SUB_TASK	ブール	タスク・テンプレートがサブタスクの作成をサポートするかどうかを示します。
TYPE	ストリング	タスク・テンプレートのカテゴリー化に使用されるタイプ。

TASK_TEMPL_CPROP ビュー:

この定義済みデータベース・ビューは、タスク・テンプレートのカスタム・プロパティーを照会するために使用します。

表 36. TASK_TEMPL_CPROP ビュー内の列

列名	タイプ	コメント
TKTID	ストリング	タスク・テンプレート ID。
NAME	ストリング	プロパティーの名前。
DATA_TYPE	ストリング	非ストリング・カスタム・プロパティーのクラスのタイプ。
STRING_VALUE	ストリング	String 型のカスタム・プロパティーの値。

TASK_TEMPL_DESC ビュー:

この定義済みデータベース・ビューは、タスク・テンプレート・オブジェクトのマルチリンガル記述データを照会するために使用します。

表 37. TASK_TEMPL_DESC ビュー内の列

列名	タイプ	コメント
TKTID	ストリング	タスク・テンプレート ID。
LOCALE	ストリング	説明または表示名に関連付けられているロケールの名前。
DESCRIPTION	ストリング	タスク・テンプレートの説明。
DISPLAY_NAME	ストリング	タスク・テンプレートの記述名。

WORK_ITEM ビュー:

この定義済みデータベース・ビューは、作業項目の照会や、プロセス、タスクおよびエスカレーション用の許可データの照会に使用します。

表 38. 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 38. *WORK_ITEM* ビュー内の列 (続き)

列名	タイプ	コメント
OBJECT_TYPE	整数	関連オブジェクトのタイプ。指定可能な値は、以下のとおりです。 **OBJECT_TYPE_ACTIVITY (1)** その作業項目がアクティビティー用に作成されたものであることを明示します。 **OBJECT_TYPE_PROCESS_INSTANCE (3)** その作業項目がプロセス・インスタンス用に作成されたものであることを明示します。 **OBJECT_TYPE_TASK_INSTANCE (5)** その作業項目がタスク用に作成されたものであることを明示します。 **OBJECT_TYPE_TASK_TEMPLATE (6)** その作業項目がタスク・テンプレート用に作成されたものであることを明示します。 **OBJECT_TYPE_ESCALATION_ INSTANCE (7)** その作業項目がエスカレーション・インスタンス用に作成されたものであることを明示します。 **OBJECT_TYPE_APPLICATION_ COMPONENT (9)** その作業項目がアプリケーション・コンポーネント用に作成されたものであることを示します。
OBJECT_ID	ID	関連オブジェクト (例えば、関連プロセスやタスクなど) の ID。
ASSOC_OBJECT_TYPE	整数	ASSOC_OID 属性によって参照されるオブジェクトのタイプ。例えば、タスク、プロセス、または外部オブジェクトなど。OBJECT_TYPE 属性の値を使用します。
ASSOC_OID	ID	作業項目のあるオブジェクト関連オブジェクトの ID。例えば、この作業項目が作成されたアクティビティー・インスタンスを含んでいるプロセス・インスタンスの、プロセス・インスタンス ID など。

表 38. WORK_ITEM ビュー内の列 (続き)

列名	タイプ	コメント
REASON	整数	作業項目の割り当て理由。指定可能な値は、以下のとおりです。 REASON_POTENTIAL_STARTER (5) REASON_POTENTIAL_INSTANCE_ CREATOR (11) REASON_POTENTIAL_STARTER (1) REASON_EDITOR (2) REASON_READER (3) REASON_ORIGINATOR (9) REASON_OWNER (4) REASON_STARTER (6) REASON_ESCALATION_RECEIVER (10) REASON_ADMINISTRATOR (7)
CREATION_TIME	タイム・スタンプ	作業項目が作成された日時。

照会で変数を使用することによるデータのフィルタリング

照会結果は、照会基準に一致するオブジェクトを戻します。この結果を、変数の値でフィルタリングすることもできます。

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。

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。

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));
   ```

このアクションにより、基準を満たすオブジェクトが戻されます。この場合は、A で始まる顧客オーダー。

3. 使用可能な共通保管照会文の名前をリストします。

   以下のコードの断片では、戻される照会のリストを共通照会のみに限定する方法を示しています。

   ```
   String[] storedQuery = process.getStoredQueryNames(StoredQueryData.KIND_PUBLIC);
   ```

4. オプション: 特定の保管照会文で定義された照会を検査します。

   専用の保管照会文には、共通の保管照会文と同じ名前を付けることができます。名前が同じである場合は、専用の保管照会文が戻されます。以下のコードの断片では、指定した名前の共通照会のみを戻す方法を示しています。タスク・ベース・オブジェクトでこの照会を実行する場合は、戻されるオブジェクト・タイプとして、StoredQueryData ではなく StoredQuery を指定してください。

   ```
   StoredQueryData storedQuery = process.getStoredQuery
      (StoredQueryData.KIND_PUBLIC, "CustomerOrdersStartingWithA");
   String selectClause = storedQuery.getSelectClause();
   String whereClause = storedQuery.getWhereClause();
   String orderByClause = storedQuery.getOrderByClause();
   Integer threshold = storedQuery.getThreshold();String owner = storedQuery.getOwner();
   ```

5. 共通の保管照会文を削除します。

   以下のコードの断片では、ステップ 1 で作成した保管照会文の削除方法を示しています。

   ```
   process.deleteStoredQuery("CustomerOrdersStartingWithA");
   ```

他のユーザーの専用保管照会文の管理

専用照会はどのユーザーでも作成できます。この照会は、照会の所有者とシステム管理者しか使用できません。

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。

   ```
   StoredQuery storedQuery = process.getStoredQuery
      ("Smith", "CustomerOrdersStartingWithA");
   String selectClause = storedQuery.getSelectClause();
   String whereClause = storedQuery.getWhereClause();
   String orderByClause = storedQuery.getOrderByClause();
   Integer threshold = storedQuery.getThreshold();String owner = storedQuery.getOwner();
   ```

5. 専用の保管照会文を削除します。

   以下のコードの断片では、ユーザー Smith が所有する専用照会を削除する方法を示しています。

   ```
   process.deleteStoredQuery("Smith", "CustomerOrdersStartingWithA");
   ```

専用保管照会文の操作

システム管理者でなくても、自分専用の保管照会文は作成、実行、および削除できます。また、システム管理者が作成した共通の保管照会文を使用することもできます。

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。

   ```
   StoredQuery storedQuery = process.getStoredQuery
      ("CustomerOrdersStartingWithA");
   String selectClause = storedQuery.getSelectClause();
   String whereClause = storedQuery.getWhereClause();
   String orderByClause = storedQuery.getOrderByClause();
   Integer threshold = storedQuery.getThreshold();String owner = storedQuery.getOwner();
   ```

5. 専用の保管照会文を削除します。

   以下のコードの断片は、専用の保管照会文の削除方法を示します。

   ```
   process.deleteStoredQuery("CustomerOrdersStartingWithA");
   ```

ビジネス・プロセス用のアプリケーションの開発

ビジネス・プロセスは、ビジネス・ゴールを達成するために特定のシーケンスで呼び出される、ビジネス関連の一連のアクティビティーです。プロセスに対する標準のアクションに対応したアプリケーションを開発する方法を示した例が提供されています。

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		
	リーダー	スターター	管理者
createMessage	x	x	x
createWorkItem			x
delete			x
deleteWorkItem			x
forceTerminate			x
getActiveEventHandlers	x		x
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

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				
	リーダー	エディター	潜在的な所有者	所有者	管理者
cancelClaim				x	x
claim			x		x
complete				x	x
createMessage	x	x	x	x	x
createWorkItem					x
deleteWorkItem					x
forceComplete					x
forceRetry					x
getActivityInstance	x	x	x	x	x
getAllWorkItems	x	x	x	x	x
getClientUISettings	x	x	x	x	x
getCustomProperties	x	x	x	x	x
getCustomProperty	x	x	x	x	x
getCustomPropertyNames	x	x	x	x	x
getFaultMessage	x	x	x	x	x
getFaultNames	x	x	x	x	x
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

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/応答操作がインプリメントされている場合、つまり、プロセスに応答が入っている場合、call メソッドを使用してそのプロセスを実行し、その呼び出しでパラメーターとしてプロセス・テンプレート名を渡すことができます。

microflow が片方向操作である場合は、sendMessage メソッドを使用してプロセスを実行します。このメソッドは、次の例には含まれていません。

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/応答操作がインプリメントされている場合、つまり、プロセスに応答が入っている場合、call メソッドを使用してそのプロセスを実行し、その呼び出しで開始サービスの ID を渡すことができます。

microflow が片方向操作である場合は、sendMessage メソッドを使用してプロセスを実行します。このメソッドは、次の例には含まれていません。

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.EXCECUTION_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、リーダー、およびエディターが決定され、プロセス・インスタンスの作業項目を受信します。追加のアクティビティー・インスタンスが決定されます。これらは自動的に開始されるか、または 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.EXCECUTION_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、リーダー、およびエディターが決定され、プロセス・インスタンスの作業項目を受信します。追加のアクティビティー・インスタンスが決定されます。これらは自動的に開始されるか、または 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

呼び出し元は、プロセス・インスタンスの管理者、またはビジネス・プロセス管理者でなければなりません。プロセス・インスタンスを中断するには、プロセス・インスタンスが実行状態または失敗状態でなければなりません。

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。

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。 autonomy 属性が child に設定されたサブプロセスも、状態が実行中、失敗、終了、または補正になっていれば中断されます。このプロセス・インスタンスに関連するインライン・タスクも中断されますが、このプロセス・インスタンスに関連するスタンドアロン・タスクは中断されません。

   この状態では、開始されたアクティビティーはまだ完了することはできますが、新規のアクティビティーはアクティブ化されません。例えば、要求済み状態の human task アクティビティーは完了することができます。

3. プロセス・インスタンスを再開します。

   ```
   process.resume( piid );
   ```

   このアクションにより、プロセス・インスタンスとそのサブプロセスが中断前の状態に戻ります。

ビジネス・プロセスの再開

完了、終了、失敗、補正のいずれかの状態にあるプロセス・インスタンスを再開させることができます。

始める前に

呼び出し元は、プロセス・インスタンスの管理者、またはビジネス・プロセス管理者でなければなりません。

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/応答操作をインプリメントする場合、関連する reply アクティビティーがナビゲートされると、応答が再度送信されます。

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

プロセス管理者権限を持つユーザーが、リカバリー不能状態として認識されているトップレベルのプロセス・インスタンスを終了する必要がある場合があります。プロセス・インスタンスは、未解決のサブプロセスやアクティビティーがあってもこれらを待たずに即時に終了するため、プロセス・インスタンスの終了は例外的な場合にのみ行ってください。

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。

human task アクティビティーの処理

ビジネス・プロセス内の human task 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

human task 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。 human task 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。

1 ユーザーのワークフローの処理

ワークフローの中には、1 人のユーザーだけで実行されるものがあります。例えば、オンライン・ブックストアでの本の注文などです。このタイプのワークフローには、並列パスは存在しません。 completeAndClaimSuccessor API は、このタイプのワークフローの処理をサポートします。

このタスクについて

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、human task 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 human task アクティビティーの要求に相当します。このタイプのワークフローは、ページ・フロー とも呼ばれます。ユーザー・インターフェース定義が、ユーザー・インターフェースのダイアログのフローを制御するアクティビティーと関連付けられているためです。

completeAndClaimSuccessor API は human task アクティビティーを完了し、ログオン・ユーザーの同じプロセス・インスタンスで次のアクティビティーを要求します。そして、次に要求したアクティビティーの情報 (処理される入力メッセージなど) を戻します。次のアクティビティーは、完了したアクティビティーと同じトランザクション内で使用可能になるため、プロセス・モデル内のすべての human task 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. アクティビティー・シーケンスで最初のアクティビティーを要求します。

```
//
//Query the list of activities that can be claimed by the logged-on user
//
QueryResultSet result =
     process.query("ACTIVITY.AIID",
                   "PROCESS_INSTANCE.NAME = 'CustomerOrder' AND
                    ACTIVITY.STATE = ACTIVITY.STATE.STATE_READY AND
                    ACTIVITY.KIND = ACTIVITY.KIND.KIND_STAFF AND
                    WORK_ITEM.REASON =
                         WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                    (String)null, (Integer)null, (TimeZone)null);
...
//
//Claim the first activity
//
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ClientObjectWrapper input = process.claim(aiid);
 DataObject activityInput = null ;
  if ( input.getObject()!= null && input.getObject() instanceof DataObject )
  {
    activityInput = (DataObject)input.getObject();
    // read the values
    ...
  }
}
```

   アクティビティーが要求されると、アクティビティーの入力メッセージが戻されます。

2. アクティビティーの作業が終了したら、そのアクティビティーを完了して次のアクティビティーを要求します。

   アクティビティーを完了するには、出力メッセージを渡します。出力メッセージを作成する場合、メッセージ・タイプ名を指定して、メッセージ定義が含まれるようにする必要があります。

```
ActivityInstanceData activity = process.getActivityInstance(aiid);
ClientObjectWrapper output =
      process.createMessage(aiid, activity.getOutputMessageTypeName());
DataObject myMessage = null ;
if ( output.getObject()!= null && output.getObject() instanceof DataObject )
{
  myMessage = (DataObject)output.getObject();
  //set the parts in your message, for example, an order number
```

```
  myMessage.setInt("OrderNo", 4711);
}

//complete the activity and claim the next one
CompleteAndClaimSuccessorResult successor =
   process.completeAndClaimSuccessor(aiid, output);
```

このアクションは、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 human task 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。

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

このタスクについて

長期実行プロセスのアクティビティーで、障害が発生することがあります。これらの障害が、囲んでいるスコープ内で障害ハンドラーによって catch されておらず、関連したアクティビティー・テンプレートが、エラー発生時にアクティビティーが停止するように指定している場合、アクティビティーは修復することができるように停止状態になります。この状態で、アクティビティーの完了を強制することができます。

例えば、アクティビティーが応答しない場合、実行状態のアクティビティーを強制的に完了することもできます。

特定のタイプのアクティビティーでは、追加要件が存在します。

human task 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 human task アクティビティーなどのアクティビティーを完了します。

   この例では、出力メッセージが渡されます。

```
if (result.size() > 0)
{
   result.first();
   AIID aiid = (AIID) result.getOID(1);
   ActivityInstanceData activity = process.getActivityInstance(aiid);
  ClientObjectWrapper output =
        process.createMessage(aiid, activity.getOutputMessageTypeName());
  DataObject myMessage = null;
  if ( output.getObject()!= null && output.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)output.getObject();
     //set the parts in your message, for example, an order number
     myMessage.setInt("OrderNo", 4711);
   }

   boolean continueOnError = true;
   process.forceComplete(aiid, output, continueOnError);
}
```

   このアクションによって、アクティビティーが完了します。エラーが発生した場合、**continueOnError** パラメーターが、forceComplete 要求によって障害が発生する場合に取るべきアクションを決定します。

   例では、**continueOnError** が true です。この値は、障害が発生した場合、アクティビティーは失敗状態になることを意味します。障害は、処理されるかプロセス・スコープに到達するまで、アクティビティーの囲んでいるスコープに伝搬されます。次にプロセスは障害状態になり、最終的に失敗状態になります。

停止されたアクティビティーの再試行

このタスクについて

長期実行プロセスのアクティビティーに、囲んでいるスコープ内で catch されていない障害が発生した場合、関連したアクティビティー・テンプレートが、エラー発生時にアクティビティーの停止を指定しているときは、アクティビティーは停止状態になり、修復することができます。アクティビティーの実行を再試行することができます。

アクティビティーが使用する変数を設定することができます。また、script アクティビティーの例外と共に、アクティビティーが予想したメッセージなど、強制再試行呼び出しのパラメーターを渡すこともできます。

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 human task 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。

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 39. 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 40. 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。
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。

表 40. 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
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 41. プロセス・インスタンスのライフ・サイクルを制御するための API メソッド

メソッド	説明
suspend	実行状態または失敗状態にある、長期実行中のトップレベルのプロセス・インスタンスの実行を中断します。
resume	中断状態にある、長期実行中のトップレベルのプロセス・インスタンスの実行を再開します。
restart	完了、失敗、または終了状態にある、長期実行中のトップレベルのプロセス・インスタンスを再始動します。
forceTerminate	指定されたトップレベルのプロセス・インスタンスと、子 autonomy を含むそのサブプロセス、およびその実行中のアクティビティー、要求済みのアクティビティー、または待機中のアクティビティーを終了します。
delete	指定されたトップレベルのプロセス・インスタンスと、子 autonomy を含むそのサブプロセスを削除します。
query	データベースから、検索基準に一致するプロパティーを取得します。

アクティビティー

invoke アクティビティーの場合、プロセス・モデルで、それらのアクティビティーがエラー状態でも続行されるように指定できます。continueOnError フラグを false に設定し、未処理エラーが発生すると、そのアクティビティーは停止状態になります。その場合は、プロセス管理者が、そのアクティビティーを修復することができます。continueOnError フラグおよびそれに関連する修復機能は、例えば、invoke アクティビティーが失敗することがある長期実行プロセスなどで使用することができますが、補正および障害処理のモデル化には、かなりの労力が必要です。

アクティビティーの操作および修復には、以下のメソッドが使用可能です。

表 42. 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。

表 42. 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
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。
query	データベースから、検索基準に一致するプロパティーを取得します。

変数およびカスタム・プロパティー

このインターフェースは、変数の値を取得および設定するための get および set メソッドを提供します。指定されたプロパティーをプロセス・インスタンスおよびアクティビティー・インスタンスに関連付けたり、指定されたプロパティーをプロセス・インスタンスおよびアクティビティー・インスタンスから取得することもできます。カスタム・プロパティーの名前および値は、java.lang.String 型である必要があります。

表 43. 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。

表 43. 変数およびカスタム・プロパティーの API メソッド (続き)

メソッド	説明
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・タスクの名前を探します。

   タスクの名前が既に分かっている場合は、このステップはオプションです。

```
TaskTemplate[] taskTemplates = task.queryTaskTemplates
("TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_HUMAN",
 "TASK_TEMPL.NAME",
  new Integer(50),
  (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 human task 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。

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 human task 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。

- 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、リーダーのロールがあるユーザー Smith の作業項目が削除されます。

- 作業項目を転送します。

```
// query the task that is to be rescheduled
QueryResultSet result =
     task.query("DISTINCT TASK.TKIID",
                "TASK.NAME='CustomerOrder' AND
                 TASK.STATE=TASK.STATE.STATE_READY AND
                 WORK_ITEM.REASON=WORK_ITEM.REASON.REASON_POTENTIAL_OWNER AND
                 WORK_ITEM.OWNER_ID='Miller'",
                 (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。

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。

関連概念

43 ページの『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。

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 );
     ```

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" );
     ```

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" );
     ```

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" );
     ```

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 44. 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。

表 44. 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。
createTask	指定されたタスク・テンプレートからタスク・インスタンスを作成します。
createInputMessage	指定されたタスク・テンプレート用の入力メッセージを作成します。例えば、タスクの開始に使用できるメッセージを作成します。
queryTaskTemplates	データベースに保管されているタスク・テンプレートを取得します。

タスク・インスタンス

タスク・インスタンスを扱う作業には、以下のメソッドが使用可能です。

表 45. 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。
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。
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。
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 46. エスカレーションで使用できる API メソッド

メソッド	説明
getEscalation	指定されたエスカレーション・インスタンスを取得します。

カスタム・プロパティー

タスク、タスク・テンプレート、およびエスカレーションはすべてカスタム・プロパティーを持つことができます。このインターフェースは、カスタム・プロパティーの値を取得および設定するための get および set メソッドを提供します。指定されたプロパティーをプロセス・インスタンスおよびアクティビティー・インスタンスに関連付けたり、指定されたプロパティーをタスク・インスタンスから取得することもできます。カスタム・プロパティーの名前および値は、java.lang.String 型である必要があります。次のメソッドは、タスク、タスク・テンプレート、およびエスカレーションで有効です。

表 47. 変数およびカスタム・プロパティーの API メソッド

メソッド	説明
getCustomProperty	指定されたタスク・インスタンスの指定された 1 つのカスタム・プロパティーを取得します。
getCustomProperties	指定されたタスク・インスタンスのカスタム・プロパティー (複数) を取得します。
getCustomPropertyNames	タスク・インスタンスのすべてのカスタム・プロパティーの名前を取得します。
setCustomProperty	指定されたタスク・インスタンスのカスタム固有値を保管します。

各タスクで許可されるアクション

タスクに対して実行可能なアクションは、予定タスク、コラボレーション・タスク、呼び出しタスク、または管理タスクのいずれであるかによって異なります。

HumanTaskManager インターフェースで提供されるすべてのアクションが、すべての種類のタスクに対して使用できるわけではありません。次の表に、タスクの種類ごとに実行可能なアクションを示します。

	タスクの種類			
アクション	予定タスク	コラボレーション・タスク	呼び出しタスク	管理タスク
callTask			X	
cancelClaim	X	X[1]		
claim	X	X[1]		
complete	X	X[1]		X

アクション	タスクの種類			
	予定タスク	コラボレーション・タスク	呼び出しタスク	管理タスク
completeWithFollowOnTask[4]	X	X[1]		
completeWithFollowOnTask[5]		X[3]	X[3]	
createFaultMessage	X	X	X	X
createInputMessage	X	X	X	X
createOutputMessage	X	X	X	X
createWorkItem	X	X[1]	X	X
delete	X[1]	X[1]	X	X[1]
deleteWorkItem	X	X[1]	X	X
getCustomProperty	X	X[1]	X	X
getDocumentation	X	X[1]	X	X
getFaultNames	X	X[1]		
getFaultMessage	X	X[1]	X	
getInputMessage	X	X[1]	X	
getOutputMessage	X	X[1]	X	
getUsersInRole	X	X[1]	X	X
getTask	X	X[1]	X	X
getUISettings	X	X[1]	X	X
resume	X	X[1]		
setCustomProperty	X	X[1]	X	X
setFaultMessage	X	X[1]		
setOutputMessage	X	X[1]		
startTask	X[1]	X[1]	X	X
startTaskAsSubtask[6]	X	X[1]		
startTaskAsSubtask[7]		X[3]	X[3]	
suspend	X	X[1]		
suspendWithCancelClaim	X	X[1]		
terminate	X[1]	X[1]	X[1]	
transferWorkItem	X	X[1]	X	X
update	X	X[1]	X	X

注:

1. スタンドアロンのタスク、臨時のタスク、およびタスク・テンプレートの場合のみ
2. スタンドアロンのタスク、ビジネス・プロセス内のインライン・タスク、および臨時のタスクの場合のみ
3. スタンドアロンのタスクおよび臨時のタスクの場合のみ
4. 後続タスクを持つことのできるタスクの種類
5. 後続タスクとして使用できるタスクの種類
6. サブタスクを持つことのできるタスクの種類
7. サブタスクとして使用できるタスクの種類

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 human task 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 human task 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 human task 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 human task 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。
- human task 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。 human task 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。言い換えると、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。テンプレートの完全なリストを入手するには、プロセス・テンプレートと関連する管理タスク・テンプレートを読み取り、ログオン・ユーザーが管理者である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、一連の human task アクティビティー (予定タスク) として、書籍を注文するための一連のアクションを実装する方法を示しています。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

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、human task 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 human task 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 human task アクティビティーの最初の human task 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 human task 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 human task 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 human task アクティビティーを要求します。

   human task 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 human task 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 human task 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 human task 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、human task アクティビティーを完了し、次の human task アクティビティーを取得します。

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。

API 例外の処理

このタスクについて

BusinessFlowManagerService インターフェースまたは HumanTaskManagerService インターフェースのメソッドが正常に完了しない場合、エラーの原因を示す例外がスローされます。この例外を特別に処理して、呼び出し元にガイダンスを提供することができます。

ただし、例外のサブセットのみを特別に処理し、その他の潜在的な例外に対しては汎用のガイダンスを提供するのが一般的です。すべての固有の例外は、汎用の ProcessException または TaskException から継承しています。最終の `catch(ProcessException)` または `catch(TaskException)` ステートメントを使用して汎用の例外を catch することが*最良実例* です。このステートメントでは、発生する可能性があるその他すべての例外を考慮に入れるため、このステートメントによって、ご使用のアプリケーション・プログラムの上位互換性を確保することができます。

アクティビティーに設定された障害の検査

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

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 human task 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();
  }
}
```

第 12 章 Web サービス API クライアント・アプリケーションの開発

Web サービス API を介してビジネス・プロセス・アプリケーションとヒューマン・タスク・アプリケーションにアクセスするクライアント・アプリケーションを開発できます。

このタスクについて

クライアント・アプリケーションは、任意の Web サービス・クライアント環境 (Java Web サービス、Microsoft .NET など) で開発できます。

概要: Web サービス

Web サービスは、クライアント・アプリケーションとデータを交換するために、オープンな XML ベース標準およびトランスポート・プロトコルを使用する Web ベースのエンタープライズ・アプリケーションです。 Web サービスにより、言語および環境に中立なプログラミング・モデルを使用することができます。

Web サービスは次のコア・テクノロジーを使用します。

- XML (Extensible Markup Language)。 XML は、データ独立性の問題を解決します。これを使用してデータを記述し、さらにアプリケーションまたはプログラミング言語との間でデータをマップします。
- WSDL (Web サービス記述言語)。この XML ベースの言語を使用して、基礎となるアプリケーションの記述を作成します。この記述は、基礎となるアプリケーションとその他の Web 対応アプリケーションとの間のインターフェースとして機能することにより、アプリケーションを Web サービスに変換します。
- SOAP (Simple Object Access Protocol)。SOAP は、Web 用のコアとなる通信プロトコルで、ほとんどの Web サービスはこのプロトコルを使用して相互に対話します。

Web サービス・コンポーネントおよび一連の制御

多くのクライアント・サイドおよびサーバー・サイドのコンポーネントは、Web サービスの要求と応答を表す一連の制御に関与します。

標準的な一連の制御は以下のとおりです。

1. クライアント・サイド:
   a. クライアント・アプリケーション (ユーザーによって提供される) は、Web サービスの要求を発行します。
   b. プロキシー・クライアント (ユーザーによっても提供されるが、クライアント・サイド・ユーティリティーを使用して自動的に生成することが可能) は、サービス要求を SOAP 要求エンベロープでラップします。

c. クライアント・サイド開発インフラストラクチャーは、Web サービスのエンドポイントとして定義された URL に要求を転送します。

2. ネットワークは、HTTP または HTTPS を使用して Web サービス・エンドポイントに要求を送信します。
3. サーバー・サイド:
   a. 汎用 Web サービス API は、要求を受信し、デコードします。
   b. 要求は、Business Flow Manager または Human Task Manager の汎用コンポーネントによって直接処理されるか、指定されたビジネス・プロセスまたはヒューマン・タスクに転送されます。
   c. 戻されたデータは SOAP 応答エンベロープでラップされます。
4. ネットワークは、HTTP または HTTPS を使用してクライアント・サイド環境に応答を送信します。
5. クライアント・サイドに戻る:
   a. クライアント・サイド開発インフラストラクチャーは、SOAP 応答エンベロープをアンラップします。
   b. プロキシー・クライアントはデータを SOAP 応答から抽出して、それをクライアント・アプリケーションに受け渡します。
   c. クライアント・アプリケーションは、必要に応じて、戻されたデータを処理します。

Web サービス API の概要

Web サービス API により、Business Process Choreographer 環境で実行しているビジネス・プロセスおよびヒューマン・タスクに Web サービスを使用してアクセスするクライアント・アプリケーションを開発できます。

Business Process Choreographer Web サービス API には、次の 2 つの別個の Web サービス・インターフェース (WSDL ポート・タイプ) があります。

- Business Flow Manager API。クライアント・アプリケーションが Microflow および長期間のプロセスと対話できるようにします。例えば、以下の操作を実行できます。
  - プロセス・テンプレートとプロセス・インスタンスの作成
  - 既存のプロセスの要求
  - ID によるプロセスの照会

  実行可能なアクションの詳細なリストについては、 493 ページの『ビジネス・プロセス用のアプリケーションの開発』を参照してください。
- Human Task Manager API。クライアント・アプリケーションは以下の操作を実行できます。
  - タスクの作成と開始
  - 既存のタスクの要求
  - タスクの完了
  - ID によるタスクの照会
  - タスクの集合の照会

実行可能なアクションの詳細なリストについては、 516 ページの『ヒューマン・タスク用のアプリケーションの開発』を参照してください。

クライアント・アプリケーションは、Web サービス・インターフェースのいずれかまたは両方を使用できます。

例

Human Task Manager Web サービス API にアクセスして、関係するヒューマン・タスクを処理するクライアント・アプリケーションの概要としては以下のようなものが考えられます。

1. クライアント・アプリケーションは、query Web サービス呼び出しを WebSphere Process Server に対して発行します。これにより、ユーザーによって処理される参加タスクのリストを要求します。
2. 参加タスクのリストが、SOAP/HTTP 応答エンベロープで戻されます。
3. クライアント・アプリケーションは、claim Web サービス呼び出しを発行して、いずれかの参加タスクを要求します。
4. WebSphere Process Server は、タスクの入力メッセージを戻します。
5. クライアント・アプリケーションは、complete Web サービス呼び出しを発行して、出力メッセージまたは障害メッセージのあるタスクを完了します。

ビジネス・プロセスおよびヒューマン・タスクの要件

Business Process Choreographer 上で実行するように WebSphere Integration Developer で開発されたビジネス・プロセスとヒューマン・タスクが、Web サービス API によってアクセス可能となるためには、特定の規則に準拠する必要があります。

要件は以下のとおりです。

1. ビジネス・プロセスとヒューマン・タスクのインターフェースは、Java API for XML-based RPC (JAX-RPC 1.1) 仕様で定義された「document/literal wrapped」スタイルを使用して定義する必要があります。これは、WID で開発するすべてのビジネス・プロセスとヒューマン・タスクのデフォルト・スタイルです。
2. Web サービス・オペレーションのビジネス・プロセスとヒューマン・タスクによって出される障害メッセージは、XML スキーマ・エレメントで定義された単一の WSDL メッセージ部分で構成されている必要があります。以下に例を示します。

   ```
   <wsdl:part name="myFault" element="myNamespace:myFaultElement"/>
   ```

   関連情報

   Java API for XML based RPC (JAX-RPC) のダウンロード・ページ

   使用すべき WSDL のスタイル

クライアント・アプリケーションの開発

クライアント・アプリケーションの開発プロセスは多数のステップで構成されています。

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。 あるいは、WebSphere Process Server クライアント CD からファイルをコピーすることもできます。
3. 選択したクライアント・アプリケーション開発環境で、エクスポートした成果物を使用してプロキシー・クライアントを生成します。
4. オプション: ヘルパー・クラスを生成します。 ヘルパー・クラスが必要になるのは、クライアント・アプリケーションが、WebSphere サーバー上の具象プロセスまたはタスクと直接対話する場合です。ただし、クライアント・アプリケーションが汎用タスク (照会の発行など) を実行するだけの場合は、ヘルパー・クラスは不要です。
5. クライアント・アプリケーション用のコードを開発します。
6. 必要なセキュリティー・メカニズムをクライアント・アプリケーションに追加します。

成果物のコピー

クライアント・アプリケーションの作成を補助するために、いくつかの成果物を WebSphere 環境からコピーする必要があります。

成果物を取得するには、以下の 2 つの方法があります。

- WebSphere Process Server 環境から成果物を公開およびエクスポートします。
- WebSphere Process Server クライアント CD からファイルをコピーします。

サーバー環境からの成果物の公開およびエクスポート

クライアント・アプリケーションを開発して Web サービス API にアクセスする前に、WebSphere サーバー環境から、いくつかの成果物を公開およびエクスポートする必要があります。

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。

- Web サービス API を構成するポート・タイプおよび操作を記述する Web サービス記述言語 (WSDL) ファイル。
- WSDL ファイルのサービスおよびメソッドによって参照されるデータ型の定義を含む XML スキーマ定義 (XSD)。
- ビジネス・オブジェクトを記述するその他の WSDL および XSD ファイル。ビジネス・オブジェクトは、WebSphere サーバーで実行する具象ビジネス・プロセスまたはヒューマン・タスクを記述します。このその他のファイルは、クライア

ント・アプリケーションが Web サービス API を介して具象ビジネス・プロセスまたはヒューマン・タスクに直接対話する必要がある場合のみ必要です。クライアント・アプリケーションが汎用タスク (照会の発行など) を実行するだけの場合は、これらは不要です。

これらの成果物は、公開された後、ご使用のクライアント・プログラミング環境にコピーする必要があります。それらは、プロキシー・クライアントおよびヘルパー・クラスを生成するために使用されます。

関連タスク

555 ページの『クライアント CD からファイルのコピー』
Web サービス API にアクセスするのに必要なファイルは、WebSphere Process Server クライアント CD で入手できます。

Web サービス・エンドポイント・アドレスの指定

Web サービス・エンドポイント・アドレスは、クライアント・アプリケーションが Web サービス API にアクセスするために指定する必要のある URL です。エンドポイント・アドレスは、クライアント・アプリケーション用のプロキシー・クライアントを生成するためにエクスポートする WSDL ファイルに書き込まれます。

このタスクについて

使用する Web サービス・エンドポイント・アドレスは、WebSphere サーバーの構成によって異なります。

- シナリオ 1。WebSphere サーバーが 1 台だけの場合。指定する WebSphere エンドポイント・アドレスは、サーバーのホスト名とポート番号です (例: **host1:9080**)。
- シナリオ 2。複数のサーバーで構成される WebSphere クラスターの場合。指定する WebSphere エンドポイント・アドレスは、Web サービス API をホスティングするサーバーのホスト名とポートです (例: **host2:9081**)。
- シナリオ 3。Web サーバーをフロントエンドとして使用する場合。指定する WebSphere エンドポイント・アドレスは、Web サーバーのホスト名とポートです (例: **host:80**)。

デフォルトでは、Web サービス・エンドポイント・アドレスの形式は *protocol*://*host*:*port*/*context_root*/*fixed_path* です。各部の意味は、次のとおりです。

- *protocol*。クライアント・アプリケーションと WebSphere サーバー間で使用される通信プロトコル。デフォルト・プロトコルは HTTP です。代わりに、もっと安全な HTTPS (HTTP over SSL) プロトコルを使用することもできます。HTTPS を使用することをお勧めします。
- *host:port*。Web サービス API をホスティングするマシンへのアクセスに使用するホスト名とポート番号。この値は、クライアント・アプリケーションがアプリケーションに直接アクセスするか、Web サーバー・フロントエンド経由でアクセスするかなど、WebSphere サーバー構成によって異なります。
- *context_root*。コンテキスト・ルートには、任意の値を選択できます。ただし、選択する値は個々の WebSphere セル内で固有でなければなりません。デフォルト値は、「node_server/cluster」サフィックスを使用して、ネーミング競合のリスクを回避しています。

- *fixed_path* は、/sca/com/ibm/bpe/api/BFMWS (Business Flow Manager API の場合) または /sca/com/ibm/task/api/HTMWS (Human Task Manager API の場合) のどちらかで、変更することはできません。

Web サービス・エンドポイント・アドレスは、最初は、ビジネス・プロセス・コンテナーまたはヒューマン・タスク・コンテナーの構成時に指定されます。

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**」を選択します。

   注: 「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」を選択して、使用可能なすべてのエンタープライズ・アプリケーションのリストを表示することもできます。
3. SCA モジュールまたはアプリケーションのリストから、「**BPEContainer**」(ビジネス・プロセス・コンテナーの場合) または「**TaskContainer**」(ヒューマン・タスク・コンテナーの場合) を選択します。
4. 「**追加プロパティー**」リストから、「**HTTP エンドポイント URL 情報を提供 (Provide HTTP endpoint URL information)**」を選択します。
5. リストからデフォルトのプレフィックスのいずれか 1 つを選択するか、カスタム・プレフィックスを入力します。 クライアント・アプリケーションを、Web サービス API をホスティングするアプリケーション・サーバーに直接接続する場合は、デフォルト・プレフィックス・リストのプレフィックスを使用します。そうでない場合は、カスタム・プレフィックスを指定します。
6. 「**適用**」をクリックして、選択したプレフィックスを SCA モジュールにコピーします。
7. 「**OK**」をクリックします。 URL 情報がワークスペースに保管されます。

結果

管理コンソールで現在の値を表示できます (例えばビジネス・プロセス・コンテナーの場合は、「**エンタープライズ・アプリケーション**」 → 「**BPEContainer**」 → 「**デプロイメント記述子の表示**」)。

エクスポートされた WSDL ファイルでは、soap:address エレメントの location 属性に、指定した Web サービス・エンドポイント・アドレスが含まれています。以下に例を示します。

```
<wsdl:service name="BFMWSService">
  <wsdl:port name="BFMWSPort" binding="this:BFMWSBinding">
   <soap:address location=
         "https://myserver:9080/WebServicesAPIs/sca/com/ibm/bpe/api/BFMWS"/>
```

関連概念

565 ページの『セキュリティーの追加 (Java Web サービス)』
クライアント・アプリケーションにセキュリティー・メカニズムを実装することにより、 Web サービス通信を保護する必要があります。

関連タスク

575 ページの『セキュリティーの追加 (.NET)』
Web サービス通信は、クライアント・アプリケーションにセキュリティー・メカニズムを組み込むことにより保護できます。

『WSDL ファイルの公開』
Web サービス記述言語 (WSDL) ファイルには、Web サービス API で使用できるすべての操作の詳細な説明が含まれています。Business Flow Manager Web サービス API と Human Task Manager Web サービス API では、使用できる WSDL ファイルが異なります。これらの WSDL ファイルは、まず公開して、その後 WebSphere 環境からご使用の開発環境にコピーし、プロキシー・クライアントの生成に使用することになります。

WSDL ファイルの公開

Web サービス記述言語 (WSDL) ファイルには、Web サービス API で使用できるすべての操作の詳細な説明が含まれています。Business Flow Manager Web サービス API と Human Task Manager Web サービス API では、使用できる WSDL ファイルが異なります。これらの WSDL ファイルは、まず公開して、その後 WebSphere 環境からご使用の開発環境にコピーし、プロキシー・クライアントの生成に使用することになります。

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

WSDL ファイルの公開が必要なのは一度だけです。

注: WebSphere Process Server クライアント CD を所有している場合は、代わりに、その CD からクライアント・プログラミング環境に直接ファイルをコピーすることもできます。

関連タスク

570 ページの『プロキシー・クライアントの生成 (.NET)』
.NET クライアント・アプリケーションは、プロキシー・クライアント を使用して Web サービス API と対話します。プロキシー・クライアントのおかげで、クライアント・アプリケーションは、複雑な Web サービス・メッセージング・プロトコルを意識する必要がなくなります。

549 ページの『Web サービス・エンドポイント・アドレスの指定』
Web サービス・エンドポイント・アドレスは、クライアント・アプリケーションが Web サービス API にアクセスするために指定する必要のある URL です。エンドポイント・アドレスは、クライアント・アプリケーション用のプロキシー・クライアントを生成するためにエクスポートする WSDL ファイルに書き込まれます。

ビジネス・プロセス WSDL の公開:

管理コンソールを使用して、WSDL ファイルを公開します。

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**」を選択します。

   注: 「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」を選択して、使用可能なすべてのエンタープライズ・アプリケーションのリストを表示することもできます。
3. SCA モジュールまたはアプリケーションのリストから **BPEContainer** アプリケーションを選択します。
4. 「**追加プロパティー**」のリストから「**WSDL ファイルの公開 (Publish WSDL files)**」を選択します。
5. リスト内の zip ファイルをクリックします。
6. 表示された「ファイルのダウンロード (File Download)」ウィンドウで、「**保管**」をクリックします。
7. ローカル・フォルダーを参照し、「**保管**」をクリックします。

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、Web サービスを記述する WSDL ファイルと、その WSDL ファイル内から参照される XSD ファイルが含まれています。

ヒューマン・タスク WSDL の公開:

管理コンソールを使用して、WSDL ファイルを公開します。

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**」を選択します。

   注: 「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」を選択して、使用可能なすべてのエンタープライズ・アプリケーションのリストを表示することもできます。
3. SCA モジュールまたはアプリケーションのリストから **TaskContainer** アプリケーションを選択します。
4. 「**追加プロパティー**」のリストから「**WSDL ファイルの公開 (Publish WSDL files)**」を選択します。
5. リスト内の zip ファイルをクリックします。
6. 表示された「ファイルのダウンロード (File Download)」ウィンドウで、「**保管**」をクリックします。
7. ローカル・フォルダーを参照し、「**保管**」をクリックします。

結果

エクスポートされた zip ファイルの名前は TaskContainer_WSDLFiles.zip です。 zip ファイルには、Web サービスを記述する WSDL ファイルと、その WSDL ファイル内から参照される XSD ファイルが含まれています。

ビジネス・オブジェクトのエクスポート

ビジネス・プロセスおよびヒューマン・タスクには、Web サービスとして外部にアクセスできるよう明確に定義されたインターフェースが用意されています。これらのインターフェースがビジネス・オブジェクトを参照する場合は、インターフェース定義とビジネス・オブジェクトをクライアント・プログラミング環境にエクスポートする必要があります。

このタスクについて

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・オブジェクトごとに、繰り返してください。

WebSphere Process Server では、ビジネス・オブジェクトが、ビジネス・プロセスまたはヒューマン・タスクと対話する要求、応答、および障害メッセージの形式を定義します。これらのメッセージに、複合データ・タイプの定義を含めることもできます。

例えば、ヒューマン・タスクを作成して開始するには、以下の情報項目をタスク・インターフェースに渡す必要があります。

- タスク・テンプレート名
- タスク・テンプレート・ネームスペース
- 入力メッセージ (フォーマット済みのビジネス・データを含む)
- 応答メッセージを戻すための応答ラッパー
- 障害および例外を戻すための障害メッセージ

以上の項目は、単一のビジネス・オブジェクト内でカプセル化されます。Web サービス・インターフェースのすべての操作は、「document/literal wrapped」操作としてモデル化されます。これらの操作の入出力パラメーターは、ラッパー文書でカプセル化されます。他のビジネス・オブジェクトは、それに対応する応答および障害のメッセージ形式を定義します。

Web サービスを介してビジネス・プロセスまたはヒューマン・タスクを作成および開始するためには、これらのラッパー・オブジェクトがクライアント・サイドのクライアント・アプリケーションで使用可能になる必要があります。

そのためには、WebSphere 環境からビジネス・オブジェクトを Web サービス記述言語 (WSDL) ファイルおよび XML スキーマ定義 (XSD) ファイルとしてエクスポートし、データ・タイプ定義をクライアント・プログラミング環境にインポートしてから、それをクライアント・アプリケーションで使用できるヘルパー・クラスに変換します。

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、<xsd:any> パラメーターを使用して Web サービス API に渡したり Web サービス API から受け取ったりする必要があります。これらのユーザー定義エレメントは、エクスポートされた XSD ファイルから生成されるヘルパー・クラスを使用して、クライアント・アプリケーション上で作成、直列化、および非直列化されます。

関連タスク

572 ページの『BPEL プロセス用ヘルパー・クラスの作成 (.NET)』
特定の Web サービス API 操作では、クライアント・アプリケーションが「document/literal」スタイルのラップ・エレメントを使用する必要があります。クライアント・アプリケーションは、ヘルパー・クラスに、必要なラッパー・エレメントの生成を担当させます。

572 ページの『BPEL プロセス用ヘルパー・クラスの作成 (.NET)』
特定の Web サービス API 操作では、クライアント・アプリケーションが「document/literal」スタイルのラップ・エレメントを使用する必要があります。クライアント・アプリケーションは、ヘルパー・クラスに、必要なラッパー・エレメントの生成を担当させます。

562 ページの『BPEL プロセス用ヘルパー・クラスの作成 (Java Web サービス)』
具象 API 要求 (sendMessage、call など) で参照されるビジネス・オブジェクトでは、クライアント・アプリケーションが「document/literal wrapped」スタイ

ル・エレメントを使用する必要があります。クライアント・アプリケーションは、ヘルパー・クラスに、必要なラッパー・エレメントの生成を担当させます。

クライアント CD のファイルの使用

WebSphere サーバー環境からの成果物をエクスポートする代わりに、WebSphere Process Server クライアント CD からクライアント・アプリケーションの生成に必要なファイルをコピーできます。

この場合は、Business Flow Manager API または Human Task Manager API のデフォルト Web サービス・エンドポイント・アドレスを手動で変更する必要があります。

クライアント・アプリケーションが両方の API にアクセスする場合は、両方の API のデフォルト・エンドポイント・アドレスを編集する必要があります。

クライアント CD からファイルのコピー

Web サービス API にアクセスするのに必要なファイルは、WebSphere Process Server クライアント CD で入手できます。

手順

1. クライアント CD にアクセスして、ProcessChoreographer¥client ディレクトリーを参照します。
2. 必要なファイルをクライアント・アプリケーション開発環境にコピーします。

   Business Flow Manager API の場合、以下をコピーします。

   BFMWS.wsdl
   Business Flow Manager Web サービス API で使用できる Web サービスを記述します。このファイルには、エンドポイント・アドレスが含まれています。

   BFMIF.wsdl
   Business Flow Manager Web サービス API の各 Web サービスのパラメーターおよびデータ型を記述します。

   BFMIF.xsd
   Business Flow Manager Web サービス API で使用されるデータ型を記述します。

   BPCGEN.xsd
   Business Flow Manager と Human Task Manager の Web サービス API 間で共通するデータ型を含みます。

   Human Task Manager API の場合は、以下をコピーします。

   HTMWS.wsdl
   Human Task Manager Web サービス API で使用できる Web サービスを記述します。このファイルには、エンドポイント・アドレスが含まれています。

   HTMIF.wsdl
   Human Task Manager Web サービス API の各 Web サービスのパラメーターおよびデータ型を記述します。

HTMIF.xsd

Human Task Manager Web サービス API で使用されるデータ型を記述します。

BPCGEN.xsd

Business Flow Manager と Human Task Manager の Web サービス API 間で共通するデータ型を含みます。

注: BPCGen.xsd ファイルは、両方の API に共通です。

ファイルをコピーした後、BFMWS.wsdl または HTMWS.wsdl ファイルの Web サービス API エンドポイント・アドレスを、Web サービス API をホスティングする WebSphere アプリケーション・サーバーのアドレスに手動で変更する必要があります。

関連タスク

『Web サービス・エンドポイント・アドレスの手動変更』
クライアント CD からファイルをコピーする場合は、WSDL ファイルで指定したデフォルトの Web サービス・エンドポイント・アドレスを、Web サービス API をホスティングするサーバーのアドレスに変更する必要があります。

548 ページの『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 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。

Web サービス・エンドポイント・アドレスの手動変更

クライアント CD からファイルをコピーする場合は、WSDL ファイルで指定したデフォルトの Web サービス・エンドポイント・アドレスを、Web サービス API をホスティングするサーバーのアドレスに変更する必要があります。

このタスクについて

管理コンソールを使用すると、WSDL ファイルをエクスポートする前に Web サービス・エンドポイント・アドレスを設定することができます。ただし、WSDL ファイルを WebSphere Process Server クライアント CD からコピーする場合は、デフォルトの Web サービス・エンドポイント・アドレスを手動で変更する必要があります。

使用する Web サービス・エンドポイント・アドレスは、WebSphere サーバーの構成によって異なります。

- シナリオ 1。WebSphere サーバーが 1 台だけの場合。指定する WebSphere エンドポイント・アドレスは、サーバーのホスト名とポート番号です (例: **host1:9080**)。
- シナリオ 2。複数のサーバーで構成される WebSphere クラスターの場合。指定する WebSphere エンドポイント・アドレスは、Web サービス API をホスティングするサーバーのホスト名とポートです (例: **host2:9081**)。
- シナリオ 3。Web サーバーをフロントエンドとして使用する場合。指定する WebSphere エンドポイント・アドレスは、Web サーバーのホスト名とポートです (例: **host:80**)。

関連タスク

555 ページの『クライアント CD からファイルのコピー』
Web サービス API にアクセスするのに必要なファイルは、WebSphere Process Server クライアント CD で入手できます。

Business Flow Manager API エンドポイントの変更:

WebSphere Process Server クライアント CD から Business Flow Manager API ファイルをコピーする場合は、デフォルトのエンドポイント・アドレスを手動で編集する必要があります。

手順

1. クライアント CD からコピーされたファイルを含むディレクトリーへナビゲートします。
2. テキスト・エディターまたは XML エディターで BFMWS.wsdl ファイルを開きます。
3. `soap:address` エレメントを検索します (ファイルの最後の方にあります)。
4. `location` 属性の値を、Web サービス API が実行しているサーバーの HTTP URL で変更します。 これを行うには、次のようにします。
   a. オプションで、より機密保護機能のある HTTPS プロトコルを使用するために `http` を `https` で置換します。
   b. *localhost* を、Web サービス API サーバー・エンドポイント・アドレスのホスト名または IP アドレスで置換します。
   c. *9080* をアプリケーション・サーバーのポート番号で置換します。
   d. *BPEContainer_N1_server1* を、Web サービス API を実行しているアプリケーションのコンテキスト・ルートで置換します。 デフォルト・コンテキスト・ルートは以下で構成されます。
      - *BPEContainer*。アプリケーション名。
      - *N1*。ノード名。
      - *server1*。サーバー名。
   e. URL の固定部分 (`/sca/com/ibm/bpe/api/BFMWS`) は変更しないでください。

   例えば、アプリケーションがサーバー **s1.n1.ibm.com** で実行していて、サーバーがポート **9080** で SOAP/HTTP 要求を受け入れている場合、`soap:address` エレメントを以下のように変更します。

```
<soap:address location="http://si.n1.ibm.com:9080/
                  BPEContainer_N1_server1/sca/com/ibm/bpe/api/BFMWS"/>
```

関連概念

565 ページの『セキュリティーの追加 (Java Web サービス)』
クライアント・アプリケーションにセキュリティー・メカニズムを実装することにより、 Web サービス通信を保護する必要があります。

関連タスク

575 ページの『セキュリティーの追加 (.NET)』
Web サービス通信は、クライアント・アプリケーションにセキュリティー・メカニズムを組み込むことにより保護できます。

Human Task Manager API エンドポイントの変更:

WebSphere Process Server クライアント CD から Human Task Manager API ファイルをコピーする場合は、デフォルトのエンドポイント・アドレスを手動で編集する必要があります。

手順

1. クライアント CD からコピーされたファイルを含むディレクトリーへナビゲートします。
2. テキスト・エディターまたは XML エディターで HTMWS.wsdl ファイルを開きます。
3. `soap:address` エレメントを検索します (ファイルの最後の方にあります)。
4. `location` 属性の値を、正しいエンドポイント・アドレスで置換します。 これを行うには、次のようにします。
   a. オプションで、より機密保護機能のある HTTPS プロトコルを使用するために `http` を `https` で置換します。
   b. *localhost* を、Web サービス API サーバーのエンドポイント・アドレスのホスト名または IP アドレスで置換します。
   c. *9080* をアプリケーション・サーバーのポート番号で置換します。
   d. *HTMContainer_N1_server1* を、Web サービス API を実行しているアプリケーションのコンテキスト・ルートで置換します。 デフォルト・コンテキスト・ルートは以下で構成されます。
      - *HTMContainer*。アプリケーション名。
      - *N1*。ノード名。
      - *server1*。サーバー名。
   e. URL の固定部分 (`/sca/com/ibm/task/api/HTMWS`) は変更しないでください。

   例えば、アプリケーションがサーバー **s1.n1.ibm.com** で実行していて、サーバーがポート **9081** で SOAP/HTTP 要求を受け入れている場合、`soap:address` を以下のように変更します。

```
<soap:address location="https://si.n1.ibm.com:9081/
                  HTMContainer_N1_server1/sca/com/ibm/task/api/HTMWS"/>
```

関連概念

565 ページの『セキュリティーの追加 (Java Web サービス)』
クライアント・アプリケーションにセキュリティー・メカニズムを実装することにより、 Web サービス通信を保護する必要があります。

関連タスク

575 ページの『セキュリティーの追加 (.NET)』
Web サービス通信は、クライアント・アプリケーションにセキュリティー・メカニズムを組み込むことにより保護できます。

Java Web サービス環境でのクライアント・アプリケーションの開発

Java Web サービスと互換性のある Java ベースの開発環境を使用して、Web サービス API 用のクライアント・アプリケーションを開発できます。

プロキシー・クライアントの生成 (Java Web サービス)

Java Web サービス・クライアント・アプリケーションは、*プロキシー・クライアント* を使用して Web サービス API と対話します。

このタスクについて

Java Web サービスのプロキシー・クライアントには、Web サービス要求を実行するためにクライアント・アプリケーションが呼び出す、いくつかの Java Bean クラスが含まれています。プロキシー・クライアントは、サービス・パラメーターの SOAP メッセージへのアセンブリーを処理し、その SOAP メッセージを HTTP 経由で Web サービスに送信し、Web サービスから応答を受信して、戻されたデータをクライアント・アプリケーションに渡します。

したがって、基本的にプロキシー・クライアントは、クライアント・アプリケーションが Web サービスをローカル機能のように呼び出せるようにするためのものです。

注: プロキシー・クライアントの生成が必要なのは一度だけです。同じ Web サービス API にアクセスするクライアント・アプリケーションはすべて、以後は同じプロキシー・クライアントを使用できます。

IBM Web サービス環境でプロキシー・クライアントを生成するには、以下の 2 つの方法があります。

- Rational® Application Developer または WebSphere Integration Developer の統合開発環境を使用する。
- WSDL2Java コマンド行ツールを使用する。

他の Java Web サービス開発環境には通常、WSDL2Java ツールまたは独自のクライアント・アプリケーション生成機能が含まれます。

Rational Application Developer によるプロキシー・クライアントの生成

Rational Application Developer の統合開発環境では、クライアント・アプリケーション用のプロキシー・クライアントを生成できます。

始める前に

プロキシー・クライアントを生成するには、その前に、ビジネス・プロセスまたはヒューマン・タスクの Web サービス・インターフェースを記述した WSDL ファイルを WebSphere 環境 (または WebSphere Process Server クライアント CD) からエクスポートし、それをクライアントのプログラミング環境にコピーしておく必要があります。

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。

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/b.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。
  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/h.jar からインポートします。

新規ディレクトリー wsdl 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. Web サービス・ウィザード・プロパティーを変更します。
   a. Rational Application Developer で、「**設定**」 → 「**Web サービス**」 → 「**コード生成**」 → 「**IBM Websphere ランタイム (IBM WebSphere runtime)**」を選択します。
   b. 「**ラップ・スタイルを使用せずに WSDL から Java を生成 (Generate Java from WSDL using the no wrapped style)**」オプションを選択します。

   注: 「**設定**」メニューで「**Web サービス**」オプションを選択できない場合は、まず「**ウィンドウ**」 → 「**設定**」 → 「**ワークベンチ**」 → 「**機能**」と進んで、必要な機能を有効にする必要があります。「**Web サービス開発者 (Web Service Developer)**」をクリックして、「**OK**」をクリックします。次いで「設定」ウィンドウを再び開き、「**コード生成**」オプションを変更します。
3. 新たに作成された wsdl ディレクトリー内にある BFMWS.WSDL または HTMWS.WSDL ファイルを選択します。
4. 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。
5. 「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。
6. 「Web サービス選択」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。
7. 「クライアント環境構成」ウィンドウで、以下のようにします。
   a. 「**編集**」をクリックして、「Web サービス・ランタイム」オプションを IBM WebSphere に変更します。
   b. 「J2EE のバージョン」オプションを 1.4 に変更します。
   c. 「**OK**」をクリックします。
   d. 「**次へ**」をクリックします。
8. このステップは、ビジネス・プロセス Web サービス API とヒューマン・タスク Web サービス API の両方を含む Web サービス・クライアントを生成する必要がある場合にのみ必要です。なぜなら、両方の WSDL ファイルに重複するメソッドがあるからです。
   a. 「Web サービス・プロキシー」ウィンドウで、「名前空間からパッケージへのカスタム・マッピングを定義」を選択して、「**OK**」をクリックします。

b. 「Web サービス・クライアントの名前空間からパッケージへのマッピング」ウィンドウで、以下のネーム・スペースおよびパッケージを追加します。

BFMWS.wsdl の場合:

ネーム・スペース	パッケージ
http://www.ibm.com/xmlns/prod/websphere/business-process/types/6.0	com.ibm.sca.bpe
http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0	com.ibm.sca.bpe
http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0/Binding	com.ibm.sca.bpe
http://www.ibm.com/xmlns/prod/websphere/bpc-common/types/6.0	com.ibm.sca.bpe

HTMWS.wsdl の場合:

ネーム・スペース	パッケージ
http://www.ibm.com/xmlns/prod/websphere/human-task/types/6.0	com.ibm.sca.task
http://www.ibm.com/xmlns/prod/websphere/human-task/services/6.0	com.ibm.sca.task
http://www.ibm.com/xmlns/prod/websphere/human-task/services/6.0/Binding	com.ibm.sca.task
http://www.ibm.com/xmlns/prod/websphere/bpc-common/types/6.0	com.ibm.sca.task

上書きを確認するように求められる場合、「**YesToAll**」をクリックします。

9. 「**終了**」をクリックします。

結果

複数のプロキシー、ロケーター、ヘルパー Java クラスで構成されるプロキシー・クライアントが生成され、プロジェクトに追加されます。デプロイメント記述子も更新されます。

WSDL2Java によるプロキシー・クライアントの生成

WSDL2Java は、プロキシー・クライアントを生成するコマンド行ツールです。プロキシー・クライアントによって、クライアント・アプリケーションのプログラミングが容易になります。

始める前に

プロキシー・クライアントを生成するには、その前に、ビジネス・プロセスまたはヒューマン・タスクの Web サービス API を記述した WSDL ファイルを WebSphere 環境 (または WebSphere Process Server クライアント CD) からエクスポートし、それをクライアントのプログラミング環境にコピーしておく必要があります。

このタスクについて

手順

1. WSDL2Java ツールを使用してプロキシー・クライアントを生成します。 タイプ:

wsdl2java *options WSDLfilepath*

各部の意味は、次のとおりです。

- *オプション* は以下のとおりです。

  -noWrappedOperations (-w)
  ラップ操作の検出を使用不可にします。要求および応答メッセージ用の Java Bean が生成されます。

  注: これはデフォルト値ではありません。

  -role (-r)
  値 **client** を指定すると、クライアント・サイド開発用のファイルおよびバインディング・ファイルが生成されます。

  -container (-c)
  使用するクライアント・サイド・コンテナー。有効な引数は以下のとおりです。

  クライアント
  クライアント・コンテナー

  ejb Enterprise JavaBeans (EJB) コンテナー。

  none コンテナーなし

  web Web コンテナー

  -output (-o)
  生成されたファイルを保管するフォルダー。

  WSDL2Java パラメーターの完全なリストが必要な場合は、**/help** コマンド行スイッチを使用するか、WID/RAD で WSDL2Java ツールのオンライン・ヘルプを参照してください。

- *WSDLfilepath* は、WebSphere 環境からエクスポートしたか、クライアント CD からコピーした WSDL ファイルのパスおよびファイル名です。

次の例では、ヒューマン・タスク・アクティビティー Web サービス API 用のプロキシー・クライアントが生成されます。

```
call wsdl2java.bat -r client -c client -noWrappedOperations
                   -output c:¥ws¥proxyClient  c:¥ws¥bin¥HTMWS.wsdl
```

2. 生成されたクラス・ファイルをプロジェクトに組み込みます。

関連タスク

564 ページの『クライアント・アプリケーションの作成 (Java Web サービス)』
クライアント・アプリケーションは、Web サービス API に要求を送信し、Web サービス API からの応答を受信します。プロキシー・クライアントを使用して、複合データ・タイプのフォーマット設定を行う通信クラスおよびヘルパー・クラスを管理すると、クライアント・アプリケーションから、Web サービス・メソッドをローカル機能のように呼び出すことができます。

BPEL プロセス用ヘルパー・クラスの作成 (Java Web サービス)

具象 API 要求 (sendMessage、call など) で参照されるビジネス・オブジェクトでは、クライアント・アプリケーションが「document/literal wrapped」スタイル・エレ

メントを使用する必要があります。クライアント・アプリケーションは、ヘルパー・クラスに、必要なラッパー・エレメントの生成を担当させます。

始める前に

ヘルパー・クラスを作成するには、Web サービス API の WSDL ファイルを、WebSphere Process Server 環境からエクスポートしておく必要があります。

このタスクについて

Web サービス API の call() 命令や sendMessage() 命令を実行すると、WebSphere Process Server 上で、BPEL プロセスとの対話ができるようになります。call() 命令の入力メッセージは、プロセスの入力メッセージの document/literal ラッパーが提供されると予想します。

BPEL プロセスまたはヒューマン・タスク用のヘルパー・クラスを生成するには、次のようないくつかの技法があります。

1. SoapElement オブジェクトを使用します。

   WebSphere Integration Developer で使用可能な Rational Application Developer 環境では、Web サービス・エンジンが JAX-RPC 1.1 をサポートします。JAX-RPC 1.1 では、SoapElement オブジェクトが Document Object Model (DOM) エレメントを拡張するため、DOM API を使用した SOAP メッセージの作成、読み取り、ロード、および保管が可能になりました。

   例えば、ワークフロー・プロセスまたはヒューマン・タスク用として以下の入力メッセージが WSDL ファイルに含まれることを想定してみます。

   ```
   <xsd:element name="operation1">
           <xsd:complexType>
             <xsd:sequence>
               <xsd:element name="input1" nillable="true" type="xsd:string"/>
             </xsd:sequence>
           </xsd:complexType>
         </xsd:element>
   ```

   プロセスまたはヒューマン・タスク・モジュールを開発する際に、WSDL ファイルが作成されます。

   DOM API を使用して、クライアント・アプリケーション内の対応する SOAP メッセージを作成するには、次のようにします。

   ```
   SOAPFactory soapfactoryinstance = SOAPFactory.newInstance();
   SOAPElement soapmessage = soapfactoryinstance.createElement
                             ("operation1", namespaceprefix, interfaceURI);
   SOAPElement inputelement = soapfactoryinstance.createElement("input1");
   inputelement.addTextNode( message value);
   soapmessage.addChildElement(outputelement);
   ```

   以下の例では、クライアント・アプリケーションで sendMessage 操作用の入力パラメーターを作成する方法を示します。

   ```
   SendMessage inWsend = new SendMessage();
   inWsend.setProcessTemplateName(processtemplatename);
   inWsend.setPortType(porttype);
   inWsend.setOperation(operationname);
   inWsend.set_any(soapmessage);
   ```

2. WebSphere Custom Data Binding 機能を使用します。

この技法は、以下の developerWorks 記事に記載されています。

- How to choose a custom mapping technology for Web services
- Developing Web Services with EMF SDOs for complex XML schema

関連タスク

553 ページの『ビジネス・オブジェクトのエクスポート』
ビジネス・プロセスおよびヒューマン・タスクには、Web サービスとして外部にアクセスできるよう明確に定義されたインターフェースが用意されています。これらのインターフェースがビジネス・オブジェクトを参照する場合は、インターフェース定義とビジネス・オブジェクトをクライアント・プログラミング環境にエクスポートする必要があります。

Interoperability With Patterns and Strategies for Document-Based Web Services

Web Services support for Schema/WSDL(s) containing optional JAX-RPC 1.0/1.1 XML Schema Types

クライアント・アプリケーションの作成 (Java Web サービス)

クライアント・アプリケーションは、Web サービス API に要求を送信し、Web サービス API からの応答を受信します。プロキシー・クライアントを使用して、複合データ・タイプのフォーマット設定を行う通信クラスおよびヘルパー・クラスを管理すると、クライアント・アプリケーションから、Web サービス・メソッドをローカル機能のように呼び出すことができます。

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、プロキシー・クライアントと必要なヘルパー・クラスを生成します。

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 (RAD) など) を使用すれば、クライアント・アプリケーションを開発できます。どのタイプの Web サービス・アプリケーションをビルドしても、Web サービス API を呼び出すことができます。

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. プロキシー・クライアントを生成し 、Java ヘルパー・クラスをプロジェクトに追加します。
3. クライアント・アプリケーションをコーディングします。
4. プロジェクトをビルドします。
5. クライアント・アプリケーションを実行します。

次の例では、Business Flow Manager Web サービス API を使用する方法を示します。

```
// create the proxy
       BFMIFProxy proxy = new BFMIFProxy();
 // prepare the input data for the operation
      GetProcessTemplate iW = new GetProcessTemplate();
      iW.setIdentifier(your_process_template_name);

 // invoke the operation
      GetProcessTemplateResponse oW = proxy.getProcessTemplate(iW);

 // process output of the operation
      ProcessTemplateType ptd = oW.getProcessTemplate();
      System.out.println("getName= " + ptd.getName());
      System.out.println("getPtid= " + ptd.getPtid());
```

関連タスク

559 ページの『プロキシー・クライアントの生成 (Java Web サービス)』
Java Web サービス・クライアント・アプリケーションは、プロキシー・クライアント を使用して Web サービス API と対話します。

562 ページの『BPEL プロセス用ヘルパー・クラスの作成 (Java Web サービス)』
具象 API 要求 (sendMessage、call など) で参照されるビジネス・オブジェクトでは、クライアント・アプリケーションが「document/literal wrapped」スタイル・エレメントを使用する必要があります。クライアント・アプリケーションは、ヘルパー・クラスに、必要なラッパー・エレメントの生成を担当させます。

561 ページの『WSDL2Java によるプロキシー・クライアントの生成』
WSDL2Java は、プロキシー・クライアントを生成するコマンド行ツールです。プロキシー・クライアントによって、クライアント・アプリケーションのプログラミングが容易になります。

セキュリティーの追加 (Java Web サービス)

クライアント・アプリケーションにセキュリティー・メカニズムを実装することにより、 Web サービス通信を保護する必要があります。

WebSphere Application Server は現在、Web サービス API の次のセキュリティー・メカニズムをサポートしています。

- ユーザー名トークン
- Lightweight Third Party Authentication (LTPA)

関連概念

75 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 J2EE のロールまたはインスタンス・ベースのロールとすることができます。

36 ページの『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、J2EE のロールまたはインスタンス・ベースのロールとすることができます。

ユーザー名トークンのインプリメント

ユーザー名トークンのセキュリティー・メカニズムは、ユーザー名とパスワードの信任状を提供します。

このタスクについて

ユーザー名トークンのセキュリティー・メカニズムで、さまざまなコールバック・ハンドラーをインプリメントすることが選択できます。次の選択方法があります。

- クライアント・アプリケーションを実行するたびに、ユーザー名とパスワードを入力するようプロンプトが出されます。
- ユーザー名とパスワードは、デプロイメント記述子に書き込まれます。

いずれの場合も、入力したユーザー名とパスワードは、対応するビジネス・プロセス・コンテナーまたはヒューマン・タスク・コンテナーの権限があるロールのユーザー名とパスワードと一致する必要があります。

ユーザー名とパスワードは、要求メッセージ・エンベロープにカプセル化されるので、SOAP メッセージ・ヘッダーには「明確に」表示されます。したがって、HTTPS (HTTP over SSL) 通信プロトコルを使用できるようクライアント・アプリケーションを構成することをお勧めします。そうすると、すべての通信が暗号化されます。Web サービス API のエンドポイント URL アドレスを指定するときに、HTTPS 通信プロトコルを選択することができます。

ユーザー名トークンを定義するには、次のようにします。

手順

1. セキュリティー・トークンを作成します。
   a. モジュールの「**デプロイメント・エディター (Deployment Editor)**」を開きます。
   b. 「**WS 拡張**」タブをクリックします。
   c. 「**サービス参照**」の下に、次の「Web サービス参照 (Web Service References)」がリストされます。
      - service/BFMWSService (ビジネス・プロセスの場合)
      - service/HTMWSService (ヒューマン・タスクの場合)

      どちらがリストされるかは、プロキシー・クライアントの生成時に、BFMWS.wsdl (ビジネス・プロセスの場合)、HTMWS.wsdl (ヒューマン・タスクの場合)、またはその両方のうちのどれが追加されたかによって決まります。
   d. どちらのサービス参照の場合も、以下のようにします。
      1) いずれかの「**サービス参照**」を選択します。
      2) 「**要求生成プログラム構成**」セクションを展開します。
      3) 「**セキュリティー・トークン**」 サブセクションを展開します。
      4) 「**追加**」をクリックします。「セキュリティー・トークン」ウィンドウが開きます。
      5) 「**名前**」フィールドに、新規のセキュリティー・トークンの名前として **UserNameTokenBFM** または **UserNameTokenHTM** を入力します。
      6) 「**トークン・タイプ**」ドロップダウン・リストで、「**ユーザー名**」を選択します (「**ローカル名**」フィールドにはデフォルト値が自動的に入力されます)。

7) 「**URI**」 フィールドはブランクのままにします。ユーザー名トークンには URI 値は不要です。

8) 「**OK**」をクリックします。

2. 以下のように、トークン生成プログラムを作成します。

a. モジュールの「**デプロイメント・エディター (Deployment Editor)**」を開きます。

b. 「**WS バインディング**」タブをクリックします。

c. 「**サービス参照**」の下に、前のステップと同じ Web サービス参照がリストされます。

- service/BFMWSService (ビジネス・プロセスの場合)
- service/HTMWSService (ヒューマン・タスクの場合)

d. どちらのサービス参照の場合も、以下のようにします。

1) いずれかの「**サービス参照**」を選択します。

2) 「**セキュリティー要求生成プログラムのバインディング構成**」セクションを展開します。

3) 「**トークン生成プログラム**」サブセクションを展開します。

4) 「**追加**」をクリックします。「トークン生成プログラム」ウィンドウが開きます。

5) 「**名前**」 フィールドに、新規のトークン生成プログラムの名前 (「UserNameTokenGeneratorBFM」または 「UserNameTokenGeneratorHTM」など) を入力します。

6) 「**トークン生成クラス**」フィールドで、トークン生成プログラム・クラス **com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator** が選択されていることを確認します。

7) 「**セキュリティー・トークン**」ドロップダウン・リストで、前に作成した適切なセキュリティー・トークンを選択します。

8) 「**値タイプの使用**」チェック・ボックスを選択します。

9) 「**値のタイプ**」フィールドで、「**ユーザー名トークン (Username Token)**」を選択します(「**ローカル名**」フィールドは、「**ユーザー名トークン**」の選択を反映して自動的に入力されます)。

10) 「**コールバック・ハンドラー**」フィールドで、 「com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler」(クライアント・アプリケーションを実行すると、ユーザー名およびパスワードを入力するようプロンプトが表示される)、または 「com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler」と入力します。

11) **NonPromptCallbackHandler** を選択した場合は、有効なユーザー名とパスワードを、デプロイメント記述子の対応するフィールドに指定する必要があります。

12) 「**OK**」をクリックします。

関連タスク

549 ページの『Web サービス・エンドポイント・アドレスの指定』

Web サービス・エンドポイント・アドレスは、クライアント・アプリケーショ

ンが Web サービス API にアクセスするために 指定する必要のある URL です。エンドポイント・アドレスは、クライアント・アプリケーション用の プロキシー・クライアントを生成するためにエクスポートする WSDL ファイルに書き込まれます。

関連情報

IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6

LTPA セキュリティー・メカニズムのインプリメント

Lightweight Third Party Authentication (LTPA) セキュリティー・メカニズムは、クライアント・アプリケーションが前に設定済みのセキュリティー・コンテキスト内で実行しているとき使用することができます。

このタスクについて

LTPA セキュリティー・メカニズムは、セキュリティー・コンテキストがすでに確立されている安全な環境でクライアント・アプリケーションが実行している場合にのみ使用可能です。例えば、クライアント・アプリケーションが Enterprise JavaBeans (EJB) コンテナーで実行されている場合、EJB クライアントはクライアント・アプリケーションを呼び出す前にログインする必要があります。次に、セキュリティー・コンテキストが確立されます。次に、EJB クライアントが Web サービスを呼び出すと、LTPA コールバック・ハンドラーが LTPA トークンをセキュリティー・コンテキストから取得し、それを SOAP 要求メッセージに追加します。サーバー・サイドでは、LTPA トークンが LTPA メカニズムによって処理されます。

LTPA セキュリティー・メカニズムをインプリメントするには、以下を実行します。

手順

1. WebSphere Integration Developer 内で使用可能な Rational Application Developer 環境で、「**WS バインディング (WS Binding)**」 → 「**セキュリティー要求ジェネレーターのバインディング構成 (Security Request Generator Binding Configuration)**」 → 「**トークン・ジェネレーター (Token Generator)**」を選択します。
2. セキュリティー・トークンを作成します。
   a. モジュールの「**デプロイメント・エディター (Deployment Editor)**」を開きます。
   b. 「**WS 拡張**」タブをクリックします。
   c. 「**サービス参照**」の下に、次の「**Web サービス参照 (Web Service References)**」がリストされます。
      - service/BFMWSService (ビジネス・プロセスの場合)
      - service/HTMWSService (ヒューマン・タスクの場合)

      どちらがリストされるかは、プロキシー・クライアントの生成時に、BFMWS.wsdl (ビジネス・プロセスの場合)、HTMWS.wsdl (ヒューマン・タスクの場合)、またはその両方のうちのどれが追加されたかによって決まります。

d. どちらのサービス参照の場合も、以下のようにします。
   1) いずれかの「**サービス参照**」を選択します。
   2) 「**要求生成プログラム構成**」セクションを展開します。
   3) 「**セキュリティー・トークン**」 サブセクションを展開します。
   4) 「**追加**」をクリックします。「セキュリティー・トークン」ウィンドウが開きます。
   5) 「**名前**」フィールドに、新規のセキュリティー・トークンの名前として **LTPATokenBFM** または **LTPATokenHTM** を入力します。
   6) 「**トークン・タイプ**」ドロップダウン・リストで、「**LTPAToken**」を選択します (「**URI**」および「**ローカル名**」フィールドには、自動的にデフォルト値が入力されます)。
   7) 「**OK**」をクリックします。

3. 以下のように、トークン生成プログラムを作成します。
   a. モジュールの「**デプロイメント・エディター (Deployment Editor)**」を開きます。
   b. 「**WS バインディング**」タブをクリックします。
   c. 「**サービス参照**」の下に、前のステップと同じ Web サービス参照がリストされます。
      - service/BFMWSService (ビジネス・プロセスの場合)
      - service/HTMWSService (ヒューマン・タスクの場合)
   d. どちらのサービス参照の場合も、以下のようにします。
      1) いずれかの「**サービス参照**」を選択します。
      2) 「**セキュリティー要求生成プログラムのバインディング構成**」セクションを展開します。
      3) 「**トークン生成プログラム**」サブセクションを展開します。
      4) 「**追加**」をクリックします。「トークン生成プログラム」ウィンドウが開きます。
      5) 「**名前**」 フィールドに、新規のトークン生成プログラムの名前 (「LTPATokenGeneratorBFM」または「LTPATokenGeneratorHTM」など) を入力します。
      6) 「**トークン生成クラス**」フィールドで、トークン生成プログラム・クラス **com.ibm.wsspi.wssecurity.token.LTPATokenGenerator** が選択されていることを確認します。
      7) 「**セキュリティー・トークン**」ドロップダウン・リストで、前に作成した適切なセキュリティー・トークンを選択します。
      8) 「**値タイプの使用**」チェック・ボックスを選択します。
      9) 「**値のタイプ**」フィールドで、「**LTPAToken**」を選択します (「**URI**」および「**ローカル名**」フィールドは、「**LTPA トークン (LTPA Token)**」の選択を反映して自動的に入力されます)。
      10) 「**コールバック・ハンドラー**」フィールドに、「com.ibm.wsspi.wssecurity.auth.callback.LTPATokenCallbackHandler」と入力します。
      11) 「**OK**」をクリックします。

結果

実行時に、**LTPATokenCallbackHandler** は、LTPA トークンを既存のセキュリティー・コンテキストから取得し、それを SOAP 要求メッセージに追加します。

トランザクション・サポートの追加 (Java Web サービス)

Java Web サービス・クライアント・アプリケーションは、クライアント・アプリケーション・コンテキストをサービス要求の一部として渡すことによって、サーバー・サイドの要求処理がクライアントのトランザクションに参加するように構成することができます。このアトミック・トランザクション・サポートは、Web Services-Atomic Transaction (WS-AT) 仕様で定義されています。

このタスクについて

WebSphere Application Server は、個々の Web サービス API 要求を別々のアトミック・トランザクションとして実行します。クライアント・アプリケーションは、以下のいずれかの方法でトランザクション・サポートを使用するよう構成できます。

- トランザクションに参加する。サーバー・サイドの要求処理は、クライアント・アプリケーション・トランザクション・コンテキスト内で実行されます。この場合、Web サービス API 要求の実行中やロールバック中にサーバーに問題が発生すると、クライアント・アプリケーションの要求もロールバックされます。
- トランザクション・サポートを使用しない。WebSphere Application Server はやはり新しいトランザクションを作成して、その中で要求を実行しますが、サーバー・サイドの要求処理はクライアント・アプリケーション・トランザクション・コンテキストで実行されません。

.NET 環境でのクライアント・アプリケーションの開発

Microsoft .NET は、Web サービスを使用してアプリケーションを接続する強力な開発環境を提供します。

プロキシー・クライアントの生成 (.NET)

.NET クライアント・アプリケーションは、*プロキシー・クライアント* を使用して Web サービス API と対話します。プロキシー・クライアントのおかげで、クライアント・アプリケーションは、複雑な Web サービス・メッセージング・プロトコルを意識する必要がなくなります。

始める前に

プロキシー・クライアントを作成するには、まず WebSphere 環境からいくつかの WSDL ファイルをエクスポートして、それをクライアントのプログラミング環境にコピーする必要があります。

注: WebSphere Process Server クライアント CD を所有している場合は、代わりにその CD からファイルをコピーすることもできます。

このタスクについて

プロキシー・クライアントは、C# Bean クラスのセットで構成されています。各クラスには、単一の Web サービスで公開されるメソッドおよびオブジェクトがすべて含まれています。サービス・メソッドは、パラメーターを完全な SOAP メッセージにアセンブルし、その SOAP メッセージを HTTP 経由で Web サービスに送信し、Web サービスから応答を受信して、戻されたデータを処理します。

注: プロキシー・クライアントの生成が必要なのは一度だけです。 Web サービス API にアクセスするクライアント・アプリケーションはすべて、以後は同じプロキシー・クライアントを使用できます。

手順

1. プロキシー・クライアントの生成には WSDL コマンドを使用します。 タイプ:

   wsdl *options WSDLfilepath*

   各部の意味は、次のとおりです。

   - オプション は以下のとおりです。

     /language
     プロキシー・クラスの作成に使用する言語を指定できます。デフォルトは C# です。言語引数として、**VB** (Visual Basic)、**JS** (JScript)、または **VJS** (Visual J#) を指定することもできます。

     /output
     該当するサフィックスの付いた出力ファイルの名前。例えば、proxy.cs です。

     /protocol
     プロキシー・クラスでインプリメントされるプロトコル。**SOAP** がデフォルトの設定です。

     WSDL.exe パラメーターの完全なリストが必要な場合は、**/?** コマンド行スイッチを使用するか、Visual Studio で WSDL ツールのオンライン・ヘルプを参照してください。

   - *WSDLfilepath* は、WebSphere 環境からエクスポートしたか、クライアント CD からコピーした WSDL ファイルのパスおよびファイル名です。

   次の例では、ヒューマン・タスク・マネージャー Web サービス API 用のプロキシー・クライアントが生成されます。

   ```
   wsdl /language:cs /output:proxyclient.cs c:¥ws¥bin¥HTMWS.wsdl
   ```

2. プロキシー・クライアントをダイナミック・リンク・ライブラリー (DLL) ファイルとしてコンパイルします。

関連タスク

551 ページの『WSDL ファイルの公開』
Web サービス記述言語 (WSDL) ファイルには、Web サービス API で使用できるすべての操作の詳細な説明が含まれています。Business Flow Manager Web サービス API と Human Task Manager Web サービス API では、使用できる WSDL ファイルが異なります。これらの WSDL ファイルは、まず公開して、その後 WebSphere 環境からご使用の開発環境にコピーし、プロキシー・クライアントの生成に使用することになります。

BPEL プロセス用ヘルパー・クラスの作成 (.NET)

特定の Web サービス API 操作では、クライアント・アプリケーションが「document/literal」スタイルのラップ・エレメントを使用する必要があります。クライアント・アプリケーションは、ヘルパー・クラスに、必要なラッパー・エレメントの生成を担当させます。

始める前に

ヘルパー・クラスを作成するには、Web サービス API の WSDL ファイルを、WebSphere Process Server 環境からエクスポートしておく必要があります。

このタスクについて

Web サービス API の call() 命令と sendMessage() 命令により、BPEL プロセスが WebSphere Process Server 内で起動します。call() 命令の入力メッセージは、BPEL プロセスの入力メッセージの document/literal ラッパーが提供されると予想します。BPEL プロセスに必要な Bean およびクラスを生成するには、`<wsdl:types>` エレメントを新規 XSD ファイルにコピーしてから、xsd.exe ツールを使用してヘルパー・クラスを生成します。

手順

1. BPEL プロセス・インターフェースの WSDL ファイルをまだ WebSphere Integration Developer からエクスポートしていない場合は、ここでエクスポートします。
2. テキスト・エディターまたは XML エディターで WSDL ファイルを開きます。
3. `<wsdl:types>` エレメントのすべての子エレメントの内容をコピーし、新しいスケルトン XSD ファイルに貼り付けます。
4. XSD ファイル上で `xsd.exe` ツールを実行します。

   call xsd.exe file.xsd /classes /o

   各部の意味は、次のとおりです。

   file.xsd 変換する XML スキーマ定義ファイル。

   /classes (/c)
   指定した XSD ファイル (複数も可) の内容に対応するヘルパー・クラスを生成します。

   /output (/o)
   生成したファイルの出力ディレクトリーを指定します。このディレクトリーを省略した場合、デフォルトは現行ディレクトリーです。

   以下に例を示します。

   call xsd.exe ProcessCustomer.xsd /classes /output:c:¥temp
5. 生成されるクラス・ファイルをクライアント・アプリケーションに追加します。例えば Visual Studio を使用する場合は、「**プロジェクト**」 → 「**既存項目の追加 (Add Existing Item)**」メニュー・オプションを実行します。

ProcessCustomer.wsdl ファイルの内容が以下のような場合:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:bons1="http://com/ibm/bpe/unittest/sca"
                  xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  name="ProcessCustomer"
                  targetNamespace="http://ProcessTypes/bpel/ProcessCustomer">
  <wsdl:types>
    <xsd:schema targetNamespace="http://ProcessTypes/bpel/ProcessCustomer"
                xmlns:bons1="http://com/ibm/bpe/unittest/sca"
                xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:import namespace="http://com/ibm/bpe/unittest/sca"
                schemaLocation="xsd-includes/http.com.ibm.bpe.unittest.sca.xsd"/>
      <xsd:element name="doit">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="input1" nillable="true" type="bons1:Customer"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="doitResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="output1" nillable="true" type="bons1:Customer"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
    <wsdl:message name="doitRequestMsg">
    <wsdl:part element="tns:doit" name="doitParameters"/>
  </wsdl:message>
    <wsdl:message name="doitResponseMsg">
    <wsdl:part element="tns:doitResponse" name="doitResult"/>
  </wsdl:message>
    <wsdl:portType name="ProcessCustomer">
    <wsdl:operation name="doit">
      <wsdl:input message="tns:doitRequestMsg" name="doitRequest"/>
      <wsdl:output message="tns:doitResponseMsg" name="doitResponse"/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

結果として出力される XSD ファイルは以下のようになります。

```
<xsd:schema xmlns:bons1="http://com/ibm/bpe/unittest/sca"
                     xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
                     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                     targetNamespace="http://ProcessTypes/bpel/ProcessCustomer">
 <xsd:import namespace="http://com/ibm/bpe/unittest/sca"
             schemaLocation="Customer.xsd"/>
 <xsd:element name="doit">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="input1" type="bons1:Customer" nillable="true"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="doitResponse">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="output1" type="bons1:Customer" nillable="true"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

関連タスク

553 ページの『ビジネス・オブジェクトのエクスポート』
ビジネス・プロセスおよびヒューマン・タスクには、Web サービスとして外部にアクセスできるよう明確に定義されたインターフェースが用意されています。これらのインターフェースがビジネス・オブジェクトを参照する場合は、インターフェース定義とビジネス・オブジェクトをクライアント・プログラミング環境にエクスポートする必要があります。

関連情報

XML スキーマ定義ツール (XSD.EXE) の Microsoft 文書

クライアント・アプリケーションの作成 (.NET)

クライアント・アプリケーションは、Web サービス API に要求を送信し、Web サービス API からの応答を受信します。プロキシー・クライアントを使用して、複合データ・タイプのフォーマット設定を行う通信クラスおよびヘルパー・クラスを管理すると、クライアント・アプリケーションから、Web サービス・メソッドをローカル機能のように呼び出すことができます。

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、プロキシー・クライアントと必要なヘルパー・クラスを生成します。

このタスクについて

.NET 対応の開発ツール (Visual Studio .NET など) を使用すれば、.NET クライアント・アプリケーションを開発できます。どのタイプの .NET アプリケーションをビルドしても、汎用の Web サービス API を呼び出すことができます。

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。 例えば、Visual Studio で **WinFX Windows Application** を作成します。
2. プロジェクト・オプションで、プロキシー・クライアントのダイナミック・リンク・ライブラリー (DLL) ファイルに参照を追加します。 ビジネス・オブジェクト定義を含むすべてのヘルパー・クラスをプロジェクトに追加します。例えば Visual Studio では、**「プロジェクト」 → 「既存項目の追加 (Add existing item)」**オプションを実行します。
3. プロキシー・クライアント・オブジェクトを作成します。 以下に例を挙げます。

```
HTMClient.HTMReference.HumanTaskManagerComponent1Export_HumanTaskManagerHttpService service =
    new HTMClient.HTMReference.HumanTaskManagerComponent1Export_HumanTaskManagerHttpService();
```

4. メッセージで使用され、Web サービスとの間で送受信されるビジネス・オブジェクト・データ・タイプを宣言します。 以下に例を挙げます。

```
HTMClient.HTMReference.TKIID id = new HTMClient.HTMReference.TKIID();

ClipBG bg = new ClipBG();
Clip clip = new Clip();
```

5. 特定の Web サービス関数を呼び出し、必要なパラメーターを指定します。 例えば、ヒューマン・タスクを作成して開始するには、以下のようにします。

```
HTMClient.HTMReference.createAndStartTask task =
   new HTMClient.HTMReference.createAndStartTask();
HTMClient.HTMReference.StartTask sTask = new HTMClient.HTMReference.StartTask();

sTask.taskName = "SimpleTask";
sTask.taskNamespace = "http://myProcess/com/acme/task";
sTask.inputMessage = bg;
task.inputTask = sTask;

id = service.createAndStartTask(task).outputTask;
```

6. リモートのプロセスおよびタスクは、永続 ID (上記の例では id) で識別されます。 例えば、上記で作成したヒューマン・タスクを要求するには、以下のようにします。

```
HTMClient.HTMReference.claimTask claim = new HTMClient.HTMReference.claimTask();
claim.inputTask = id;
```

関連タスク

570 ページの『プロキシー・クライアントの生成 (.NET)』
.NET クライアント・アプリケーションは、*プロキシー・クライアント* を使用して Web サービス API と対話します。プロキシー・クライアントのおかげで、クライアント・アプリケーションは、複雑な Web サービス・メッセージング・プロトコルを意識する必要がなくなります。

572 ページの『BPEL プロセス用ヘルパー・クラスの作成 (.NET)』
特定の Web サービス API 操作では、クライアント・アプリケーションが「document/literal」スタイルのラップ・エレメントを使用する必要があります。クライアント・アプリケーションは、ヘルパー・クラスに、必要なラッパー・エレメントの生成を担当させます。

セキュリティーの追加 (.NET)

Web サービス通信は、クライアント・アプリケーションにセキュリティー・メカニズムを組み込むことにより保護できます。

このタスクについて

このセキュリティー・メカニズムには、ユーザー名トークン (ユーザー名とパスワード)、またはカスタム・バイナリーおよび XML ベースのセキュリティー・トークンが使用できます。

手順

1. Web Services Enhancements (WSE) 2.0 SP3 for Microsoft .NET をダウンロードしてインストールします。 入手先は以下のとおりです。

   http://www.microsoft.com/downloads/details.aspx?familyid=1ba1f631-c3e7-420a-bc1e-ef18bab66122&displaylang=en

2. 生成されたプロキシー・クライアント・コードを以下のように変更します。

   変更前:

```
public class Export1_MyMicroflowHttpService : System.Web.Services.Protocols.SoapHttpClientProtocol {
```

   変更後:

```
public class Export1_MyMicroflowHttpService : Microsoft.Web.Services2.WebServicesClientProtocol {
```

注: この変更内容は、WSDL.exe ツールを実行してプロキシー・クライアントを再生成すると失われます。

3. ファイルの先頭に以下の行を追加して、クライアント・アプリケーション・コードを変更します。

```
using System.Web.Services.Protocols;
using Microsoft.Web.Services2;
using Microsoft.Web.Services2.Security.Tokens;
...
```

4. 必要なセキュリティー・メカニズムをインプリメントするコードを追加します。例えば、ユーザー名とパスワード保護を追加するコードは次のとおりです。

```
string user = "U1";
string pwd = "password";
UsernameToken token =
   new UsernameToken(user, pwd, PasswordOption.SendPlainText);

me._proxy.RequestSoapContext.Security.Tokens.Clear();
me._proxy.RequestSoapContext.Security.Tokens.Add(token);
```

ビジネス・プロセスおよびタスク関連のオブジェクトの照会

Web サービス API を使用すると、Business Process Choreographer データベース内のビジネス・プロセス・オブジェクトおよびタスク関連オブジェクトを照会して、これらのオブジェクトの特定のプロパティーを取得することができます。

このタスクについて

Business Process Choreographer データベースは、ビジネス・プロセスおよびタスクの管理用のテンプレート (モデル) とインスタンス (ランタイム) のデータを保管します。

クライアント・アプリケーションは、Web サービス API 経由で照会を発行し、データベースからビジネス・プロセスおよびビジネス・タスクに関する情報を取得することができます。

クライアント・アプリケーションは、1 回限りの照会を発行して、オブジェクトの特定のプロパティーを取得することができます。使用頻度の高い照会は、保管しておくことができます。クライアント・アプリケーションは、このような保管照会文を後で取り出して使用することができます。

ビジネス・プロセスおよびタスク関連オブジェクトに対する照会

Web サービス API の QUERY インターフェースを使用して、ビジネス・プロセスおよびタスクに関する情報を取得します。

クライアント・アプリケーションは、SQL 形式の構文を使用して、データベースを照会します。

Java Web サービスの例

```
string processTemplateName = "ProcessCustomerLR";
query query1 = new query();
query1.selectClause = "DISTINCT PROCESS_INSTANCE.STARTED, PROCESS_INSTANCE.PIID";
query1.whereClause =
       "PROCESS_INSTANCE.TEMPLATE_NAME = '" + processTemplateName + "'";
query1.orderByClause = "PROCESS_INSTANCE.STARTED";
```

```
query1.threshold = null;
query1.timeZone = "UTC"; query1.skipTuples = null;
queryResponse queryResponse1 = proxy.query(query1);
```

データベースから取り出した情報は、Web サービス API を使用して*照会結果セット*として戻されます。

以下に例を挙げます。

```
QueryResultSetType queryResultSet = queryResponse1.queryResultSet;
if (queryResultSet != null) {
   Console.WriteLine("--> QueryResultSetType");
   Console.WriteLine(" . size= " + queryResultSet.size);
   Console.WriteLine(" . numberColumns= " + queryResultSet.numberColumns);
   string indent = " . ";

   // -- the query column info
   QueryColumnInfoType[] queryColumnInfo = queryResultSet.QueryColumnInfo;
   if (queryColumnInfo.Length > 0) {
      Console.WriteLine();
      Console.WriteLine("= . QueryColumnInfoType size= " + queryColumnInfo.Length);
      Console.Write( " | tableName ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         Console.Write( " | " + queryColumnInfo[i].tableName.PadLeft(20) );
      }
      Console.WriteLine();
      Console.Write( " | columnName ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         Console.Write( " | " + queryColumnInfo[i].columnName.PadLeft(20) );
      }
      Console.WriteLine();
      Console.Write( " | data type ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         QueryColumnInfoTypeType tt = queryColumnInfo[i].type;
         Console.WriteLine( " | " + tt.ToString());
      }
         Console.WriteLine();
   }
   else {
      Console.WriteLine("--> queryColumnInfo= <null>");
   }

   // - the query result values
   string[][] result = queryResultSet.result;
   if (result !=null) {
      Console.WriteLine();
      Console.WriteLine("= . result size= " + result.Length);
      for (int i = 0; i &lt; result.Length; i++) {
        Console.Write(indent +i );
        string[] row = result[i];
        for (int j = 0; j &lt; row.Length; j++ ) {
           Console.Write(" | " + row[j]);
        }
        Console.WriteLine();
      }
   }
   else {
      Console.WriteLine("--> result= <null>");
   }
}
else {
   Console.WriteLine("--> QueryResultSetType= <null>");
}
```

照会関数は、呼び出し元の権限に応じてオブジェクトを戻します。照会結果セットには、呼び出し元が表示を許可されているオブジェクトのプロパティーのみが含まれます。

オブジェクトのプロパティーを照会するために、事前定義データベース・ビューが提供されています。プロセス・テンプレートの場合、照会関数には以下の構文があります。

```
ProcessTemplateData[] queryProcessTemplates
                      (java.lang.String whereClause,
                       java.lang.String orderByClause,
                       java.lang.Integer threshold,
                       java.util.TimeZone timezone);
```

タスク・テンプレートの場合、照会関数には以下の構文があります。

```
TaskTemplate[] queryTaskTemplates
               (java.lang.String whereClause,
                java.lang.String orderByClause,
                java.lang.Integer threshold,
                java.util.TimeZone timezone);
```

他のビジネス・プロセスおよびタスク関連オブジェクトの場合、照会関数には以下の構文があります。

```
QueryResultSet query (java.lang.String selectClause,
                      java.lang.String whereClause,
                      java.lang.String orderByClause,
                      java.lang.Integer skipTuples
                      java.lang.Integer threshold,
                      java.util.TimeZone timezone);
```

QUERY インターフェースには、queryAll メソッドも含まれています。このメソッドを使用して、オブジェクトに関係のあるデータすべてを、モニターなどの目的で取得することができます。queryAll メソッドの呼び出し元には、Java 2 Platform Enterprise Edition (J2EE) ロールの、BPESystemAdministrator、BPESystemMonitor、TaskSystemAdministrator、または TaskSystemMonitor のいずれかが必要です。オブジェクトの対応する作業項目を使用した許可検査は適用されません。

.NET の例

```
ProcessTemplateType[] templates = null;

   try {
        queryProcessTemplates iW = new queryProcessTemplates();
        iW.whereClause = "PROCESS_TEMPLATE.STATE=PROCESS_TEMPLATE.STATE.STATE_STARTED";
        iW.orderByClause = null;
        iW.threshold = null;
        iW.timeZone = null;

        Console.WriteLine("--> queryProcessTemplates ... ");
        Console.WriteLine("--> query: WHERE " + iW.whereClause + " ORDER BY " +
          iW.orderByClause + " THRESHOLD " + iW.threshold + " TIMEZONE" + iW.timeZone);

        templates = proxy.queryProcessTemplates(iW);

        if (templates.Length < 1) {
          Console.WriteLine("--> No templates found :-(");
        }
        else {
```

```
    for (int i = 0; i < templates.Length ; i++) {
      Console.Write("--> found template with ptid: " + templates[i].ptid);
      Console.WriteLine(" and name: " + templates[i].name);
     /*  ... other properties of ProcessTemplateType ... */
    }
  }
}
catch( Exception e ) {
  Console.WriteLine("exception= " + e);
}
```

照会パラメーター

各照会では、複数の SQL 形式の文節およびパラメーターを指定する必要があります。

照会は以下のもので構成されます。

- select 文節
- where 文節
- order-by 文節
- スキップ・タプル・パラメーター
- しきい値パラメーター
- 時間帯パラメーター

関連概念

456 ページの『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。

457 ページの『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。

459 ページの『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。

459 ページの『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。

460 ページの『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。

460 ページの『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。

461 ページの『照会結果』
照会結果セットには、照会の結果が入ります。

ビジネス・プロセス・オブジェクトおよびヒューマン・タスク・オブジェクトの照会のための事前定義ビュー

ビジネス・プロセス・オブジェクトおよびヒューマン・タスク・オブジェクト用に、事前定義データベース・ビューが提供されています。

これらのオブジェクトの参照データを照会する場合は、これらのビューを使用します。これらのビューを使用する場合は、ビューの列に明示的に述部を結合する必要はありません。これらの構成要素は自動的に追加されます。Web サービス API の照会機能を使用して、このデータを照会できます。

関連資料

469 ページの『ACTIVITY ビュー』
この定義済みデータベース・ビューは、アクティビティーの照会に使用します。

470 ページの『ACTIVITY_ATTRIBUTE ビュー』
この定義済みデータベース・ビューは、アクティビティーのカスタム・プロパティーの照会に使用します。

471 ページの『ACTIVITY_SERVICE ビュー』
この定義済みデータベース・ビューは、アクティビティー・サービスの照会に使用します。

471 ページの『APPLICATION_COMP ビュー』
この定義済みデータベース・ビューは、アプリケーション・コンポーネント ID、およびタスクのデフォルト設定の照会に使用します。

472 ページの『ESCALATION ビュー』
この定義済みデータベース・ビューは、エスカレーションのデータの照会に使用します。

474 ページの『ESCALATION_CPROP ビュー』
この定義済みデータベース・ビューは、エスカレーションのカスタム・プロパティーを照会するために使用します。

474 ページの『ESCALATION_DESC ビュー』
この定義済みデータベース・ビューは、エスカレーションのマルチリンガル記述データを照会するために使用します。

477 ページの『PROCESS_ATTRIBUTE ビュー』
この定義済みデータベース・ビューは、プロセスのカスタム・プロパティーの照会に使用します。

477 ページの『PROCESS_INSTANCE ビュー』
この定義済みデータベース・ビューは、プロセス・インスタンスの照会に使用します。

478 ページの『PROCESS_TEMPLATE ビュー』
この定義済みデータベース・ビューは、プロセス・テンプレートの照会に使用します。

479 ページの『QUERY_PROPERTY ビュー』
この定義済みデータベース・ビューは、プロセス・レベル変数の照会に使用します。

480 ページの『TASK ビュー』
この定義済みデータベース・ビューは、タスク・オブジェクトの照会に使用します。

483 ページの『TASK_CPROP ビュー』
この定義済みデータベース・ビューは、タスク・オブジェクトのカスタム・プロパティーを照会するために使用します。

484 ページの『TASK_DESC ビュー』
この定義済みデータベース・ビューは、タスク・オブジェクトのマルチリンガル記述データを照会するために使用します。

484 ページの『TASK_TEMPL ビュー』
この定義済みデータベース・ビューは、タスクのインスタンスを生成するために使用できるデータを保持します。

486 ページの『TASK_TEMPL_CPROP ビュー』
この定義済みデータベース・ビューは、タスク・テンプレートのカスタム・プロパティーを照会するために使用します。

486 ページの『TASK_TEMPL_DESC ビュー』
この定義済みデータベース・ビューは、タスク・テンプレート・オブジェクトのマルチリンガル記述データを照会するために使用します。

487 ページの『WORK_ITEM ビュー』
この定義済みデータベース・ビューは、作業項目の照会や、プロセス、タスクおよびエスカレーション用の許可データの照会に使用します。

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。

共通保管照会文の管理
共通保管照会文はシステム管理者によって作成されます。 この照会は、全ユーザーが使用できます。

他のユーザーの専用保管照会文の管理
専用照会はどのユーザーでも作成できます。この照会は、照会の所有者とシステム管理者しか使用できません。

専用保管照会文の操作
システム管理者でなくても、自分専用の保管照会文は作成、実行、 および削除できます。また、システム管理者が作成した共通の保管照会文を 使用することもできます。

第 13 章 JMS 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 の紹介

WebSphere Process Server バージョン 6.1 は、通信手段として、Java Messaging Service (JMS) プログラミング・インターフェースに基づく非同期メッセージングをサポートします。

JMS は、要求を JMS メッセージとして作成、送信、受信、および読み取るため共通の方法を Java クライアント (クライアント・アプリケーションまたは J2EE アプリケーション) に提供します。

JMS は、以下を行う非同期メッセージングに基づくインターフェースです。

- **point-to-point メッセージングまたはパブリッシュ/サブスクライブ・メッセージング**のいずれかを使用します。メッセージに基づくフレームワークは、情報を明示的に要求しなくても、他のアプリケーションに情報をプッシュできます。同じ情報を多数のサブスクライバーに並行して送達できます。 Business Process Choreographer の JMS インターフェースは point-to-point メッセージングのみサポートします。
- **リズムの独立性**を提供します。JMS フレームワークは非同期に機能しますが、同期要求/応答モードをシミュレートすることもできます。これにより、ソース・システムおよびターゲット・システムは相互に待機しなくても同時に作業できます。この機能は Business Process Choreographer にとって非常に役立ちます。なぜなら、長期実行のビジネス・プロセスと非同期で対話する機能を提供するからです。
- **トランザクション**をサポートします。トランザクションにより、クライアント・アプリケーションは送受信されたメッセージのグループを単一のアトミック単位として処理できます。 JMS トランザクションはサーバーのトランザクション内で実行されます。 Business Process Choreographer の JMS インターフェースの場合、通常はトランザクションごとに単一のメッセージが送受信されます。
- **情報の送達を保証します**。JMS フレームワークはメッセージをトランザクション・モードで管理し、メッセージ送達を保証できます (とはいえ、送達の適時性を保証するものではありません)。 Business Process Choreographer の場合、この信頼できるメッセージ送達機能はビジネス・プロセスを処理するため、特に重要です。
- **異機種フレームワーク間のインターオペラビリティー**を保証します。ソース・アプリケーションとターゲット・アプリケーションは異機種環境で作動でき、それぞれのフレームワークに関連した通信および実行の問題を処理する必要がありません。

- **交換をさらに流動的にします。**メッセージ・モードを切り替えると、きめの細かい情報を交換できます。

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・オペレーションのビジネス・プロセスとヒューマン・タスクによって出される障害メッセージは、XML スキーマ・エレメントで定義された単一の WSDL メッセージ部分で構成されている必要があります。以下に例を示します。

   ```
   <wsdl:part name="myFault" element="myNamespace:myFaultElement"/>
   ```

   関連情報

   Java API for XML based RPC (JAX-RPC) のダウンロード・ページ

   使用すべき WSDL のスタイル

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。 JMS クライアントを J2SE 環境で実行する場合は、http://www-1.ibm.com/support/docview.wss?uid=swg24012804の「IBM Client for JMS on J2SE with IBM WebSphere Application Server」を参照してください。

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-acknowledgement.
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。メッセージを SCA Export の一部として処理している間 (例えば、メッセージの非直列化中) に障害が早期に発生する場合、SCA Export の受信宛先によって指定された失敗した送達の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

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 文書が入ったストリングです。

有効な要求メッセージ本文の単純な例を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<_6:queryProcessTemplates xmlns:_6="http://www.ibm.com/xmlns/prod/
      websphere/business-process/services/6.0">
<whereClause>PROCESS_TEMPLATE.STATE IN (1)</whereClause>
</_6:queryProcessTemplates>
```

関連タスク

591 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 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 API の概要

JMS メッセージ・インターフェース (今後は「JMS API」と表します) により、Business Process Choreographer 環境で実行しているビジネス・プロセス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。

JMS API により、クライアント・アプリケーションは Microflow および長期実行プロセスと非同期で対話できます。

JMS API は Web サービス API と同じインターフェースを公開します。ただし、以下の例外があります。

- Web サービス API と共に使用する場合、call 操作は Microflow を呼び出すためにのみ使用できます。しかし、JMS API を使用する場合、call 操作を使用して Microflow と長期実行プロセスの両方を呼び出すことができます。
- 以下の操作は JMS API を使用して公開されません。
  - callAsync 操作 (およびそれに関連したコールバック操作)
  - completeAndClaimSuccessor 操作および getParticipatingTask 操作

例 - 長期実行プロセスの実行

一般のクライアント・アプリケーションが長期実行プロセスで作業する場合、以下の一連のステップを実行します。

1. 584 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. 以下のように、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. オプションで、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。
   - 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。
   - getOutputMessage を発行します。
7. オプションで、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。
   - 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。
   - 入力メッセージを作成します。
   - sendMessage を発行してメッセージを送信します。

8. オプションで、プロセスに対して定義された、またはアクティビティーを含むカスタム・プロパティーを、getCustomProperties および setCustomProperties で取得および設定します。
9. オプションで、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。
   - 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。

JMS アプリケーションの開発

JMS クライアント・アプリケーションは、Java で Java 2 Enterprise Edition (J2EE) 環境を使用して開発する必要があります。

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。

成果物のコピー

JMS クライアント・アプリケーションの作成を補助するために、いくつかの成果物を WebSphere 環境からコピーすることができます。

BOXMLSerializer を使用して JMS メッセージ本文を作成する場合に限り、これらの成果物の使用が必須です。

成果物を取得するには、以下の 2 つの方法があります。

- WebSphere Process Server 環境から成果物を公開およびエクスポートします。

  WebSphere Process Server 6.1 の場合、すべてのクライアント成果物は *install_root*¥ProcessChoreographer¥client ディレクトリーにあります。 JMS API の場合、成果物は以下のとおりです。

  BFMIF.wsdl

  BFMIF.xsd

  BPCGen.xsd

- WebSphere Process Server クライアント CD からファイルをコピーします。

サーバー環境からの成果物の公開

JMS API 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るように、WebSphere サーバー環境から、いくつかの成果物を公開することができます。

このタスクについて

WebSphere Process Server 6.1 の場合、すべてのクライアント成果物は *was_home*¥ProcessChoreographer¥client ディレクトリーにあります。 JMS API の場合、成果物は以下のとおりです。

BFMIF.wsdl

BFMIF.xsd

BPCGen.xsd

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します。

クライアント CD からファイルのコピー

JMS API にアクセスするのに必要なファイルは、WebSphere Process Server クライアント CD で入手できます。

手順

1. クライアント CD にアクセスして、ProcessChoreographer¥client ディレクトリーを参照します。
2. 必要なファイルをクライアント・アプリケーション開発環境にコピーします。

   WebSphere Process Server 6.1 の場合、すべてのクライアント成果物は ¥ProcessChoreographer¥client ディレクトリーにあります。 JMS API の場合、成果物は以下のとおりです。

   BFMIF.wsdl

   BFMIF.xsd

   BPCGen.xsd

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。

```
// 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

586 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。

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。

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

   Web アプリケーションをリモート・サーバーにデプロイする場合、以下のファイルも追加してください。 Business Process Choreographer API にリモート側でアクセスするには、以下のファイルが必要です。

   - bpe137650.jar
   - task137650.jar

   WebSphere Process Server では、これらのすべてのファイルは以下のディレクトリーにあります。

   - Windows システムの場合: *install_root*¥ProcessChoreographer¥client
   - UNIX、Linux、および i5/OS システムの場合: *install_root*/ProcessChoreographer/client
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。

```
<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・コードをインプリメントします。

5. アプリケーションをデプロイします。

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 48. 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

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。

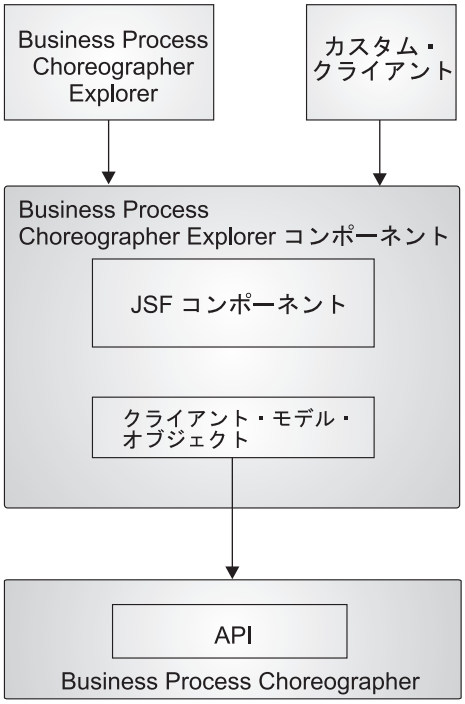
次のタスク

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インプリメントし、元のオブジェクトをラップします。クライアント・モデル・オブジェクトは、ラベルの各国語サポートと、一部のプロパティーのコンバーターを提供します。

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インプリメントします。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インプリメントもしない場合
- コンポーネント内でエラー・メッセージが表示され、メッセージのハイパーリンクに従っている場合

com.ibm.bpc.clientcore.util.ErrorBeanImpl インターフェースのデフォルト・インプリメンテーションを使用できます。

インターフェースは次のように定義されます。

```
public interface ErrorBean {

    public void setException(Exception ex);

    /*
     * This setter method call allows a locale and
     * the exception to be passed. This allows the
```

```
     * getExceptionMessage methods to return localized Strings
     *
     */
    public void setException(Exception ex, Locale locale);

    public Exception getException();
    public String getStack();
    public String getNestedExceptionMessage();
    public String getNestedExceptionStack();
    public String getRootExceptionMessage();
    public String getRootExceptionStack();

    /*
     * This method returns the exception message
     * concatenated recursively with the messages of all
     * the nested exceptions.
     */
    public String getAllExceptionMessages();

    /*
     * This method is returns the exception stack
     * concatenated recursively with the stacks of all
     * the nested exceptions.
     */
    public String getAllExceptionStacks();
}
```

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インプリメントします。

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 49. 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

表 49. *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 (続き)

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.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.
```

```
    //
    QueryResultSet queryResult = taskService.query(
     "DISTINCT TASK.TKIID, TASK.NAME, TASK.KIND, TASK.STATE, TASK.TYPE,"
      + "TASK.STARTED, TASK.ACTIVATED, TASK.DUE, TASK.EXPIRES, TASK.PRIORITY" ,
      "TASK.KIND IN(101,102,105) AND TASK.STATE IN(2)
        AND WORK_ITEM.REASON IN (1)",
      (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インプリメントする必要があります。リスト・ハンドラーは、内容を最新表示するときに、照会の execute メソッドを呼び出します。呼び出しによって、Java オブジェクトのリストが戻されます。getType メソッドは、戻された Java オブジェクトのクラス名を戻す必要があります。

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インプリメントされます。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インプリメントすることができます。

照会のインプリメンテーション

リスト・ハンドラーを使用すると、すべての種類のオブジェクトおよびそれらのプロパティーを表示できます。表示されるリストの内容は、リスト・ハンドラー用に構成された com.ibm.bpc.clientcore.Query インターフェースのインプリメンテーションによって戻されるオブジェクトのリストによって異なります。照会は、BPCListHandler クラスの setQuery メソッドを使用してプログラマチックに設定することもできますし、アプリケーションの JSF 構成ファイルで構成することもできます。

照会は、Business Process Choreographer API に対してだけでなく、コンテンツ管理システムやデータベースなど、アプリケーションからアクセスできるその他の情報ソースに対しても実行できます。要件は、照会の結果が execute メソッドでオブジェクトの `java.util.List` として戻されることだけです。

戻されるオブジェクトのタイプは、照会が定義されたリストの列に表示されるすべてのプロパティーに対して適切な getter メソッドを使用できることを保証する必要があります。戻されるオブジェクトのタイプがリスト定義に適合することを確認するには、faces 構成ファイルで定義されている BPCListHandler インスタンス上のタイプ・プロパティーの値を、戻されるオブジェクトの完全修飾クラス名に設定します。この名前は、照会インプリメンテーションの getType 呼び出しで戻すことができます。実行時に、リスト・ハンドラーはオブジェクト・タイプが定義に準拠していることを確認します。

リスト内の特定の項目にエラー・メッセージをマップするには、照会によって戻されたオブジェクトが署名 `public Object getID()` でメソッドをインプリメントする必要があります。

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インプリメンテーションが **user** を管理対象 Bean 名として設定されていると予期します。構成ファイル内でこの項目が欠けている場合は、WebSphere Process Server が動作している時間帯が戻されます。

com.ibm.bpc.clientcore.util.User インターフェースは次のように定義されています。

```
public interface User {

    /**
     * The locale used by the client of the user.
     * @return Locale.
     */
    public Locale getLocale();
   /**
    * The time zone used by the client of the user.
    * @return TimeZone.
    */
    public TimeZone getTimeZone();

   /**
    * The name of the user.
    * @return name of the user.
    */
    public String getName();
}
```

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インプリメンテーションに渡します。例外が設定されている場合は、エラー・ナビゲーション・ターゲットが呼び出されます。

リストに表示される項目の処理時に発生するエラー

BPCListHandler クラスは、com.ibm.bpe.jsf.handler.ErrorHandler インターフェースをインプリメントします。 setErrors メソッドでタイプ java.util.Map のマップ・パラメーターを使用して、これらのエラーに関する情報を提供することができます。このマップには、キーとして ID が、値として例外が含まれています。 ID は、エラーの原因となったオブジェクトの getID メソッドによって戻された値である必要があります。マップが設定されていて、リスト内に表示されている項目のいずれかに ID のいずれかが一致する場合は、リスト・ハンドラーによって、エラー・メッセージを含む列が自動的にリストに追加されます。

リスト内のエラー・メッセージが古くなるのを避けるため、エラー・マップをリセットしてください。次の状況では、マップは自動的にリセットされます。

- refreshList メソッドの BPCListHandler クラスが呼び出される。
- BPCListHandler クラスで新規照会が設定されている。
- CommandBar コンポーネントを使用して、リストの項目でアクションがトリガーされている。 CommandBar コンポーネントは、エラー処理のメソッドの 1 つとしてこのメカニズムを使用します。

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 50. 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。
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 51. 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。

表 51. bpe:column 属性 (続き)

属性	必須	説明
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。

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 52. 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 53. 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。JSF アプリケーション内の CommandBar コンポーネントを追加および拡張することができます。

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インプリメントし、選択された Java オブジェクトを提供する必要があります。CommandBar コンポーネントは、通常、同一の JSP ファイル内の List コンポーネントまたは Details コンポーネントのいずれかと共に使用されます。一般に、タグで指定されたモデルは、同一ページの List コンポーネントまたは Details コンポーネントで指定されたモデルと同じです。そのため、例えば List コンポーネントの場合、コマンドはリスト内の選択された項目に対して作動します。

この例では、**model** 属性は TaskInstanceList 管理対象 Bean を参照します。この Bean は、選択されたオブジェクトをタスク・インスタンス・リストに示します。この Bean は ItemProvider インターフェースをインプリメントする必要があります。 このインターフェースは、BPCListHandler クラスおよび BPCDetailsHandler クラスによってインプリメントされます。

2. オプション: `bpe:commandbar` タグで参照されている管理対象 Bean を構成します。

   CommandBar **model** 属性が、例えばリスト・ハンドラーまたは詳細ハンドラー用に既に構成済みの管理対象 Bean を参照する場合、それ以上の構成は必要ありません。これらのハンドラーのいずれかの構成を変更する場合、または異なる管理対象 Bean を使用する場合は、ItemProvider インターフェースをインプリメントする管理対象 Bean を JSF 構成ファイルに追加してください。
3. カスタム・コマンドをインプリメントするコードを JSF アプリケーションに追加します。

   以下のコード断片は、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インプリメントするカスタム・コマンド・インプリメンテーションを参照します。つまり、コマンドは public String execute(List selectedObjects) throws ClientException メソッドをインプリメントする必要があります。コマンドが戻した結果は、JSF アプリケーションの次のナビゲーション規則を決定するために使用されます。

c. コマンドの完了後、CommandBar コンポーネントは **action** 属性を評価します。 **action** 属性は、静的ストリングである場合も、`public String Method()` というシグニチャーの JSF アクション・メソッドへのメソッド・バインディングである場合もあります。 **action** 属性を使用して、コマンド・クラスの結果をオーバーライドするか、またはナビゲーション規則の結果を明示的に指定します。コマンドが ErrorsInCommandException 例外以外の例外を生成した場合、**action** 属性は処理されません。

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。

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・バーをインプリメントする 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インプリメンテーションを示します。
2. CommandBar コンポーネントに関連するモデルが com.ibm.bpe.jsf.handler.ErrorHandler インターフェースをインプリメントすると、前のイベントからのエラー・メッセージを削除するため、clearErrorMap メソッドが呼び出されます。
3. ItemProvider インターフェースの getSelectedItems メソッドが呼び出されます。戻された項目のリストは、コマンドの execute メソッドに渡され、コマンドが呼び出されます。
4. CommandBar コンポーネントは、JavaServer Faces (JSF) ナビゲーション・ターゲットを決定します。 `bpe:commandbar` タグで **action** 属性が指定されていない場合は、execute メソッドの戻り値によってナビゲーション・ターゲットが指定されます。 **action** 属性が JSF メソッド・バインディングに設定されている場合は、メソッドによって戻されたストリングがナビゲーション・ターゲットと解釈されます。 **action** 属性は、明示的なナビゲーション・ターゲットも指定します。

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 54. 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インプリメントする管理対象 Bean に対する値バインディング式。通常、この管理対象 Bean は、同一の JavaServer Pages (JSP) ファイル内の List コンポーネントまたは Details コンポーネントによって CommandBar コンポーネントとして使用される com.ibm.bpe.jsf.handler.BPCListHandler クラスまたは com.ibm.bpe.jsf.handler.BPCDetailsHandler クラスです。
styleClass	いいえ	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 55. 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。

表 55. bpe:command 属性 (続き)

属性	必須	説明
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インプリメントする方法を示します。

```
public class MyHandler implements ItemListener {

  private TaskInstanceBean taskBean;
  private MessageWrapper inputMessage, outputMessage

  /* Listener method, e.g. when a task instance was selected in a list handler.
   * Ensure that the handler is registered in the faces-config.xml or manually.
   */
  public void itemChanged(Object item) {
    if( item instanceof TaskInstanceBean ) {
        taskBean = (TaskInstanceBean) item ;
    }
  }

  /* Get the input message wrapper
   */
  public MessageWrapper getInputMessage() {
    try{
        inputMessage = taskBean.getInputMessageWrapper() ;
    }
    catch( Exception e ) {
        ;       //...ignore errors for simplicity
    }
    return inputMessage;
  }

  /* Get the output message wrapper
   */
  public MessageWrapper getOutputMessage() {
    // Retrieve the message from the bean. If there is no message, create
```

```
    // one if the task has been claimed by the user. Ensure that only
    // potential owners or owners can manipulate the output message.
    try{
        outputMessage = taskBean.getOutputMessageWrapper();
        if( outputMessage == null
         && taskBean.getState() == TaskInstanceBean.STATE_CLAIMED ) {
            HumanTaskManagerService htm = getHumanTaskManagerService();
            outputMessage = new MessageWrapperImpl();
            outputMessage.setMessage(
               htm.createOutputMessage( taskBean.getID() ).getObject()
            );
         }
    }
    catch( Exception e ) {
        ;       //...ignore errors for simplicity
    }
    return outputMessage
  }
}
```

MyHandler 管理対象 Bean は、リスト・ハンドラーへの項目リスナーとして登録できるように、com.ibm.jsf.handler.ItemListener インターフェースをインプリメントします。ユーザーがリスト内の項目をクリックすると、選択された項目について MyHandler Bean が itemChanged(Object item) メソッドで通知されます。ハンドラーは、項目タイプを検査してから、関連した TaskInstanceBean オブジェクトへの参照を保管します。このインターフェースを使用するには、faces-config.xml ファイル内の適切なリスト・ハンドラーの 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。

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 56. 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。
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。

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。

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. APIEventHandlerPlugin3 インターフェースをインプリメントするクラス、または APIEventHandler インプリメンテーション・クラスを拡張するクラスを作成します。 このクラスは他のクラスのメソッドを呼び出すことができます。
   - APIEventHandlerPlugin3 インターフェースを使用する場合は、APIEventHandlerPlugin3 インターフェースおよび APIEventHandlerPlugin インターフェースのすべてのメソッドをインプリメントする必要があります。
   - APIEventHandler インプリメンテーション・クラスを拡張する場合は、必要なメソッドを上書きしてください。

   このクラスは、Java 2 Enterprise Edition (J2EE) Enterprise 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 2 サービス・プロバイダー・インターフェース仕様に準拠します。

   a. `com.ibm.task.spi.`*`plug-in_name`*`APIEventHandlerPlugin` という名前のファイルを作成します。 *plug-in_name* はプラグインの名前です。

      例えば、`Customer` という名前のプラグインが com.ibm.task.spi.APIEventHandlerPlugin3 インターフェースをインプリメントする場合、構成ファイルの名前は `com.ibm.task.spi.`**`Customer`**`APIEventHandlerPlugin` になります。

   b. このファイル内のコメント行 (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。

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・インプリメンテーションは同じ名前でなければなりません (例えば、SPI インプリメンテーションのイベント・ハンドラー名として `Customer` など)。

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。

関連概念

65 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。

API イベント・ハンドラー

ヒューマン・タスクが変更されたり、その状態が変わったりすると、API イベントが発生します。これらの API イベントを処理するために、タスクが変更される前 (pre-event メソッド)、および API 呼び出しが戻る直前 (post-event メソッド) に、イベント・ハンドラーが直接呼び出されます。

pre-event メソッドが ApplicationVetoException 例外をスローする場合、API アクションは実行されません。例外は API 呼び出し元に戻され、イベントに関連したトランザクションはロールバックされます。pre-event メソッドが内部イベントによって起動され、ApplicationVetoException 例外がスローされる場合、自動要求などの内部イベントは実行されませんが、例外はクライアント・アプリケーションに戻されません。この場合、情報メッセージが SystemOut.log ファイルに書き込まれます。API メソッドが処理中に例外をスローする場合、例外がキャッチされ、post-event メソッドに渡されます。 post-event メソッドが戻ると、例外は再び呼び出し元に渡されます。

以下の規則が、pre-event メソッドに適用されます。

- pre-event メソッドは、関連した API メソッドまたは内部イベントのパラメーターを受け取ります。
- pre-event メソッドは、処理が継続されないようにするため、ApplicationVetoException 例外をスローできます。

以下の規則が、post-event メソッドに適用されます。

- post-event メソッドは、API 呼び出しに提供されたパラメーター、および戻り値を受け取ります。例外が API メソッド・インプリメンテーションによってスローされる場合、post-event メソッドも例外を受け取ります。
- post-event メソッドは、戻り値を変更できません。

- post-event メソッドは例外をスローできません。実行時例外がログに記録されますが、無視されます。

API イベント・ハンドラーをインプリメントするには、APIEventHandlerPlugin インターフェースを拡張する 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 インプリメンテーション・クラスを拡張します。イベント・ハンドラーは、デフォルトのインプリメンテーション・クラスから継承される場合、常に最新バージョンの SPI をインプリメントします。より新しいバージョンの Business Process Choreographer にアップグレードすれば、新規 SPI メソッドを活用するために必要な変更が少なくてすみます。

通知イベント・ハンドラーと API イベント・ハンドラーの両方がある場合、イベント・ハンドラー名は 1 つしか登録できないので、両方のハンドラーの名前は同じでなければなりません。

通知イベント・ハンドラーの作成

ヒューマン・タスクがエスカレートされると、通知イベントが作成されます。Business Process Choreographer は、エスカレーション処理の機能 (エスカレーション作業項目の作成または E メールの送信など) を提供します。通知イベント・ハンドラーを作成し、エスカレーションが処理される方法をカスタマイズすることができます。

このタスクについて

通知イベント・ハンドラーをインプリメントするには、NotificationEventHandlerPlugin インターフェースをインプリメントする方法と、デフォルトの com.ibm.task.spi.NotificationEventHandler サービス・プロバイダー・インターフェース (SPI) インプリメンテーション・クラスを拡張する方法があります。

通知イベント・ハンドラーを作成するには、次のステップを実行します。

手順

1. NotificationEventHandlerPlugin インターフェースをインプリメントするクラス、または NotificationEventHandler インプリメンテーション・クラスを拡張するクラスを作成します。 このクラスは他のクラスのメソッドを呼び出すことができます。

   NotificationEventHandlerPlugin インターフェースを使用する場合は、すべてのインターフェース・メソッドをインプリメントする必要があります。SPI インプリメンテーション・クラスを拡張する場合は、必要なメソッドを上書きしてください。

   このクラスは、Java 2 Enterprise Edition (J2EE) Enterprise 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 J2EE アプリケーションで使用される場合、それらのクラスを別の JAR ファイルにパッケージして、共用ライブラリーとして登録することができます。

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 2 サービス・プロバイダー・インターフェース仕様に準拠します。

   a. `com.ibm.task.spi.`*`plug-in_name`*`NotificationEventHandlerPlugin` という名前のファイルを作成します。 *plug-in_name* はプラグインの名前です。

      例えば、`HelpDeskRequest` (イベント・ハンドラー名) という名前のプラグインが com.ibm.task.spi.NotificationEventHandlerPlugin インターフェースをインプリメントする場合、構成ファイルの名前は `com.ibm.task.spi.`**`HelpDeskRequest`**`NotificationEventHandlerPlugin` になります。

   b. このファイル内のコメント行 (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。

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・インプリメンテーションは同じ名前でなければなりません (例えば、SPI インプリメンテーションのイベント・ハンドラー名として `Customer` など)。

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。

関連概念

56 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。

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、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 2 Enterprise Edition (J2EE) アプリケーションでのみ使用されるか、複数のアプリケーションで使用されるかによって異なります。

プラグインをインストールするには、以下のいずれかのステップを実行します。

- 単一の J2EE アプリケーションで使用するプラグインをインストールする場合。

アプリケーション EAR ファイルにプラグイン JAR ファイルを追加します。WebSphere Integration Developer のデプロイメント記述子エディターで、プラグインの JAR ファイルを、主要な Enterprise JavaBeans (EJB) モジュールの J2EE アプリケーション用のプロジェクト・ユーティリティー JAR ファイルとしてインストールします。

- 複数の J2EE アプリケーションで使用するプラグインをインストールする場合。

  JAR ファイルを WebSphere Application Server 共用ライブラリーに入れ、そのライブラリーを、プラグインへのアクセスが必要なアプリケーションと関連付けます。JAR ファイルを 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。

臨時タスク
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。

- 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作成およびインストール

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戻します。担当者解決で戻される担当者照会の結果を変更するプラグインを作成することができます。例えば、ワークロード・バランシングを改善するために、既にワークロードが高いユーザーを照会結果から除去するプラグインを使用することがあります。

このタスクについて

後処理プラグインは 1 つしか所有できません。つまり、そのプラグインですべてのタスクの担当者照会結果を処理する必要があります。このプラグインでは、ユーザーの追加または除去、あるいはユーザーまたはグループ情報の変更ができます。また、結果タイプを、例えばユーザー・リストからグループに、あるいは全員に、変更することもできます。

プラグインは担当者解決の完了後に実行されるので、機密性またはセキュリティーを保持するための規則が既に適用されています。プラグインは、担当者の解決中に除去されたユーザーについての情報を受け取ります (HTM_REMOVED_USERS マップ・キーで渡されます)。プラグインがこのコンテキスト情報を使用して、機密性またはセキュリティー規則が保持されるようにしてください。

担当者照会結果の後処理を実装するには、StaffQueryResultPostProcessorPlugin インターフェースを使用します。このインターフェースには、タスク、エスカレーション、タスク・テンプレート、およびアプリケーション・コンポーネントの照会結果を変更するメソッドが含まれています。

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. StaffQueryResultPostProcessorPlugin インターフェースをインプリメントするクラスを作成します。

   すべてのインターフェース・メソッドをインプリメントする必要があります。このクラスは他のクラスのメソッドを呼び出すことができます。

   このクラスは、Java 2 Enterprise Edition (J2EE) Enterprise 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。

次の例では、SpecialTask というタスクのエディター・ロールを変更する方法を示しています。

```
public StaffQueryResult processStaffQueryResult
                        (StaffQueryResult originalStaffQueryResult,
                                          Task task,
                                          int  role,
                                          Map  context)
{
  StaffQueryResult newStaffQueryResult = originalStaffQueryResult;
  StaffQueryResultFactory staffResultFactory =
                         StaffQueryResultFactory.newInstance();
  if (role == com.ibm.task.api.WorkItem.REASON_EDITOR &&
      task.getName() != null &&
      task.getName().equals("SpecialTask"))
  {
      UserData user = staffResultFactory.newUserData
                           ("SuperEditor",
                             new Locale("en-US"),
                            "SuperEditor@company.com");
      ArrayList userList = new ArrayList();
      userList.add(user);

      newStaffQueryResult = staffResultFactory.newStaffQueryResult(userList);
   }
   return(newStaffQueryResult);
}
```

2. プラグイン・クラスとそのヘルパー・クラスを JAR ファイルにアセンブルします。

   JAR を共用ライブラリーとして使用できるようにし、それをタスク・コンテナーに関連付けることができます。このようにすると、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 2 サービス・プロバイダー・インターフェース仕様に準拠します。

   a. `com.ibm.task.spi.`*`plug-in_name`*`StaffQueryResultPostProcessorPlugin` という名前のファイルを作成します。*plug-in_name* はプラグインの名前です。

      例えば、`MyHandler` という名前のプラグインが com.ibm.task.spi.StaffQueryResultPostProcessorPlugin インターフェースをインプリメントする場合、構成ファイルの名前は `com.ibm.task.spi.`**`MyHandler`**`StaffQueryResultPostProcessorPlugin` 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、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。

4. プラグインをインストールします。

   担当者照会結果の後処理プラグインは 1 つしか所有できません。プラグインを共用ライブラリーとしてインストールする必要があります。

   a. プラグイン用の WebSphere Application Server 共用ライブラリーを定義します。 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共用ライブラリーを定義します。次に、この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。このステップは一度だけ実行する必要があります。

   b. 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。

5. プラグインを登録します。

   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。

      スタンドアロン環境の場合は、「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」 をクリックします。Business Process Choreographer がクラスター内に構成されている場合は、「**サーバー**」 → 「**クラスター**」 → 「***cluster_name***」 をクリックします。「**ビジネス・インテグレーション**」の下の「**Human Task Manager**」を選択します。「**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. **Staff.PostProcessorPlugin** という名前と、プラグインにつけた名前の値 (この例では、MyHandler) を持つカスタム・プロパティーを追加します。

結果

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。

JAR ファイルを変更した場合は、共用ライブラリー内のファイルを置き換えて、サーバーを再始動してください。

関連概念

103 ページの『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。

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

アプリケーションのインストール先のアプリケーション・サーバーまたはクラスターごとに、Business Flow Manager、Human Task Manager、またはその両方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。

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。

関連概念

632 ページの『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 またはサービス・デプロイメントによりプロセスまたはタスクのデプロイメント・コードが生成されるときに、各プロセス・コンポーネントまたはタスク・コンポーネントが 1 つのセッション・エンタープライズ Bean にマップされます。すべてのデプロイメント・コードは、エンタープライズ・アプリケーション (EAR) ファイルにパッケージ化されます。また、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、プロセスごとに、そのプロセスの Java コードを表す Java クラスが生成され、EAR ファイルに埋め込まれます。デプロイするモデルの新規バージョンごとに、新しいエンタープライズ・アプリケーションにパッケージ化する必要があります。

632 ページの『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。

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、System.out ファイルまたは SystemErr.log ファイルに報告されるか、あるいはこのサーバー・インスタンスで FFDC エントリーとして報告されます。

関連タスク

631 ページの『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。

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 またはサービス・デプロイメントによりプロセスまたはタスクのデプロイメント・コードが生成されるときに、各プロセス・コンポーネントまたはタスク・コンポーネントが 1 つのセッション・エンタープライズ Bean にマップされます。すべてのデプロイメント・コードは、エンタープライズ・アプリケーション (EAR) ファイルにパッケージ化されます。また、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、プロセスごとに、そのプロセスの Java コードを表す Java クラスが生成され、EAR ファイルに埋め込まれます。デプロイするモデルの新規バージョンごとに、新しいエンタープライズ・アプリケーションにパッケージ化する必要があります。

ビジネス・プロセスまたはヒューマン・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。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。

関連タスク

631 ページの『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。

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、以下の前提条件を満たしている必要があります。

- 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。このクラスター・メンバーは、Business Process Choreographer データベースへのアクセス権限を持っています。
- 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この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。
- アプリケーションに属するすべてのビジネス・プロセス・テンプレートおよびヒューマン・タスク・テンプレートが停止状態にある必要があります。
- 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。

開発環境および単体テスト環境として使用されているスタンドアロン・サーバー環境の場合、開発モードで実行するようサーバーを構成できます。この構成では、テンプレートを停止し、インスタンスが存在しないようにする必要はありません。ただし、この構成は実稼働環境に対しては無効です。

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. アプリケーションのすべてのプロセスおよびタスクのテンプレートを停止します。

   このアクションにより、プロセスおよびタスクのインスタンスの作成が回避されます。

   a. 管理コンソールのナビゲーション・ペインで、「**アプリケーション**」 → 「**SCA モジュール**」をクリックします。
   b. 停止するテンプレートを含んでいるモジュールを選択します。
   c. 「追加プロパティー」の下で、「**ビジネス・プロセス**」か「**ヒューマン・タスク**」、または両方を必要に応じてクリックします。
   d. 適切なチェック・ボックスをクリックして、プロセスおよびタスクのテンプレートをすべて選択します。
   e. 「**停止**」をクリックします。

   ビジネス・プロセス・テンプレートまたはタスク・テンプレートが含まれるすべての EJB モジュールで、このステップを繰り返します。
2. データベース、クラスターごとに少なくとも 1 つのアプリケーション・サーバー、アプリケーションがデプロイされているスタンドアロン・サーバーが動作していることを確認してください。

ネットワーク・デプロイメント環境では、デプロイメント・マネージャー、管理対象のすべてのスタンドアロン・アプリケーション・サーバー、および少なくとも 1 つのアプリケーション・サーバーが、そのアプリケーションがインストールされているクラスターごとに稼働している必要があります。

3. アプリケーションにビジネス・プロセス・インスタンスまたはヒューマン・タスク・インスタンスがないことを確認します。

   必要であれば、管理者は、Business Process Choreographer Explorer を使用して、残存するプロセス・インスタンスまたはタスク・インスタンスを削除することができます。

4. アプリケーションを停止してアンインストールするには、以下のようにします。
   a. 管理コンソールのナビゲーション・ペインで、「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」をクリックします。
   b. アンインストールするアプリケーションを選択し、「**停止**」をクリックします。

      アプリケーションでプロセス・インスタンスまたはタスク・インスタンスがまだ存在する場合は、このステップは失敗します。
   c. アンインストールするアプリケーションを再度選択し、「**アンインストール**」をクリックします。
   d. 「**保管**」をクリックして、変更を保管します。

結果

アプリケーションはアンインストールされます。

管理コマンドを使用した、ビジネス・プロセスおよびヒューマン・タスク・アプリケーションのアンインストール

管理コマンドには、ビジネス・プロセスまたはヒューマン・タスクを含むアプリケーションをアンインストールするための、管理コンソールの代替方法が用意されています。

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、以下の前提条件を満たしている必要があります。

- 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。このクラスター・メンバーは、Business Process Choreographer データベースへのアクセス権限を持っています。
- 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この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。

- アプリケーションに属するすべてのビジネス・プロセス・テンプレートおよびヒューマン・タスク・テンプレートが停止状態にある必要があります。
- 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。

開発環境および単体テスト環境として使用されているスタンドアロン・サーバー環境の場合、開発モードで実行するようサーバーを構成できます。この構成では、テンプレートを停止し、インスタンスが存在しないようにする必要はありません。ただし、この構成は実稼働環境に対しては無効です。

さらに、グローバル・セキュリティーが使用可能に設定されている場合は、使用するユーザー ID にオペレーター権限があることを確認します。

管理クライアントが接続しているサーバー・プロセスが動作していることを確認します。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。

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。また、アプリケーションをアンインストールするには、そのアプリケーションに属しているテンプレートを停止する必要があります。 bpcTemplates.jacl スクリプトを使用して、1 つの手順でテンプレートを停止およびアンインストールできます。

アプリケーションをアンインストールする前に、 Business Process Choreographer Explorer などを使用して、アプリケーション内のテンプレートに関連したプロセス・インスタンスやタスク・インスタンスを削除できます。bpcTemplates.jacl スクリプトと一緒に **-force** オプションを使用して、テンプレートと関連したインスタンスの削除、テンプレートの停止、テンプレートのアンインストールを 1 ステップで行うこともできます。

注意:
-force オプションはすべてのプロセス・インスタンスおよびタスク・インスタンスのデータを削除するため、注意して使用する必要があります。

手順

1. Business Process Choreographer サンプル・ディレクトリーに移動します。

   Windows プラットフォームの場合は、次のように入力します。

   ```
   cd install_root¥ProcessChoreographer¥admin
   ```

   Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

   ```
   cd install_root/ProcessChoreographer/admin
   ```

2. テンプレートを停止して、対応するアプリケーションをアンインストールします。

   Windows プラットフォームの場合は、次のように入力します。

```
install_root¥bin¥wsadmin -f bpcTemplates.jacl
                            [-user user_name]
                            [-password user password]
                            -uninstall application_name
                            [-force]
```

Linux、UNIX、および i5/OS プラットフォームの場合は、次のように入力します。

```
install_root/bin/wsadmin -f bpcTemplates.jacl
                            [-user user_name]
                            [-password user password]
                            -uninstall application_name
                            [-force]
```

各部の意味は、次のとおりです。

user_name
グローバル・セキュリティーが使用可能な場合は、認証用のユーザー ID を提供します。

user_password
グローバル・セキュリティーが使用可能な場合は、認証用のユーザー・パスワードを提供します。

application_name
グローバル・セキュリティーが使用可能な場合は、認証用のユーザー・パスワードを提供します。

結果

アプリケーションはアンインストールされます。

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・イベントのモニター

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。

Business Process Choreographer バージョン 6.1 では、2 つのイベント形式が使用されることがあります。

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プロセスがある場合、または WebSphere Integration Developer 6.1 で WebSphere Business Monitor 6.0.2 形式モード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 でモデリングされたプロセスがあり、WebSphere Business Monitor 6.1 形式モード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。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、XML スキーマ定義 (XSD) ファイル BFMEvents.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。

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 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.BASE』
- 647 ページの『BPC.BFM.PROCESS.BASE』
- 648 ページの『BPC.BFM.PROCESS.STATUS』
- 648 ページの『BPC.BFM.PROCESS.START』
- 648 ページの『BPC.BFM.PROCESS.FAILURE』
- 648 ページの『BPC.BFM.PROCESS.CORREL』
- 648 ページの『BPC.BFM.PROCESS.WISTATUS』
- 649 ページの『BPC.BFM.PROCESS.WITRANSFER』
- 649 ページの『BPC.BFM.PROCESS.ESCALATED』
- 649 ページの『BPC.BFM.PROCESS.EVENT』
- 650 ページの『BPC.BFM.PROCESS.PARTNER』
- 650 ページの『BPC.BFM.PROCESS.CUSTOMPROPERTYSET』
- 651 ページの『BPC.BFM.ACTIVITY.BASE』
- 652 ページの『BPC.BFM.ACTIVITY.STATUS』
- 652 ページの『BPC.BFM.ACTIVITY.FAILURE』
- 653 ページの『BPC.BFM.ACTIVITY.MESSAGE』
- 653 ページの『BPC.BFM.ACTIVITY.CLAIM』
- 653 ページの『BPC.BFM.ACTIVITY.WISTATUS』
- 653 ページの『BPC.BFM.ACTIVITY.WITRANSFER』
- 654 ページの『BPC.BFM.ACTIVITY.FOREACH』
- 654 ページの『BPC.BFM.ACTIVITY.ESCALATED』
- 654 ページの『BPC.BFM.ACTIVITY.EVENT』
- 654 ページの『BPC.BFM.ACTIVITY.CUSTOMPROPERTYSET』
- 655 ページの『BPC.BFM.LINK.STATUS』
- 655 ページの『BPC.BFM.VARIABLE.STATUS』

BPC.BFM.BASE

BPC.BFM.BASE は、WBIMonitoringEvent の XML エレメントを継承します。

表 57. 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。

表 57. *BPC.BFM.BASE の XML エレメント (続き)*

XML エレメント	説明
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。

BPC.BFM.PROCESS.BASE

BPC.BFM.PROCESS.BASE は、 646 ページの『BPC.BFM.BASE』の XML エレメントを継承します。

表 58. *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 10 - STATE_INDOUBT 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。

表 58. BPC.BFM.PROCESS.BASE の XML エレメント (続き)

XML エレメント	説明
principal	このイベントに関連したユーザーの名前。

BPC.BFM.PROCESS.STATUS

BPC.BFM.PROCESS.STATUS は、 647 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

BPC.BFM.PROCESS.START

BPC.BFM.PROCESS.START は、 647 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 59. 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.FAILURE

BPC.BFM.PROCESS.FAILURE は、 647 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 60. 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。

BPC.BFM.PROCESS.CORREL

BPC.BFM.PROCESS.CORREL は、 647 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 61. BPC.BFM.PROCESS.CORREL の XML エレメント

XML エレメント	説明

correlationSet	相関セット・インスタンス。形式は次のとおりです。
	`<?xml version="1.0"?>`
	`<correlationSet name="correlation set name">`
	`  <property name="property name" value="property value"/>*`
	`</correlationSet>`

BPC.BFM.PROCESS.WISTATUS

BPC.BFM.PROCESS.WISTATUS は、 647 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 62. *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。

BPC.BFM.PROCESS.WITRANSFER

BPC.BFM.PROCESS.WITRANSFER は、 647 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 63. *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。

BPC.BFM.PROCESS.ESCALATED

BPC.BFM.PROCESS.ESCALATED は、 647 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 64. *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・ネーム・スペースです。

BPC.BFM.PROCESS.EVENT

BPC.BFM.PROCESS.EVENT は、 647 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 65. *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 (Event Monitor)**」タブで「**互換性のあるイベントのモニター (Monitor Compatible Events)**」オプションが選択されたかどうかによって異なります。 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。
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・ネーム・スペースです。

BPC.BFM.PROCESS.PARTNER

BPC.BFM.PROCESS.PARTNER は、 647 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 66. *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。

BPC.BFM.PROCESS.CUSTOMPROPERTYSET

BPC.BFM.PROCESS.CUSTOMPROPERTYSET は、 647 ページの『BPC.BFM.PROCESS.BASE』の XML エレメントを継承します。

表 67. *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。

BPC.BFM.ACTIVITY.BASE

BPC.BFM.ACTIVITY.BASE は、 646 ページの『BPC.BFM.BASE』の XML エレメントを継承します。

表 68. 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 48 - KIND_FOR_EACH_PARALLEL 1000 - SQLSnippet 1001 - RetrieveSet 1002 - InvokeInformationService 1003 - AtomicSQLSnippetSequence

表 68. *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>*。この属性の値は、次の値のいずれかになります。 1 - STATE_INACTIVE 2 - STATE_READY 3 - STATE_RUNNING 4 - STATE_SKIPPED 5 - STATE_FINISHED 6 - STATE_FAILED 7 - STATE_TERMINATED 8 - STATE_CLAIMED 9 - STATE_TERMINATING 10 - STATE_FAILING 11 - STATE_WAITING 12 - STATE_EXPIRED 13 - STATE_STOPPED
bpelId	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	アクティビティーを要求したユーザーの名前。

BPC.BFM.ACTIVITY.STATUS

BPC.BFM.ACTIVITY.STATUS は、 651 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

BPC.BFM.ACTIVITY.FAILURE

BPC.BFM.ACTIVITY.FAILURE は、 651 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 69. *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

BPC.BFM.ACTIVITY.MESSAGE

BPC.BFM.ACTIVITY.MESSAGE は、 651 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 70. 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 (Event Monitor)**」タブで「**互換性のあるイベントのモニター (Monitor Compatible Events)**」オプションが選択されたかどうかによって異なります。 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.CLAIM

BPC.BFM.ACTIVITY.CLAIM は、 651 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 71. 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.WISTATUS

BPC.BFM.ACTIVITY.WISTATUS は、 651 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 72. 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。

BPC.BFM.ACTIVITY.WITRANSFER

BPC.BFM.ACTIVITY.WITRANSFER は、 651 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 73. *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。

BPC.BFM.ACTIVITY.FOREACH

BPC.BFM.ACTIVITY.FOREACH は、 651 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 74. *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.ESCALATED

BPC.BFM.ACTIVITY.ESCALATED は、 651 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 75. *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、 651 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 76. *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.CUSTOMPROPERTYSET

BPC.BFM.ACTIVITY.CUSTOMPROPERTYSET は、 651 ページの『BPC.BFM.ACTIVITY.BASE』の XML エレメントを継承します。

表 77. *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。

表 77. *BPC.BFM.ACTIVITY.CUSTOMPROPERTYSET の XML エレメント (続き)*

XML エレメント	説明
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。

BPC.BFM.LINK.STATUS

BPC.BFM.LINK.STATUS は、 646 ページの『BPC.BFM.BASE』の XML エレメントを継承します。

表 78. *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。

BPC.BFM.VARIABLE.STATUS

BPC.BFM.VARIABLE.STATUS は、 646 ページの『BPC.BFM.BASE』の XML エレメントを継承します。

表 79. *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 (Event Monitor)**」タブで「**互換性のあるイベントのモニター (Monitor Compatible Events)**」オプションが選択されたかどうかによって異なります。 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。

表 79. *BPC.BFM.VARIABLE.STATUS* の *XML* エレメント *(続き)*

XML エレメント	説明
principal	変数を更新したユーザーの名前。

ビジネス・プロセス・イベント

WebSphere Integration Developer 内のビジネス・プロセス・エレメントのモニターが要求された場合、ビジネス・プロセス・イベントが送信されます。ここでは、ビジネス・プロセスによって発行可能なすべてのイベントのリストを示します。

プロセスまたはアクティビティーの状態が変更された時点でイベントが発行されます。ビジネス・プロセスによって発生するイベントには以下のタイプがあります。

- 657 ページの『プロセス・イベント』
- 659 ページの『アクティビティー・イベント』
- 664 ページの『アクティビティー・スコープ・イベント』
- 664 ページの『リンク・イベント』
- 665 ページの『変数イベント』

XML スキーマ定義 (XSD) ファイル

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。

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 *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。

拡張子名

extensionName には、Common Base Event に含まれているイベント固有情報を定義す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。拡張子名についての詳細は、645 ページの『ビジネス・プロセス・イベントの拡張子名』を参照してください。

状態 ビジネス・プロセス・イベントの状態名を指します。状態の詳細については、665 ページの『ビジネス・プロセス・イベントの状態』を参照してください。

イベント性質

WebSphere Integration Developer に表示されるときの、EventNature パラメーター内のビジネス・プロセス・エレメントのイベント状態を指すポインター。

プロセス・イベント

以下の表に、すべてのプロセス・イベントについてまとめます。

コード	拡張子名	状態	イベント性質	説明
21000	BPC.BFM.PROCESS.START	開始	ENTRY	プロセスが開始された
21001	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	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	BPC.BFM.PROCESS.STATUS	停止	EXIT	プロセスが完了した
21005	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	BPC.BFM.PROCESS.START	レポート	RESTARTED	プロセスが再始動した
21020	BPC.BFM.PROCESS.STATUS	破棄	DELETED	プロセスが削除された
42001	BPC.BFM.PROCESS. FAILURE	失敗	FAILED	プロセスが失敗した

コード	拡張子名	状態	イベント性質	説明
42003	BPC.BFM.PROCESS.STATUS	レポート	COMPENSATING	プロセスが補正中。プロセス・インスタンスを補正するには、Business Process Choreographer Explorer を使用します。
42004	BPC.BFM.PROCESS.STATUS	停止	COMPENSATED	プロセスが補正された
42009	BPC.BFM.PROCESS.STATUS	レポート	TERMINATING	プロセスが強制終了中
42010	BPC.BFM.PROCESS.STATUS	レポート	FAILING	プロセスが失敗する
42027	BPC.BFM.PROCESS.CORREL	レポート	CORRELATION	相関セットが初期化された。プロセス・インスタンスの新規相関セットが初期化された時点で発行されます。これは、例えば開始相関セットがある受信アクティビティーがメッセージを受信した場合などに該当します。
42041	BPC.BFM.PROCESS. WISTATUS	レポート	WI_DELETED	プロセス作業項目が削除された
42042	BPC.BFM.PROCESS. WISTATUS	レポート	WI_CREATED	プロセス作業項目が作成された
42046	BPC.BFM.PROCESS.STATUS	失敗	COMPFAILED	プロセス補正が失敗した
42047	BPC.BFM.PROCESS.EVENT	レポート	EV_RECEIVED	プロセス・イベントを受信した。イベントを定義するにはプロセス・インターフェースを使用します。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、このイベントが生成されます。

コード	拡張子名	状態	イベント性質	説明
42049	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、インライン呼び出しタスクがエスカレートされると生成されます。このイベントはプロセス・レベルで定義されており、onEvent イベント・ハンドラーに関連付けられています。
42056	BPC.BFM.PROCESS. WITRANSFER	レポート	WI_TRANSFERRED	プロセス作業項目が転送された
42058	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た時点で生成されます。
42059	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たときに生成されます。

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。

アクティビティー・イベント

以下の表で、すべてのアクティビティー・イベントについて説明します。

コード	拡張子名	状態	イベント性質	説明
21006	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、human task アクティビティーの開始時に生成されます。
21007	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	invoke、human task、receive、および reply の各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、human task、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	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、human task アクティビティーに対する要求が取り消された時点で生成されます。
21022	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、human task アクティビティーが要求された時点で生成されます。
21027	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。

コード	拡張子名	状態	イベント性質	説明
21080	BPC.BFM.ACTIVITY.FAILURE	失敗	FAILED	アクティビティーが失敗した
21081	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。invoke アクティビティーとインライン・ヒューマン・タスク・アクティビティーに有効期限を定義できます。
42005	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。イベントは、結合動作定義されているアクティビティーにのみ適用できます。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	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。
42013	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。
42015	BPC.BFM.ACTIVITY.STATUS	停止	STOPPED	アクティビティーが停止した

コード	拡張子名	状態	イベント性質	説明
42031	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	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	BPC.BFM.ACTIVITY.MESSAGE	レポート	EXIT	アクティビティーがメッセージを受信
42037	BPC.BFM.ACTIVITY.STATUS	レポート	CONDTRUE	ループ条件が true
42038	BPC.BFM.ACTIVITY.STATUS	レポート	CONDFALSE	ループ条件が false
42039	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、インライン・ヒューマン・タスク、および受信イベントにのみ適用できます。
42040	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、インライン・ヒューマン・タスク、および受信イベントにのみ適用できます。

コード	拡張子名	状態	イベント性質	説明
42050	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、pick、インライン・ヒューマン・タスク、および受信イベントにのみ適用できます。
42054	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、インライン・ヒューマン・タスク、および受信イベントにのみ適用できます。
42055	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、インライン・ヒューマン・タスク、および受信イベントにのみ適用できます。
42057	BPC.BFM.ACTIVITY. FOREACH	レポート	BRANCHES_STARTED	For each - アクティビティー分岐が開始された
42060	BPC.BFM.ACTIVITY. CUSTOMPROPERTYSET	レポート	CP_SET	アクティビティーのカスタム・プロパティーが設定された。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ときに生成されます。

アクティビティー・イベントの場合、以下のイベント相関範囲 ID 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。

アクティビティー・スコープ・イベント

以下の表で、すべてのアクティビティー・スコープ・イベントについて説明します。

コード	拡張子名	状態	イベント性質	説明
42020	BPC.BFM.ACTIVITY.STATUS	開始	ENTRY	スコープが開始した
42021	BPC.BFM.ACTIVITY.STATUS	レポート	SKIPPED	スコープがスキップされた
42022	BPC.BFM.ACTIVITY.FAILURE	失敗	FAILED	スコープが失敗した
42023	BPC.BFM.ACTIVITY.STATUS	レポート	FAILING	スコープが強制終了中
42024	BPC.BFM.ACTIVITY.STATUS	停止	TERMINATED	スコープが強制終了した
42026	BPC.BFM.ACTIVITY.STATUS	停止	EXIT	スコープが完了した
42043	BPC.BFM.ACTIVITY.STATUS	レポート	COMPENSATING	スコープが補正中
42044	BPC.BFM.ACTIVITY.STATUS	停止	COMPENSATED	スコープが補正された
42045	BPC.BFM.ACTIVITY.STATUS	失敗	COMPFAILED	スコープ補正が失敗した
42048	BPC.BFM.ACTIVITY.EVENT	レポート	EV_RECEIVED	アクティビティー・イベントを受信
42051	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

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。

リンク・イベント

以下の表は、すべてのリンク・イベントについて説明しています。

コード	拡張子名	状態	イベント性質	説明
21034	BPC.BFM.LINK.STATUS	レポート	CONDTRUE	リンクが true と評価された
42000	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。

変数イベント

以下の表で、変数イベントについて説明します。

コード	拡張子名	状態	イベント性質	説明
21090	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。

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`

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

第 20 章 ヒューマン・タスク・イベントのモニター

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。

Business Process Choreographer バージョン 6.1 では、2 つのイベント形式が使用されることがあります。

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 で WebSphere Business Monitor 6.0.2 形式モード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 でモデリングされたタスクがあり、WebSphere Business Monitor 6.1 形式モード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。

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.TASK.BASE』
- 『BPC.HTM.TASK.STATUS』
- 669 ページの『BPC.HTM.TASK.FOLLOW』
- 669 ページの『BPC.HTM.TASK.MESSAGE』
- 669 ページの『BPC.HTM.TASK.INTERACT』
- 669 ページの『BPC.HTM.TASK.FAILURE』
- 670 ページの『BPC.HTM.TASK.WISTATUS』
- 670 ページの『BPC.HTM.TASK.WITRANSFER』
- 670 ページの『BPC.HTM.TASK.CUSTOMPROPERTYSET』
- 670 ページの『BPC.HTM.ESCALATION.BASE』
- 671 ページの『BPC.HTM.ESCALATION.STATUS』
- 671 ページの『BPC.HTM.ESCALATION.WISTATUS』
- 671 ページの『BPC.HTM.ESCALATION.WITRANSFER』
- 671 ページの『BPC.HTM.ESCALATION.CUSTOMPROPERTYSET』

BPC.HTM.BASE

BPC.HTM.BASE は、WBIMonitoringEvent の XML エレメントを継承します。

表 80. 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。
taskTemplateId	テンプレートの ID。
taskTemplateName	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。
taskTemplateValidFrom	タスク・テンプレートが有効になる日時。

BPC.HTM.TASK.BASE

BPC.HTM.TASK.BASE は、『BPC.HTM.BASE』の XML エレメントを継承します。

表 81. 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。

BPC.HTM.TASK.STATUS

BPC.HTM.TASK.STATUS は、『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

BPC.HTM.TASK.FOLLOW

BPC.HTM.TASK.FOLLOW は、 668 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 82. 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.MESSAGE

BPC.HTM.TASK.MESSAGE は、 668 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 83. 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 (Event Monitor)**」タブで「**互換性のあるイベントのモニター (Monitor Compatible Events)**」オプションが選択されたかどうかによって異なります。 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.HTM.TASK.INTERACT

BPC.HTM.TASK.INTERACT は、 668 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 84. 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.FAILURE

BPC.HTM.TASK.FAILURE は、 668 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 85. *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。

BPC.HTM.TASK.WISTATUS

BPC.HTM.TASK.WISTATUS は、 668 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 86. *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。

BPC.HTM.TASK.WITRANSFER

BPC.HTM.TASK.WITRANSFER は、 668 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 87. *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。

BPC.HTM.TASK.CUSTOMPROPERTYSET

BPC.HTM.TASK.CUSTOMPROPERTYSET は、 668 ページの『BPC.HTM.TASK.BASE』の XML エレメントを継承します。

表 88. *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。

BPC.HTM.ESCALATION.BASE

BPC.HTM.ESCALATION.BASE は、 668 ページの『BPC.HTM.BASE』の XML エレメントを継承します。

表 89. *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。

BPC.HTM.ESCALATION.STATUS

BPC.HTM.ESCALATION.STATUS は、 670 ページの『BPC.HTM.ESCALATION.BASE』の XML エレメントを継承します。

BPC.HTM.ESCALATION.WISTATUS

BPC.HTM.ESCALATION.WISTATUS は、 670 ページの『BPC.HTM.ESCALATION.BASE』の XML エレメントを継承します。

表 90. *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。

BPC.HTM.ESCALATION.WITRANSFER

BPC.HTM.ESCALATION.WITRANSFER は、 670 ページの『BPC.HTM.ESCALATION.BASE』の XML エレメントを継承します。

表 91. *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。

BPC.HTM.ESCALATION.CUSTOMPROPERTYSET

BPC.HTM.ESCALATION.CUSTOMPROPERTYSET は、 670 ページの『BPC.HTM.ESCALATION.BASE』の XML エレメントを継承します。

表 92. *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。

ヒューマン・タスク・イベント

WebSphere Integration Developer 内のタスクのエレメントについてモニターが要求された場合、ヒューマン・タスク・イベントが送信されます。ここでは、ヒューマン・タスクによって発行可能なすべてのイベントのリストを示します。

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。

- 673 ページの『タスク・イベント』
- 674 ページの『エスカレーション・イベント』

注: タスク・モデルにおいてビジネス関連性フラグが true に設定されている場合、イベントは随時タスクの場合のみ発行されます。

インライン・タスクのイベントはアクティビティー・イベントとして発行されます。これらのイベントのリストについては、656 ページの『ビジネス・プロセス・イベント』を参照してください。

XML スキーマ定義 (XSD) ファイル

イベントの構造は、XML スキーマ定義 (XSD) ファイル HTMEvents.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。

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。

拡張子名

Common Base Event の extensionName 属性の値として使用されるストリング値。

WebSphere Business Integration Modeler を使用して基本のタスク・モデルを作成する場合、有効搭載量にメッセージ・データを含むイベントの拡張名は、ハッシュ文字 (#) とそれに続く追加文字によって拡張できます。これらの追加文字は、いろいろなメッセージ・オブジェクトを運ぶ Common Base Events の識別に使用されます。メッセージ・データを出力するイベントにも、データ・オブジェクトの内容を報告するために、追加のネストされた extendedDataElements が含まれます。詳しくは、WebSphere Business Integration Modeler の資料を参照してください。

状態 ヒューマン・タスク・イベントの状態名を指します。状態の詳細については、675 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	拡張子名	状態	イベント性質	説明
51001	BPC.HTM.TASK. INTERACT	レポート	CREATED	タスクが作成された
51002	BPC.HTM.TASK.STATUS	破棄	DELETED	タスクが削除された
51003	BPC.HTM.TASK.STATUS	開始	ENTRY	タスクが開始された
51004	BPC.HTM.TASK.STATUS	停止	EXIT	タスクが完了した
51005	BPC.HTM.TASK.STATUS	レポート	DEASSIGNED	要求がキャンセルされた
51006	BPC.HTM.TASK. INTERACT	レポート	ASSIGNED	タスクが要求された
51007	BPC.HTM.TASK.STATUS	停止	TERMINATED	タスクが強制終了された
51008	BPC.HTM.TASK. FAILURE	失敗	FAILED	タスクが失敗した
51009	BPC.HTM.TASK.STATUS	レポート	EXPIRED	タスクの期限切れ
51010	BPC.HTM.TASK.STATUS	レポート	WAITFORSUBTASK	サブタスクを待機中
51011	BPC.HTM.TASK.STATUS	停止	SUBTASKCOMPLETED	サブタスクが完了した
51012	BPC.HTM.TASK.STATUS	レポート	RESTARTED	タスクが再始動した
51013	BPC.HTM.TASK.STATUS	レポート	SUSPENDED	タスクが一時停止した
51014	BPC.HTM.TASK.STATUS	レポート	RESUMED	タスクが再開した
51015	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
51101	BPC.HTM.TASK.STATUS	レポート	UPDATED	タスク・プロパティーが更新された
51103	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	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	BPC.HTM.TASK. WISTATUS	破棄	WI_DELETED	作業項目が削除された
51202	BPC.HTM.TASK. WISTATUS	レポート	WI_CREATED	作業項目が作成された

コード	拡張子名	状態	イベント性質	説明
51204	BPC.HTM.TASK. WITRANSFER	レポート	WI_TRANSFERRED	作業項目が転送された
51205	BPC.HTM.TASK. WISTATUS	レポート	WI_REFRESHED	作業項目が更新された
51301	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。

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	拡張子名	状態	イベント性質	説明
53001	BPC.HTM.ESCALATION. STATUS	レポート	ENTRY	エスカレーションが起動された
53201	BPC.HTM.ESCALATION. WISTATUS	破棄	WI_DELETED	作業項目が削除された
53202	BPC.HTM.ESCALATION. WISTATUS	レポート	WI_CREATED	作業項目が作成された
53204	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	BPC.HTM.ESCALATION. WISTATUS	レポート	WI_REFRESH-ED	作業項目が更新された
51302	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。

ヒューマン・タスク・イベントの状態

ヒューマン・タスク・イベントは、さまざまな状態で発行可能です。これらの状態のデータについては、状態エレメントで説明されています。

ヒューマン・タスク・イベントには、次の状態エレメントのいずれかが含まれます。

状態名	Common Base Event の内容	
開始	categoryName は StartSituation に設定されます。	
	situationType	
	Type	StartSituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
	situationQualifier	START_COMPLETED
停止	categoryName は StopSituation に設定されます。	
	situationType	
	Type	StopSituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
	situationQualifier	STOP_COMPLETED
破棄	categoryName は DestroySituation に設定されます。	
	situationType	
	Type	DestroySituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
失敗	categoryName は StopSituation に設定されます。	
	situationType	
	Type	StopSituation
	reasoningScope	EXTERNAL
	successDisposition	UNSUCCESSFUL
	situationQualifier	STOP_COMPLETED
レポート	categoryName は ReportSituation に設定されます。	
	situationType	
	Type	ReportSituation
	reasoningScope	EXTERNAL
	reportCategory	STATUS

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・ユーザーの応答時間を減らす方法を好むものもあります。その他のアプリケーションの場合は、システムで可能なトランザクションの処理速度のほうが、各トランザクションの実際の期間よりも重要になります。

2. 基本的な測定を行います。

   アプリケーションのチューニングに適した負荷、時刻、および曜日の条件下で、基本的な測定を行います。通常、最も重要な基本的な測定値は、スループットおよび応答時間です。例えば CPU 負荷が 100% に、ディスク I/O が最大値に、またはネットワーク I/O が 100% に達するなど、特定のボトルネック容量に達した後で、スループット値が測定されます。信頼性の高い応答時間の値は、単一のプロセス・インスタンスで、サーバー使用率が低いときに、最も適切に測定されます。

3. 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。なぜなら、Microflow は長期実行プロセスより快適に実行するからです。パーシスタンスが不要であり、単一スレッドを 1 つのトランザクションで処理できる機能がある場合、モデル化 Microflow を使用してください。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。データの直列化および非直列化にも費用がかかるため、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、プロセスから流れるデータのサイズおよび複雑さを最小化することも検討してください。さらに、ロギング用に発行されるイベントの数を最小化します。

4. プロセスを調整します。

   アプリケーションが長期間にわたって実行するプロセスと microflow のどちらを使用するかに応じて、以下のステップを実行します。

- 長期間にわたって実行するプロセスを調整するには、『長期間にわたって実行するプロセスのチューニング』で説明されているステップを実行します。これらのプロセスは、長時間実行が継続する傾向がありますが、イベントまたは人との対話で中断される場合もあります。したがってそのパフォーマンスは、Business Process Choreographer データベースとメッセージング・サービスのパフォーマンスによって異なります。
- microflow を調整するには、 695 ページの『microflow のチューニング』で説明されているステップを実行します。このプロセスは、どちらかといえば短期間のみ実行するプロセスです。このプロセスは、監査ロギング用のみにデータベースを使用して (監査ロギングが使用可能な場合)、テンプレート情報を取得します。このプロセスでは、永続データの保管のためにメッセージング・サポートが使用されません。このプロセスは、人との対話を必要としません。

5. 除去可能なパフォーマンスのボトルネックがないかどうか、現在の構成を検討します。

   考慮するべき可能性のある項目は以下のとおりです。
   - より多くのプロセッサー、メモリー、およびより高速なディスクをインストールします。
   - データベースのログをデータとは異なる物理ディスクに保管して、データをいくつかのディスクに分散させます。
   - 最適なパフォーマンスを得るために、Cloudscape ではなく、DB2を使用します。
6. 基本的な測定項目に関して、同様の負荷条件でベンチマーク測定を繰り返します。

   アプリケーションのパフォーマンス測定の永続レコードを保持して、パフォーマンスにおける将来の変更点を客観的に測定します。

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

このタスクを使用して、長期間にわたって実行するビジネス・プロセスのパフォーマンスを改善します。

このタスクについて

長期間にわたって実行するプロセスには、ユーザーとの対話、非同期呼び出し、複数の受信、ピック、イベント・ハンドラーなどがあります。これらのプロセスでは、永続状態を保管するためにデータベースとメッセージング・サブシステムを使用します。以下のトピックでは、長期間にわたって実行するプロセスのパフォーマンスを向上させる方法を説明します。

ハードウェア・リソースの平衡化

ハードウェア・リソースを平衡化することにより、長期実行ビジネス・プロセスのパフォーマンスを改善します。

このタスクについて

システムの調整を始める前に、使用しているコンピューターが平衡の取れた状態になっていること、すなわち、使用可能なリソース (CPU、メモリー、および I/O) が適正な比率に保たれていること、を検証してください。(1 つまたは多数の) 非常に高速な CPU と、少ないメモリー、貧弱な I/O パフォーマンスを持つコンピューターは、調整しにくいものです。割り込み可能プロセスにおいては、適切な処理能力および十分なメモリーと並んで、複数の高速ディスク・ドライブによって提供される高い I/O パフォーマンスが重要です。

実動システムでは、DBMS の実行用とアプリケーション・サーバーの実行用に別個のマシンを使用して、データベースをアプリケーション・サーバーと分離することをお勧めします。高負荷または高可用性の構成の場合、ビジネス・プロセス実行用の複数のマシン上にある WebSphere クラスターと、それとは別個のデータベース用マシンを使用することを検討してください。

手順

1. データベース・マシンで、十分なディスクが割り振られるようにします。
2. 十分なメモリーを割り振ります。

   割り振るメモリーの量は、プラットフォームによって異なります。

   - 4 GB の物理メモリーと、ローカル・データベース管理システムを装備する 32 ビットの Windows システムの場合は、以下の設定を使用してメモリーを割り振ります。
     - Windows システム用に 512 MB
     - WebSphere Application Server 用に 768 MB
     - DB2 を使用する場合、データベース用に 1.5 GB。 Oracle を使用する場合、システム・グローバル域 (SGA) 用に 1 GB 以内、プログラム・グローバル域 (PGA) 用に 500 MB。
   - 8 GB の物理メモリーと、ローカル・データベース管理システムを装備する 64 ビットの AIX システムの場合は、以下の設定を使用してメモリーを割り振ります。
     - AIX システム用に 512 MB
     - WebSphere Application Server 用に 1024 MB
     - データベース用に 5 GB。プロセス・データベースに 4 GB、メッセージング・データベースまたはデータベースに 1 GB を割り振ります。

       ヒント: ファイルのキャッシュなどでもメモリーを消費するため、最適なパフォーマンスを確保するためには、データベースにすべてのメモリーを割り当てないでください。使用可能なメモリーが不足するため、データをディスクにスワップしなければならないような状態は避けてください。

- i5/OS システムの場合、Work with System Status (WRKSYSSTS) コマンドを使用して、システムがメモリーのページングを行わないようにしてください。大量のページ不在が発生する場合、以下のアクションの 1 つまたは複数を実行してください。
  a. WebSphere Process Server または Enterprise Service Bus サーバー・サブシステム・メモリー・プールに使用可能なメモリーを増やします。
  b. WebSphere Process Server または Enterprise Service Bus サーバーを別のメモリー・プールに移動させます。
  c. ジョブを WebSphere Process Server または Enterprise Service Bus サーバー・サブシステム・メモリー・プールから除去します。
- アプリケーション・サーバーのヒープ・サイズを細密チューニングします。

  注: アプリケーション・サーバーが i5/OS システムで稼働する場合、ヒープ・サイズは調整できません。

3. ネットワーク使用率を監視します。 アプリケーションのパフォーマンスは、メッセージがサーバーとデータベース・サーバー間を通過する速度によって異なります。可能であれば、ネットワークの遅延を削減します。
4. ワークロードを他のサーバーに移動します。

   他のサーバーに移動可能なアプリケーションまたはサブシステムについて考慮します。

結果

これで、ご使用のコンピューターが平衡の取れた状態になりました。

関連タスク

132 ページの『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。

688 ページの『アプリケーション・サーバーのチューニング』
このタスクを使用して、アプリケーション・サーバーを調整します。

初期 DB2 データベース設定の指定

このタスクを使用して、初期 DB2 データベース設定を指定します。この情報は、例を示す目的で提供しています。

このタスクについて

重要: 以下に示すのは、Business Process Choreographer データベースに関する情報です。WebSphere のデフォルトのメッセージング・データベースのチューニングについて詳しくは、WebSphere Application Server Network Deployment インフォメーション・センターのメッセージング・エンジンのデータ・ストアのチューニングと問題解決を参照してください。

データベースを良好に運用するには、データベースの初期設定値を指定します。さらに、ストライプ・サイズが異なる 2 つの別個の論理ディスクを使用します。また、インスタンスごとに 1 つのデータベースを使用して、メッセージング・エンジンがデータ・ストアとしてデータベースを使用する場合にメッセージング用に

Business Process Choreographer データベースを使用できるようにするか、またはメッセージング・データベース用に別個のデータベース・マシンを使用できるようにします。

注: こうした設定はこのデータベース・タイプに標準装備されているため、この情報は、DB2 UDB for iSeries には該当しません。

手順

1. ログ・ファイルをデータ・ファイルから分離します。

   データから分離されたディスク・ドライブにデータベースのログ・ファイルを配置すると、十分な数のディスク・ドライブが使用可能であれば、パフォーマンスが向上する傾向があります。

   例えば、Windows システムで DB2 を使用する場合は、次のコマンドを入力することによって、BPEDB という名前のデータベースのログ・ファイルの場所を F:¥db2logs ディレクトリーに変更することができます。

   ```
   db2 UPDATE DB CFG FOR BPEDB USING NEWLOGPATH F:¥db2logs
   ```

2. テーブル・スペースを作成します。

   データベースを作成したら、テーブル・スペースを明示的に作成します。テーブル・スペースを作成するためのサンプル・スクリプトは、Business Process Choreographer により、WebSphere Application Server インストール・サブディレクトリーの ProcessChoreographer に準備されています。それらのスクリプトをカスタマイズして、特定のシナリオのニーズに対処します。テーブル・スペースを作成する場合の目標は、DB2 で使用可能なできるだけ多くのディスク・ドライブ全体に入出力操作を分散させることです。デフォルトでは、これらのスクリプトにより次のテーブル・スペースが作成されます。

   AUDITLOG
   プロセスおよびタスクの監査証跡テーブルを格納します。使用される監査の程度に応じて、このテーブル・スペース内のテーブルへのアクセス数が多くなる可能性があります。監査をオフにすると、このテーブル・スペース内のテーブルにアクセスできなくなります。

   COMP
   Business Process Choreographer バージョン 5 のビジネス・プロセス用の補正テーブルを格納します。補正可能なプロセスとアクティビティーの割合によっては、このテーブル・スペース内のテーブルでディスク高帯域幅が必要になる場合があります。ビジネス・プロセス内で補正が使用されない場合は、このテーブル・スペース内のテーブルは使用されません。

   INSTANCE
   プロセス・インスタンスおよびタスク・テーブルを格納します。実行される長期実行プロセスの種類に関係なく、常時集中的に使用されます。可能であれば、このテーブル・スペースを数台のディスク・ドライブに広げます。

   SCHEDTS
   WebSphere スケジューリング・コンポーネントが使用するテーブルを格

納します。スケジューラー・テーブル・スペース内のテーブルへのアクセスは、スケジューラーで使用されるキャッシング機構のために通常低速で行われます。

STAFFQRY
Lightweight Directory Access Protocol (LDAP) のような社員レジストリーから入手される社員照会結果を一時的に保管するために使用するテーブルを含みます。ビジネス・プロセスに数多くの person アクティビティーが含まれている場合は、このテーブル・スペース内のテーブルへのアクセスが頻繁に行われます。

TEMPLATE
プロセスおよびタスクのテンプレート情報を保管するテーブルを格納します。アプリケーションのデプロイメント中に、テーブルにデータが取り込まれます。実行時には、アクセス速度が低下します。デプロイメント中は、データは更新されず、新規データのみが挿入されます。

WORKITEM
作業項目の処理に必要なテーブルを保持します。作業項目は、ヒューマン・タスクの対話に使用されます。ビジネス・プロセス内のヒューマン・タスクの数に応じて、このテーブル・スペースのテーブルへのアクセス速度が、低速からかなりの高速まで変わる場合があります。使用されている明示的なヒューマン・タスクがない場合でも、アクセス速度はゼロになりません。これは、長期実行プロセスの管理をサポートするために、作業項目も生成されるからです。

ハイパフォーマンスを生み出すデータベースを作成するには、以下のアクションを実行します。

a. データベースを作成します。

   Windows では、宛先ドライブを指定することができます。コマンドにより、宛先ドライブ上で、サーバーのデフォルトの DB2 インスタンスと同じ名前のディレクトリー内にデータベースが作成されます。例えば、ドライブ `D:` 上にデータベースを作成する場合、ローカルのデフォルト・インスタンスが DB2 であれば、データベース・データは `D:¥DB2` に格納されます。したがって、`D:` ディレクトリーに Business Process Choreographer 用の DB2 データベースを作成するには、次のコマンドを入力します。

   ```
   CREATE DATABASE BPEDB ON D: USING CODESET UTF-8 TERRITORY en-us;
   ```

   UNIX および Linux の場合は、以下のコマンドを入力します。

   ```
   CREATE DATABASE BPEDB ON /wasdbfs USING CODESET UTF-8 TERRITORY en-us;
   ```

   ここで、`/wasdbfs` はディレクトリーを示します。

b. 希望するディスクにテーブル・スペースを作成します。

   例えば、以下のスクリプトは、WebSphere Application Server インストール・サブディレクトリーの `ProcessChoreographer` にある `createTablespaceDb2.ddl` ファイルを基にしています。Windows システムで単一の高性能ディスク・ドライブを使用してテーブル・スペースを作成します。

```
-- Scriptfile to create tablespaces for DB2 UDB
-- Replace occurence or @location@ in this file with the location
-- where you want the tablespace containers to be stored, then run:
-- db2 connect to BPEDB
-- db2 -tf createTablespaceDb2.ddl


CREATE TABLESPACE TEMPLATE MANAGED BY SYSTEM USING( 'D:/BPE/TEMPLATE' );

CREATE TABLESPACE STAFFQRY MANAGED BY SYSTEM USING( 'D:/BPE/STAFFQRY' );

CREATE TABLESPACE AUDITLOG MANAGED BY SYSTEM USING( 'D:/BPE/AUDITLOG' );

CREATE TABLESPACE COMP MANAGED BY SYSTEM USING( 'D:/BPE/COMP' );

CREATE TABLESPACE INSTANCE MANAGED BY SYSTEM USING( 'D:/BPE/INSTANCE', 'D:/BPE/INSTANCE');

CREATE TABLESPACE WORKITEM MANAGED BY SYSTEM USING( 'D:/BPE/WORKITEM' );

CREATE TABLESPACE SCHEDTS MANAGED BY SYSTEM USING( 'D:/BPE/SCHEDTS' );
```

c. テーブルを作成します。

個々のデータベース用に提供されたスクリプトを実行して、Business Process Choreographer テーブルを作成します。例えば、DB2 の場合、`ProcessChoreographer` ディレクトリーの `createSchemaDb2.ddl` ファイルを使用します。

3. データベースを調整します。

キャパシティー・プランニング・ツールを使用して、データベースの初期設定を行います。

DB2 を使用している場合は、Business Process Choreographer データベースのポップアップ・メニューから「**DB2 configuration advisor**」を選択して、DB2 コントロール・センターから DB2 Configuration Advisor を始動します。以下のアクションを実行します。

a. メモリーに DB2 を割り振ります。

「**サーバー**」では、スワッピングしないで物理的に使用可能な最大限のメモリーのみを DB2 に割り振ります。

b. ワークロードのタイプを指定します。

「**ワークロード (Workload)**」では、「**混合 (Mixed)**」 (照会とトランザクション) を選択します。

c. 「**トランザクション**」では、トランザクションの長さと、毎分処理されるトランザクションの推定数を指定します。

「**10 より大きい (More than 10)**」を選択し、長いトランザクションが使用されることを示します。

次に、「**毎分のトランザクション (Transactions per minute)**」フィールドで、毎分処理されるトランザクションの推定数を選択します。この数値を判別するには、プロセスの各アクティビティーにトランザクションが 1 つあると想定します。このとき、1 分間に実行されるトランザクション数は次のようになります。

*毎分実行されるトランザクション数 = 毎分完了するプロセス数 * 各プロセス内のアクティビティー数*

d. データベースを調整して、トランザクション・パフォーマンスがより高速になり、リカバリーが遅くなるようにします。

「**優先順位**」では、「**高速トランザクション・パフォーマンス (Faster transaction performance)**」を選択します。

e. 可能な場合、実動中の標準的な量のデータを取り込んだデータベースを調整します。 「**データ取り込み済み (Populated)**」では、「**はい**」を選択します。それ以外の場合は、「**いいえ**」を選択します。

f. 並列接続設定を調整します。

「**接続**」では、アプリケーション・サーバーに対して確立可能な並列接続の最大数を指定します。この値を決定する場合は、以下の点を指針にしてください。

- 必要なデータベース接続数は、WebSphere Application Server への JDBC (Java DataBase Connectivity) の数によって決まります。JDBC 接続は、WebSphere Application Server にある JDBC 接続プールによって準備されます。*p* JDBC 接続では、*p* * 1.1 本のデータベース接続が必要です。*p* の現実的な値の見積もり方法については、688 ページの『アプリケーション・サーバーのチューニング』で説明します。
- Business Process Choreographer とデータベースが同じ物理サーバーにインストールされている場合、Business Process Choreographer ではリモート・データベース接続が不要です。ただし、リモート・データベース管理でリモート接続が必要になる可能性があるので、ゼロではなく低い値を指定します。
- Business Process Choreographer と DB2 が別々のサーバーにインストールされている場合は、ローカル接続に関して前に説明した規則に従って、リモート・アプリケーションの数を設定します。

g. 「**分離 (Isolation)**」では、「**読み取り固定**」を選択します。 Business Process Choreographer では、この分離レベルが必要です。

構成アドバイザーに提案される変更点が表示されます。すぐに変更点を適用することも、ファイルに保管して後で適用することもできます。

結果

ユーザーの長期間にわたって実行するプロセスは、現在の環境とロード条件において可能な限り高速実行されます。

初期 Oracle データベース設定の指定

Oracle データベースのパフォーマンスの向上およびスケーラビリティーの拡張は、主にデータベース・ファイルのレイアウトを最適化し、十分なメモリーをバッファー・キャッシュに割り振って効果的なキャッシングおよびデータベース調整パラメーターを使用可能にすることによって行われます。

手順

1. 十分なスペースをバッファー・キャッシュに割り振ります。

メモリー内キャッシングを使用すると、データベース・アクセス用の応答時間の待ち時間が短くなります。これは、バッファー・キャッシュを十分な大きさにする必要があることを意味します。バッファー・キャッシュ・サイズを少なくとも 700 MB に設定し、キャッシュ使用量をモニターしてから、必要に応じてキャッシュ・サイズを大きくします。

2. ログ・ファイルの交換が減るように、ログ・ファイルをサイズ変更します。

   Oracle インスタンスのトランザクション・ログは、ラウンドロビン形式で使用されるいくつかのファイルにあります。アクティブ・ログ・ファイルはいっぱいになると交換されるので、最後のアクティブ・ログまで保存できます。ログ・ファイルの交換は費用のかかる操作であるため、交換が頻繁に発生しないようにログ・ファイルをサイズ変更してください。開始値として適切なのは 750 MB です。その後、トランザクション速度および平均ログ・サイズをモニターし、必要に応じてこの値を調整します。

3. 以下のデータベース・パラメーターを調整します。

   UNDO_TABLESPACE
   取り消しテーブル・スペースが、テーブル・スペース・サイズ制限の 70 パーセント (%) を超えて使用されないようにしてください。

   OPEN_CURSORS
   このパラメーターのデフォルト値は 50 です。ただし、これでは足りないことがよくあります。このパラメーターに使用できる最高値はオペレーティング・システムによって異なります。ほとんどのオペレーティング・システムでは、1000 までの値がサポートされています。

   MAX_SHARED_SERVERS
   同時に実行できる共用サーバー・プロセスの最大数を指定します。このパラメーターを使用して、他のプロセス用のプロセス・スロット (例えば専用サーバー) を予約します。 MAX_SHARED_SERVERS パラメーターの値が指定されている場合、それは SHARED_SERVERS パラメーターの値以上か、または PROCESSES パラメーターの値より小さくなければなりません。例えば、150 人の同時ユーザーがいる場合、このパラメーターの開始値として MAX_SHARED_SERVERS=70 が適しています。

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 Network Deployment インフォメーション・センターのサービス統

合の調整のプロパティーの設定にある説明に従って、メッセージング・エンジン用のデータ・ストアをセットアップおよび調整してください。

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。

関連タスク

143 ページの『メッセージング・エンジン・データベースの計画』
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。

アプリケーション・サーバーのチューニング

このタスクを使用して、アプリケーション・サーバーを調整します。

始める前に

このタスクを始める前に、682 ページの『初期 DB2 データベース設定の指定』で説明されるとおりに、データベースの初期設定値を指定する必要があります。

このタスクについて

ビジネス・プロセス・コンテナーが確実に最適な状態で動作できるようにするため、サーバー設定を調整する必要があります。

手順

1. ビジネス・プロセス・コンテナーごとに必要なアプリケーション・サーバーのリソースを見積もります。
   a. ビジネス・プロセスの状態情報をデータベースに読み書きするための 1 つのデータ・ソース: BPEDDataSourceDb2 (サーバー・スコープ DB2 Universal JDBC Driver Provider (XA))
   b. 以下を追加して、プロセス・ナビゲーションの最大並行トランザクション数 *t* を計算します。
      - Business Process Choreographer API を通じて並行して接続されるクライアントの最大数
      - JMS 活動化仕様 BPEInternalActivationSpec で定義される並行エンドポイント数
      - JMS 活動化仕様 HTMInternalActivationSpec で定義される並行エンドポイント数

      プロセス・サーバーの活動化仕様を表示するには、管理コンソールの「**リソース**」 → 「**JMS プロバイダー**」 → 「**デフォルトのメッセージング**」 → 「**アクティベーション・スペック**」をクリックします。
   c. Business Process Choreographer データベースの場合、必要な並列 JDBC 接続の数 $p = 1.1 * t$ を計算します。

      p の値は、そのデータベースで許可される接続数以下にしてください。

d. メッセージング・データベースの場合、必要な並列 JDBC 接続の数 $m = t + x$ を計算します。ここで x は、追加メッセージが生成され処理される必要がある過負荷状態を許容できるようにするための、追加 JMS セッション数です。

e. SQL「ステートメント・キャッシュ・サイズ」を 50 に設定します。

2. Business Process Choreographer データベース (BPEDB) の JDBC プロバイダー設定を調整します。

a. 「**最大接続数**」を値 p に設定します。 p の値は、そのデータベースで許可される接続数以下にしてください。

b. SQL「**ステートメント・キャッシュ・サイズ**」を 300 に設定します。

3. ステップ 2 を繰り返して、SCA メッセージング・エンジン・データ・ストアの JDBC プロバイダー設定を調整します。

4. メッセージング・データベースの JDBC プロバイダー設定を調整します。

「**最大接続数**」を値 m に設定します。

5. ヒープ・サイズを調整します。

32 ビット・システムでのサーバー・ヒープのサイズを決める場合の指針を以下に示します。 これらのガイドラインは、i5/OS システムで稼働するサーバーには適用しません。

- 256 MB では少なすぎ、ローパフォーマンスになる。
- 512 MB は多くのシステムの初期ヒープ・サイズとして適当。
- 1024 MB は合理的な上限。

64 ビット・システムの場合、ヒープに適したサイズは 1 から 2 GB です。

6. ビジネス・プロセスが使用するサービスを調整します。 サポート用サービスが、サービスでの Business Process Choreographer の並行性の程度および負荷要求に対処できるように調整されていることを確認してください。

結果

アプリケーション・サーバーが調整され、パフォーマンスが改善されます。

関連タスク

681 ページの『ハードウェア・リソースの平衡化』
ハードウェア・リソースを平衡化することにより、長期実行ビジネス・プロセスのパフォーマンスを改善します。

データベースの細密チューニング

このタスクを使用して、データベースを細密チューニングします。

始める前に

ビジネス・プロセス・コンテナーおよびビジネス・プロセスが実行中であることが必要です。

このタスクについて

注: DB2 を使用していない場合は、データベースのパフォーマンスのモニター、ボトルネックの識別と除去、およびパフォーマンスの細密チューニングについて、データベース管理システムの文書を参照してください。このトピックの以下の部分では、DB2 ユーザーへの提案を示します。ただし、この情報は、DB2 UDB for iSeries には該当しません。

手順

1. 使用量およびヒット率に従って、バッファー・プールにサイズを割り当てます。

   バッファー・プール・ヒット率は、既にプール内にあるデータから満たすことができるデータベース要求のパーセンテージを示します。これは 100 パーセントに近くなければなりませんが、90 パーセントを超える値であれば許容されます。ヒット率が十分な値になるまで、バッファー・プールの **SIZE** パラメーターを大きくしてください。メモリー割り振りの合計をモニターします。バッファー・プールを大きくし過ぎると、システムはスワップを始めます。この場合、バッファー・プールのサイズを小さくするか、または追加メモリーを使用可能にしてください。

   DB2 バージョン 8 を使用している場合、バッファー・プール・ヒット率を計算できます。計算に必要な値はバッファー・プール・スナップショットから取得できます。以下のコマンドを使用して、スナップショットを取得します。

   ```
   DB2 get snapshot for all bufferpools
   ```

   ヒット率の計算について詳しくは、DB2 V8 インフォメーション・センターを参照してください。

   DB2 バージョン 9 を使用している場合、BP_HITRATIO 管理ビューを使用してヒット率情報を取得します。このビューについて詳しくは、DB2 V9 インフォメーション・センターを参照してください。

   DB2 構成アドバイザーはバッファー・プール・サイズの値を推奨し、Business Process Choreographer データベースはデフォルトで IBMDEFAULTBP デフォルト・バッファー・プールのみを使用します。このバッファー・プールのサイズは以下のコマンドを使用して設定できます。

   ```
   DB2 ALTER BUFFERPOOL IBMDEFAULTBP SIZE 120000
   ```

   このコマンドにより、バッファー・プールとそのサイズ (ページ単位)、および各ページのサイズが表示されます。

   ```
   DB2 select BPNAME, NPAGES, PAGESIZE from syscat.bufferpools
   ```

2. ロック・リスト・スペースを調整して、最適なパフォーマンスを保証します。

   すべてのロックにはストレージが必要であり、このストレージは制限されています。この制限を超えるロックを要求するトランザクションは異常終了する必要があるため、そのトランザクションのパフォーマンスが低下します。

   a. ご使用の DB2 インスタンスに対する `db2diag.log` ファイルを確認します。

      次の例にあるような項目を探します。

      ```
      2005-07-24-15.53.42.078000 Instance:DB2 Node:000
      PID:2352(db2syscs.exe)   TID:4360   Appid:*LOCAL.DB2.027785142343
      data management  sqldEscalateLocks Probe:4   Database:BPEDB
      ```

```
ADM5503E The escalation of "10" locks on table
"DB2ADMIN.ACTIVITY_INSTANCE_B_T" to lock intent
"X" has failed. The SQLCODE is "-911".
```

このタイプのメッセージは、ロック・スペースを超えていることを示します。

b. MAXLOCKS および LOCKLIST パラメーターの値を増やします。

これらのパラメーターは、ロック・エスカレーションでのデータベースの動作を制御します。ロック・エスカレーションは、同じテーブルにある複数の個々の行レベルのロックを単一テーブル・レベルのロックに変換します。

トランザクションがロック・リストの MAXLOCKS パラメーターを上回る値を使用する場合、ロック・スペース制限を超えないように、データベース・マネージャーはこれらのロックを単一テーブル・ロックに変換します。ただし、ロック・エスカレーションにより、デッドロックの可能性が大幅に高まります。そのため、MAXLOCKS パラメーターの値を 60 パーセントに増やしてください。

LOCKLIST パラメーターの値を概算で 10 * *p* に増やしてください。ここで、*p* は任意の時点で必要になる並列 JDBC 接続の最大数の推定値です。例えば、Business Process Choreographer データベースである BPEDB のサイズを、値として *p*=50 を使用して変更した場合は、次のコマンドを入力します。

```
db2 UPDATE DB CFG FOR BPEDB USING LOCKLIST 500
```

3. DB2 構成アドバイザーを使用した場合、データベースのスループットは通常良好な状態になっています。ただし、以下のようにしてさらにパフォーマンスを改善することができます。
   - DB2 オンライン文書、ブック、および記事で説明されている最良事例に従って、データベースのチューニングを行ってください。
   - 以下の DB2 パラメーターを調整します。

     AVG_APPLS
     このパラメーターは、低すぎるよりも高すぎる方が、望ましい結果が得られます。例えば、最大 20 個の接続されたアプリケーションがある場合は、AVG_APPLS を 50 に設定します。

     DLCHKTIME
     このパラメーターは、デッドロック検出時間フレームを指定します。デフォルトは 10 秒です。

     LOCKTIMEOUT
     このパラメーターは、アプリケーションがロックを待機する時間を指定します。デフォルトは -1 です。これは、ロックが認可されるか、またはデッドロックが発生するまでアプリケーションが待機することを意味します。デッドロックがロック・タイムアウトではなくデッドロックとして報告されるようにするため、このパラメーターの値は常に DLCHKTIME パラメーターの値より大きくなければなりません。

このパラメーターに適した初期値は 30 秒です。負荷テストで 30 秒を超えるトランザクション時間が示される場合は、値を高めに設定することもできます。

LOGBUFSZ
DB2 ログのバッファーのサイズを大きくすると、ログ・バッファー全体をディスクに書き込まなければならない頻度が減少します。

LOG_FILSIZ
ログ・ファイルのサイズを大きくすると、ログ・ファイルが切り替えられる頻度が減少します。

4. データベースおよびデータベース・マネージャーの設定をワークロード要件に応じて調整します。 構成アドバイザーがデータベースを構成したあと、以下の設定を調整することもできます。

   MINCOMMIT
   1 の値を強くお勧めします。DB2 構成アドバイザーは他の値を提示する場合があります。

   NUM_IOCLEANERS
   値が 0 に設定された照会専用アプリケーションの場合、通常処理では 1 とシステム内のディスク・ドライブ数の間の値を使用します (NUM_IOSERVERS パラメーターも参照してください)。大容量バッファー・プールの場合、通常は高い数値を使用すると役立ちます。

   NUM_IOSERVERS
   データベースが常駐している物理ディスクの数に一致しなければなりません。少なくともディスクと同じ数の IO サーバーを持つ必要があります。IO サーバーはシステム・リソースをあまり使用しないので、低すぎる値を設定するよりは、高すぎる値を設定してください。

5. データベースの DB2 統計を更新します。

   最初にシステムで負荷をかけた後、またはデータベース内のデータ・ボリュームが大幅に変化するときには常に、統計を含む DB2 システム・カタログ・テーブルを更新することを検討してください。 RUNSTATS コマンドを使用して、統計を更新します。

   RUNSTATS コマンドは、スクリプトを使用すると最適に実行します。以下の例は、そうしたスクリプトを示しています。これは、ユーザー `bpeuser` とそのパスワード `password` でログオンしており、Business Process Choreographer データベース、`BPEDB` に接続していることを想定しています。 DB2 コマンドは、Windows コマンド・ファイルを生成し、このファイルは BPEDB データベース内の関係のあるテーブル・スペースのすべてのテーブルの統計を更新します。情報のアクセスまたは更新がめったにないため、TEMPLATE テーブル・スペースのテーブルは省略されます。

```
db2 -x "select 'db2 runstats on table '
        concat rtrim(tabschema)
        concat '.'
        concat tabname
        concat ' with distribution and detailed indexes all '
from syscat.tables
where
        type='T'  AND
         TBSPACEID IN (select TBSPACEID from sysibm.systablespaces
```

```
                where TBSPACE IN ('INSTANCE', 'WORKITEM', 'STAFFQUERY',
                   'AUDITLOG', 'SCHEDTS'))" > runStatsScript.cmd

echo db2 connect reset >> runStatsScript.cmd
```

生成されるコマンド・ファイルは、指定されたテーブルの統計を更新します。これには以下のような項目が含まれます。

```
db2 runstats on table BPEUSER.ACTIVITY_INSTANCE_B_T with distribution and
 detailed indexes all
db2 runstats on table BPEUSER.AUDIT_LOG_T with distribution and
 detailed indexes all
...
db2 connect reset
```

RUNSTATS コマンドを呼び出す前に REORG コマンドを実行するように、コマンド・ファイルを拡張できます。 REORG コマンドを使用してデータベース・テーブルを再編成する方法については、DB2 の資料を参照してください。

6. デッドロックを避けます。

   少なくとも 2 つのトランザクションが相互のリソース・アクセスをブロックすると、デッドロックが起こります。デッドロックは、貧弱なデータベース構成によって発生する可能性があります。また、Business Process Choreographer API の使用方法によっても発生する可能性があります。デッドロックを避けるには、データベース内のオブジェクトの各 API 呼び出しまたは照会がそれぞれ独自のトランザクション内で実行される必要があります。

   Business Flow Manager はデータベース・デッドロックから回復できます。ただし、デッドロックが検出される時間とロールバックされたトランザクションが再試行される時間の差が原因で、パフォーマンスに重大な影響を与えることがあります。そのため、パフォーマンス上の理由により、デッドロックは避けるべきです。

   デッドロックを検査するには、db2diag.log ファイルを調べ、DB2 モニターを使用します。

   a. db2diag.log ファイルのログ・レベルを大きくして、データベース内部のボトルネックについての詳しい情報を取得します。

      DIAGLEVEL パラメーターの値を 3 (デフォルト) から 4 に増やして、エラー、警告、および情報メッセージを含めます。この値を変更するには、以下のコマンドを使用できます。

      ```
      db2 update dbm cfg using DIAGLEVEL 4
      ```

   b. DB2 イベント・モニターを作成します。

      イベント・モニターは、特定のイベント (例えばデッドロック) について詳しい情報を提供します。

      1) 以下のコマンドを使用して、イベント・モニターを作成します。

         ```
         db2 create event monitor monitor_name for statements, connections,
           transactions, deadlocks with details write to file file_name
         ```

      2) 以下のコマンドを使用して、イベント・モニターを開始します。

         ```
         db2 set event monitor monitor_name state=1
         ```

      3) 以下のコマンドを使用して、情報を収集します。

```
db2evmon -db database_name -evm monitor_name output_file_name
```

c. データベース・スナップショット・モニターを使用して、統計を収集します。

スナップショット・モニターはデータベース・モニター・スイッチを使用します。データベース・インスタンスの場合、モニター・スイッチには以下のデフォルト設定があります。

バッファー・プール (DFT_MON_BUFPOOL) = ON

ロック (DFT_MON_LOCK) = ON

ソート (DFT_MON_SORT) = OFF

ステートメント (DFT_MONSTMT) = OFF

テーブル (DFT_MON_TABLE) = OFF

タイム・スタンプ (DFT_MON_TABLE) = ON

作業単位 (DFT_MON_Table) = OFF

データベース・インスタンスの現行設定を確認するには、以下のコマンドを使用し、DFT_MON_ で始まるすべてのパラメーターを検索します。

```
db2 get dbm cfg
```

これらの設定は、データベースの設定とは異なります。データベースの場合、モニター記録スイッチには以下のデフォルト設定があります。

db 区画番号 0 のスイッチ・リスト

バッファー・プール・アクティビティー情報 (BUFFERPOOL) = ON

ロック情報 (LOCK) = ON

ソート情報 (SORT) = OFF SQL

ステートメント情報 (STATEMENT) = OFF

テーブル・アクティビティー情報 (TABLE) = OFF

タイム・スタンプ取得情報 (TIMESTAMP) = ON

作業単位情報 (UOW) = OFF

データベースの現行設定を調べるには、以下のコマンドを使用します。

```
db2 get monitor switches
```

- データベース・モニターのいずれか (例えばロック・モニター) の設定を更新するには、以下のコマンドを使用します。

  ```
  db2 update monitor switches using lock on
  ```

  この設定は、現行データベース・セッションにのみ有効です。
- データベース・インスタンスのモニター・スイッチのいずれか (例えばロック・モニター) の設定を更新するには、以下のコマンドを使用します。

  ```
  db2 update dbm cfg using DFT_MON_LOCK OFF
  ```

  設定をアクティブにするには、データベース・インスタンスを再始動します。
- スナップショット・モニターを使用可能にする前に、以下のコマンドを使用してカウンターをリセットします。

```
db2 reset monitor all
```

- データベース・インスタンスを再始動した後で現行スナップショットを取得するには、以下のコマンドを使用します。

  ```
  db2 get snapshot for all on database_name output_file_name
  ```

結果

ユーザーの長期間にわたって実行するプロセスは、現在の環境とロード条件において可能な限り高速実行されます。

メッセージング・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・エンジンのデータ・ストアのチューニングと問題解決を参照してください。

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。

microflow のチューニング

microflow のパフォーマンスを向上させるには、このタスクを使用します。

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。監査ロギングまたは Common Event Infrastructure (CEI) が microflow で使用可能になっている場合のみ、データベース・アクセスが必要です。microflow の処理が、単一スレッド内で行われ、また通常は単一トランザクション内で行われます。 microflows のパフォーマンスは、主として呼び出されるサービスによって決まります。ただし、サーバーが使用可能なメモリーが小さすぎる場合、microflow のパフォーマンスは低下します。

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。サーバー・ヒープのサイズの指針については、688 ページの『アプリケーション・サーバーのチューニング』の関係するステップを参照してください。

2. JVM のガーベッジ・コレクションを調整します。 Throughput Garbage Collector を使用すると最高のスループットが得られますが、ヒープ・サイズによっては、

ガーベッジ・コレクションが 100 から 1000 ミリ秒の間一時停止する場合があります。スループットより応答時間の方を重視する場合は、Low Pause Garbage Collector を使用してください。

3. オブジェクト・リクエスト・ブローカー (ORB) のスレッド・プール・サイズを調整します。 リモート・クライアントがサーバー・サイド ORB に接続する場合は、ORB スレッド・プールに使用できるスレッドが十分あることを確認してください。
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***」 → 「**プロパティーの追加**」 → 「**スレッド・プール**」 → 「**デフォルト**」をクリックします。

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。

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。

照会の応答時間の短縮

データベースが照会への応答に必要とする時間を短縮します。

カスタム・クライアントを使用する場合は、必ず照会でしきい値を設定してください。使用可能度の観点から、一般的に、何百、何千もの項目の検索は望ましいことではありません。データベースの操作数が増えるほど、タスクの完了までにかかる時間が長くなること、および人は一度に少数の結果しか管理できないことがその理由です。しきい値を指定すると、データベースの負荷やネットワーク・トラフィックを最小限に押さえ、クライアントが確実にデータをすぐに提供できるようになります。

多数の項目を戻す照会の処理方法としてよりよい方法は、照会を作成し直して、戻される結果セットの項目数を減らすことです。これを行うには、特定のプロセス・インスタンスのみについて作業項目を照会するか、一定の日付を持つ作業項目のみを照会します。

フィルター基準を使用して、照会結果を削減することもできます。

全テーブルのスキャンニングの回避

照会アプリケーション・プログラミング・インターフェース (API) を使用して、データベース内のオブジェクトをリストする場合、フィルター操作を指定して、取得する結果の範囲を絞ることができます。このフィルターに、オブジェクト属性の値と範囲を指定することができます。

データベース照会の処理時に、フィルター情報が変換され、Structured Query Language (SQL) ステートメントの WHERE 文節に格納されます。この WHERE 文節は、影響を受けるデータベース・テーブル内の列名にオブジェクト属性をマップします。

照会に索引付きテーブル列への変換を行わないフィルターが指定されている場合、SQL ステートメントによってテーブルがスキャンされることになります。このスキャンニングによって、パフォーマンスに悪影響があり、デッドロックが発生するリスクが高くなります。このパフォーマンスの悪影響は、1 日に数回のみという程度であれば容認されますが、1 分間に数回という頻度になれば、効率に悪影響を及ぼします。

このような状況では、カスタム索引でこの影響を劇的に抑えることができます。実際のお客様の状況で、カスタム索引により、API の応答時間を 25 秒から 300 ミリ秒に減らすことができました。データベース・テーブルの 724 000 行を読み取る代わりに、6 行を読み取るだけでよいのです。Business Process Choreographer データベースのインスタンス・テーブルに対して volatile フラグを定義すると、DB2 オプ

ティマイザーが適切なデータ・アクセス・プランを決定する上で役立ちます。このフラグは、空のテーブルやほぼ空のテーブルであっても、常にテーブル・スキャンではなく索引が使用されることを指定します。

指定するフィルター基準により、一部の列が索引に含まれないように設定することができます。該当する場合、テーブル・スキャンを使用し、その結果照会のパフォーマンスが低下する場合は、例えば DB2 Explain を使用して、ステートメントのアクセス・パスを確認してください。必要であれば、新しい索引を定義してください。

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・テーブル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。

- 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 (Business Process Choreographer query() and queryAll methods: best practices)

ヒューマン・ワークフローの調整 (Tuning human workflows)

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. 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。

2. 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。これらの設定は管理コンソールを使用して変更できます (「**サーバー**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」 → 「**スレッド・プール**」 → 「**Web コンテナー**」)。

   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。この値を変更すると、応答時間にも影響を与える可能性があります。値 0 は待機時間がないことを示します。

3. 大規模リストの場合は、しきい値を小さくします。

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。

関連タスク

238 ページの『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 を構成できます。

関連資料

239 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』
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。

第 23 章 Business Process Choreographer Observer 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。Business Process Choreographer Observer のパフォーマンスを向上させるためのさまざまなテクニックを以下に示します。

適切なデータベース管理システムを選択する

Business Process Choreographer Observer では、DB2、 Oracle、または Derby データベースを使用できます。Derby データベースは、開発、デモンストレーション、およびプロトタイピングに最適ですが、パフォーマンスに主として重点が置かれておらず、大量のデータを格納できません。実動システムには、DB2 または Oracle データベースを使用します。これは、この 2 種類のデータベースでは、大容量データの処理速度が前述のデータベースよりも大幅に高速であるためです。

データベース統計を更新する

DB2 および Oracle 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 FOR INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.EVENT_PRC_T FOR INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.INST_ACT_T FOR INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.INST_PRC_T FOR INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.OPEN_EVENTS_T FOR INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.QUERY_T FOR INDEXES ALL;
RUNSTATS ON TABLE schema_prefix.SLICES_T FOR INDEXES ALL;
```

- Oracle 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。

```
ANALYZE TABLE schema_prefix.EVENT_ACT_T COMPUTE STATISTICS;
ANALYZE TABLE schema_prefix.EVENT_PRC_T COMPUTE STATISTICS;
ANALYZE TABLE schema_prefix.INST_ACT_T COMPUTE STATISTICS;
ANALYZE TABLE schema_prefix.INST_PRC_T COMPUTE STATISTICS;
ANALYZE TABLE schema_prefix.OPEN_EVENTS_T COMPUTE STATISTICS;
ANALYZE TABLE schema_prefix.QUERY_T COMPUTE STATISTICS;
ANALYZE TABLE schema_prefix.SLICES_T COMPUTE STATISTICS;
```

ここで *schema_prefix* は、Business Process Choreographer Observer 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。

関連するイベントのみを発行する

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。アクティビティーを含むプロセスに対してもイベントが生成される場合、Business Process Choreographer Observer ではアクティビティー監査イベントのみが認識されます。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。

Business Process Choreographer Observer からアクティビティーまたはプロセスの正確な情報が提供されるようにするには、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でおきます。

ユーザー定義関数 (UDF) 実装を選択する

Business Process Choreographer Observer では、レポートを作成するために、データベースに特定の UDF がインストールされている必要があります。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。

別のデータベースを使用する

Business Process Choreographer Observer と Business Process Choreographer が同一データベースを使用する場合、相互のパフォーマンスに悪影響が及びます。Business Process Choreographer Observer に別のデータベースを使用すると、パフォーマンスが向上します。また、Observer データベースを別のデータベース・サーバーでホストすることも検討してください。

別のマシンを使用する

Business Process Choreographer Observer がインストールされているマシンで、他のアプリケーション (BPEContainer アプリケーションや TaskContainer アプリケーションなど) がホストされている場合は、期待されるパフォーマンスを実現できる十分なリソースを備えた別のマシンで Business Process Choreographer Observer を実行することを検討してください。

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**」 → 「**アプリケーション・サーバー**」 → 「***server_name***」 → 「**トランザクション・サービス**」にナビゲートします。
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、データ・ソースの接続プール・プロパティーにナビゲー

トします。「**JDBC**」 → 「**JDBC プロバイダー**」>「***JDBC provider***」 → 「**データ・ソース**」 → 「***data_source_name***」 → 「**接続プール・プロパティー**」の順にクリックし、「**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、Observer 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。レポートで大量のデータを照会する場合は、パフォーマンスが低下します。Business Process Choreographer Observer データベース内のプロセスとアクティビティー・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が向上することがあります。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。

関連タスク

275 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』
setupEventCollector ツールを使用するか、またはスクリプトを実行して、Business Process Choreographer Observer 用のデータベースで 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ことができます。

348 ページの『Observer データベースからのデータの削除』
Business Process Choreographer Observer データベースから、指定の条件に一致したプロセス・インスタンスのデータ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。

第 7 部 トラブルシューティング

第 24 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。

関連情報

WebSphere Process Server のトラブルシューティング・ガイド

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 createBPCObjects.traceout ファイルに書き込まれます。Windows システムでは、これらのファイルは *install_root*/logs/manageprofiles/*profileName*/logs ディレクトリーにあります。Linux、UNIX、および i5/OS システムでは、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

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 が起動されるごとに上書きされるので、wsadmin を再度起動する前に必ずこのログ・ファイルを保管してください。

構成関連スクリプト

bpeconfig.jacl、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それぞれのログ・ファイルを、setupOberver.log および setupEventCollector.log ファイルに個々に書き込みます。

- setUpEventCollector.bat および setupObserver.bat (Windows システム)
- setUpEventCollector.sh および setupObserver.sh (Linux および UNIX システム)
- setUpEventCollector および setupObserver (i5/OS システム)

また、wsadmin.traceout ファイルも確認します。

管理ユーティリティー・スクリプト

ProcessChoreographer ディレクトリーの 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.traceout ファイルおよびアプリケーション・サーバーのログ・ファイルを確認します。

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。

- 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・ソースのインプリメンテーション・クラス COM.ibm.db2.jdbc.DB2XADataSource について述べるエラーが発生した場合:

- 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。

• 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。

- 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に J2EE 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・セキュリティーが使用可能になっている場合にはログインを要求しない、という 6.0.x の動作が維持され、後方互換性が保たれます。クライアントの修正後に、これらのロールを値 AllAuthenticated に変更して、非認証ユーザーが API にアクセスするのを防ぐ必要があります。新規インストールの場合、これらのロールはデフォルト値の AllAuthenticated に設定されます。
- ビジネス・プロセスまたはヒューマン・タスクを含むエンタープライズ・アプリケーションのインストール中に、データベース・エラーが発生する場合は、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。

6.0.x Business Process Choreographer API クライアントが 6.1 環境で失敗する

WebSphere Process Server バージョン 6.1 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。6.1 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

Business Process Choreographer API クライアントにユーザー認証が含まれていない場合、そのクライアントはセキュリティー・ホールに依存します。 WebSphere Process Server バージョン 6.1 では、このセキュリティー・ホールは修正されましたが、この修正が原因でクライアントが失敗します。

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「**全認証者**」から「**全員**」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「全認証者」に戻して、非認証ユーザーが API にアクセスするのを防ぎます。

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。
- 以下のログ・ファイル:
  - SystemOut.log
  - SystemErr.log
  - trace.log

  Linux、UNIX、および i5/OS システムでは、これらのファイルは *profile_root*/logs/*server_name* ディレクトリーにあります。Windows プラットフォームでは、これらのファイルは *profile_root*¥logs¥*server_name* ディレクトリーにあります。

  問題シナリオでロギングが大量に発生する場合は、ログのバックアップ・ファイルが、SystemOut_07.10.01_11.00.51.log などの名前で作成されることがあります。管理コンソールを使用すると、作成されるバックアップ・ファイルの数、およびログ・ファイルのサイズを変更できます。すべてのデータを取り込む場合には、これらの値を両方とも増やすことをお勧めします。

  関連情報

  WebSphere Process Server のトラブルシューティング・ガイド

第 25 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、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。詳しくは、『Business Process Choreographer の構成』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 を介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 WebSphere 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・テンプレートおよびヒューマン・タスク・テンプレートが Business Process Choreographer データベースに保管されます。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 を調べる必要があります。

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、このスタンドアロン・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。

この問題がビジネス・プロセスおよびヒューマン・タスク関連であることが示されている場合は、ビジネス・プロセスおよびヒューマン・タスクが含まれるアプリケーションに対してこのトレース・ステートメントを使用してください。

```
*=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、このスタンドアロン・サーバーでトレースを有効にする必要があります。

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する必要があります。

症状: バックレベルの WebSphere Process Server にアプリケーションがデプロイされない

WebSphere Integration Developer バージョン 6.1 で作成されたアプリケーションが WebSphere Process Server バージョン 6.0.2 サーバー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

バージョン 6.0.2 とバージョン 6.1 のクラスター・メンバーが含まれている (クラスター・メンバーの一部が最近マイグレーションされた) クラスターでは、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、更新、またはアンインストールすることができません。

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

このような操作を行う前に、マイグレーションを完了してください。

この制限について詳しくは、『Business Process Choreographer に関するマイグレーションの考慮事項』を参照してください。

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内部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 21268185 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: テンプレートが停止していない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。

- CWTCO0005E: ヒューマン・タスク *task_name* が停止していません。停止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。
- CWWBF0024E: プロセス *process_name* は停止していません。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、テンプレートを手動で停止してください。

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停止していないものが 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、そのすべてのテンプレートが停止状態でなければなりません。

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 (Run in development mode)**」フラグが有効になっている場合です。この場合、テンプレートが開始されていますが、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。「**開発モードでの実行 (Run in development mode)**」フラグの指定方法について詳しくは、http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/urun_rappsvr.html を参照してください。

解決方法

アンインストールするアプリケーションに含まれているすべてのビジネス・プロセス・テンプレートおよびヒューマン・タスク・テンプレートを停止します。さらに、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存在しないことを確認します。これを行うには、635 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 637 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: テンプレートが停止していない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、管理コンソールはテンプレートが停止状態であることを示している

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のアンインストールは、STARTED 状態のビジネス・プロセス・テンプレートまたはヒューマン・タスク・テンプレートがまだ存在することが原因で失敗しましたが、管理コンソールに表示されるそれらの状況は STOPPED 状態を示しています。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

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・テンプレートの状態を示す管理コンソールのエントリーは、Business Process Choreographer データベースに保管されているビジネス・プロセス・テンプレートまたはヒューマン・タスク・テンプレートの状態と同期していない可能性があります。

このようになるのは、管理コンソール内でプロセス・テンプレートまたはヒューマン・タスク・テンプレートを開始したが、構成変更を保存していない場合です。管理コンソールで開始ボタンをクリックするとすぐに Business Process Choreographer データベースで更新が実行されて、変更後の状態が反映されます。そのため、プロセス・テンプレートまたはヒューマン・タスク・テンプレートはデータベースでは開始しています。その後に変更内容を管理コンソールで保存しないと、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が次のような状態を示すことになります。

- 管理コンソールでは STOPPED
- Business Process Choreographer データベースでは STARTED

解決方法

WebSphere 構成リポジトリー内のテンプレート状態と Business Process Choreographer データベースを同期した状態に戻します。そのためには、以下の手順を実行してください。

- 管理コンソールで、アプリケーションのすべてのプロセス・テンプレートおよびタスク・テンプレートを開始します。
- 構成を保管します。
- 管理コンソールで、アプリケーションのすべてのプロセス・テンプレートおよびタスク・テンプレートを停止します。
- 構成を保管します。

上記の手順を完了したら、アプリケーションのアンインストールを再試行します。さらに、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存在しないことを確認します。アプリケーションをアンインストールするには、635 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

637 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。

上記の手順で問題が解決しない場合は、 340 ページの『未使用のプロセス・テンプレートの削除』および 343 ページの『未使用のヒューマン・タスク・テンプレートの削除』の情報を参照してください。

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

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 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 (Run in development mode)**」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。「**開発モードでの実行 (Run in development mode)**」オプションの指定方法について詳しくは、http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/urun_rappsvr.html を参照してください。

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、 635 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 637 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のアンインストールは、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。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、 346 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。これは、実稼働環境でも推奨される方法です。

-force オプションを使用してアプリケーションをアンインストールするには、スクリプト bpcTemplates.jacl 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、 637 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。これにより、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

次のすべての状態が発生した場合は、invoke アクティビティー、インライン・ヒューマン・タスク、および Java 断片を停止状態にします。

- アクティビティーにより障害が発生した
- エンクロージング・スコープで障害が処理されない
- アクティビティーの continueOnError 属性が no に設定されている

解決方法

この問題を解決するには、次の 2 つのレベルの処置が必要です。

1. 管理者は、停止したアクティビティー・インスタンスを手動で修復する必要があります。例えば、停止したアクティビティー・インスタンスを強制完了するか、強制再試行します。
2. 障害の理由を調査する必要があります。場合によっては、モデリング・エラーによって障害が発生することもあり、その場合はモデル内で修正する必要があります。

   495 ページの『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。

   510 ページの『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。

Microflow が補正されない

Microflow がサービスを呼び出したときにプロセスが失敗しましたが、元に戻すサービスが呼び出されません。

解決方法

Microflow の補正を起動するには、さまざまな条件を満たす必要があります。次の点を確認します。

1. Business Process Choreographer Explorer にログオンし、「**失敗した補正**」をクリックして、補正サービスが失敗しており、修復する必要があるかどうかを確認します。
2. Microflow の補正は、Microflow のトランザクションがロールバックした場合にのみ起動されます。この場合に該当するかどうか確認してください。
3. Microflow の compensationSphere 属性を「必須」に設定する必要があります。
4. 補正サービスが実行されるのは、対応する転送サービスが Microflow のトランザクションに関わっていない場合のみです。転送サービスがナビゲーション・トランザクションに関わっていないことを確認してください。例えば、プロセス・コンポーネントの参照時には、Service Component Architecture (SCA) の修飾子 suspendTransaction を True に設定します。

長期実行プロセスが停止しているように見える

長期実行プロセスは実行中の状態になっていますが、何も動作していないように見えます。

理由

このような振る舞いをすることにはさまざまな理由が考えられます。

1. ナビゲーション・メッセージを再試行した回数が多すぎるため、保存キューまたは保留キューに移動された。
2. Service Component Architecture (SCA) インフラストラクチャーからの応答メッセージが繰り返し失敗した。
3. プロセスが、イベント、タイムアウト、あるいは長期実行呼び出しまたはタスクが戻るのを待っている。
4. プロセスのアクティビティーが停止状態になっている。

解決方法

上述の理由それぞれに対して、異なる修正アクションが必要になります。

1. 管理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実行します。
2. 管理コンソールの失敗イベント管理ビューに何か表示されているかどうかを確認します。
   - Service Component Architecture (SCA) 応答メッセージからの失敗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。

実行中に予期しない例外が発生しました (メッセージ: CWWBA0010E)

キュー・マネージャーが実行されていないか、Business Process Choreographer 構成に誤ったデータベース・パスワードが含まれています。

解決方法

次の点を確認します。

1. systemout.log ファイルに "javax.jms.JMSException: MQJMS2005: failed to create MQQueueManager" が含まれている場合、キュー・マネージャーを開始してください。
2. Business Process Choreographer 構成に保管されているデータベース管理者パスワードが、データベースで設定されているパスワードと一致していることを確認してください。

不明のイベント (メッセージ: CWWBE0037E)

プロセス・インスタンスにイベントを送信、または新規プロセス・インスタンスを開始しようとして、「CWWBE0037E: 不明のイベント」例外が発生しました。

理由

このエラーの共通の理由は、メッセージをプロセスに送信しても、receive または pick アクティビティーが既にナビゲート済みであるために、メッセージを再度このプロセス・インスタンスでコンシュームできないというものです。

解決方法

この問題を訂正するには、次の指示に従います。

- イベントが既存のプロセス・インスタンスによってコンシュームされることになっている場合は、対応する receive または pick アクティビティーをまだナビゲートしていない既存のプロセス・インスタンスと一致する、相関セット値を渡す必要があります。
- イベントが新規プロセス・インスタンスを開始させることになっている場合は、相関セット値が既存のプロセス・インスタンスと一致してはいけません。

ビジネス・プロセスで相関セットを使用する方法について詳しくは、「technote 1171649」を参照してください。

プロセス・インスタンスを検索できないか、または作成できません (メッセージ: CWWBA0140E)

プロセス・インスタンスにイベントを送信しようとして、'CreateRejectedException' メッセージを受け取ります。

理由

このエラーの一般的な理由は、`createInstance` 属性が `no` に設定され、このアクティビティーで使用される相関セットのメッセージと一緒に渡される値が既存のプロセス・インスタンスと一致しないために、新規プロセス・インスタンスをインスタンス化できない receive または pick アクティビティーにメッセージが送信されるというものです。

解決方法

この問題を訂正するには、既存のプロセス・インスタンスと一致する相関セット値を渡す必要があります。

ビジネス・プロセスでの相関セットの使用法について詳しくは、「Correlation sets in BPEL processes」を参照してください。

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、assign アクティビティーで実行できます。例えば、変数は assign の to-spec に現れる必要があります。あるいは、Java 断片の内側で変数を初期化することが可能です。

標準障害の例外「missingReply」(メッセージ: CWWBE0071E)

microflow または長期実行プロセスを実行すると、BPEL 標準障害「missingReply」(メッセージ: CWWBE0071E) が発生するか、このエラーがシステム・ログまたは SystemOut.log ファイルで検出されます。

理由

両方向オペレーションでは応答を送信する必要があります。このエラーは、プロセスが reply アクティビティーをナビゲートせずに終了する場合に生成されます。この状態は、以下の事情の下で発生します。

- reply アクティビティーがスキップされた。

- 障害が発生し、対応する障害ハンドラーに reply アクティビティーが含まれていない。
- 障害が発生し、対応する障害ハンドラーが存在しない。

解決方法

モデルを訂正し、プロセスの終了前に reply アクティビティーが必ず実行されるようにします。

並列パスが順次化される

flow アクティビティーの内側に 2 つ以上の並列 invoke アクティビティーがありますが、invoke アクティビティーが順次実行されます。

解決方法

- 真の並列処理を実現するには、それぞれのパスを別々のトランザクションに置く必要があります。すべての並列 invoke アクティビティーの「トランザクションの振る舞い」属性を、「事前コミット (commit before)」または「所有が必要 (requires own)」に設定します。
- データベース・システムとして 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。

ネストされたデータ・オブジェクトを別のデータ・オブジェクトにコピーするとソース・オブジェクトの参照が破棄される

データ・オブジェクト Father には、別のデータ・オブジェクト Child が含まれます。 Java 断片またはクライアント・アプリケーションの内側では、Child を含むオブジェクトがフェッチされ、データ・オブジェクト Mother の副構造で設定されます。データ・オブジェクト Father での Child への参照は消失します。

理由

Child への参照は、Father から Mother に移されます。

解決方法

上記のようなデータ形式変更を Java 断片またはクライアント・アプリケーションで実行し、Father に参照を保存する場合は、データ・オブジェクトを、別のオブジェクトに割り当てられる前にコピーします。以下のコード断片はその方法を示しています。

```
BOCopy copyService = (BOCopy)ServiceManager.INSTANCE.locateService
                      ("com/ibm/websphere/bo/BOCopy");
DataObject Child = Father.get("Child");
DataObject BCopy = copyService.copy(Child);
Mother.set("Child", BCopy);
```

CScope が使用不可である

長期実行プロセスでの Microflow の開始またはナビゲーション・ステップの実行が失敗し、「事後条件違反 !(cscope != null) (postcondition violation !(cscope != null))」というアサーションが表示されます。

理由

特定の状態ではプロセス・エンジンは補正サービスを使用しますが、これは使用不可でした。

解決方法

管理についての PDF に記載のとおり、補正サービスを使用可能に設定します。

プロセス関連またはタスク関連メッセージの操作

ディスプレイに表示またはログ・ファイルに書き込まれる Business Process Choreographer メッセージの詳細情報を取得する方法について説明します。

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。

   以下のスクリプトのいずれかを実行します。

   - Windows システムの場合: *install_root*/bin/waslogbr.bat
   - Linux、UNIX、および i5/OS システムの場合: *install_root*/bin/waslogbr.sh
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 ファイル (Windows システム)
      - *profile_root*/profiles/*profile_name*/logs/activity.log file (Linux、UNIX、および i5/OS システム)
   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。

- 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、「technote 1166009」を参照してください。
- ヒューマン・タスク・アプリケーションにまだタスク・インスタンスがある間にそのアプリケーションを停止しようとすると、管理コンソールは、応答を停止します。 アプリケーションを停止するには、次のようにする必要があります。
  1. 新規インスタンスが作成されないようにするためにヒューマン・タスクを停止します。
  2. 以下のいずれかを実行します。
     - すべての既存のタスク・インスタンスが、正常な方法で終了するのを待つ。
     - すべてのタスク・インスタンスを強制終了して削除する。
  3. タスク・アプリケーションを停止します。
- 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」ループを用意しておくことをお勧めします。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。

- 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**」 → 「***HTMMailSession_server***」をクリックし、「**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「**アプリケーション・サーバー**」 → 「***server_name***」をクリックします。
  2. 「**ビジネス・インテグレーション**」の下で、「**Business Process Choreographer**」を展開し、「**Human Task Manager の構成**」をクリックします。
  3. 「**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「**アプリケーション・サーバー**」 → 「***server_name***」をクリックします。
2. 「**ビジネス・インテグレーション**」の下で、「**Business Process Choreographer**」を展開し、「**Human Task Manager の構成**」をクリックします。
3. 「**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「**アプリケーション・サーバー**」 → 「***server_name***」をクリックします。
  2. 「**ビジネス・インテグレーション**」の下で、「**Business Process Choreographer**」を展開し、「**Human Task Manager の構成**」をクリックします。
  3. 「**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**」をクリックします。
  4. **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。

- 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
- 担当者割り当てに関連したエラー・メッセージと警告メッセージ
- グループ作業項目および「グループ」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

『テクニカル・サポート検索』ページでも、追加情報を検索できます。

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・プロバイダーに使用される XSL 変換ファイルは、例えば `LDAPTransformation.xsl` です。このファイルは Linux、UNIX、および i5/OS プラットフォームでは、*install-root*/ProcessChoreographer/Staff ディレクトリーにあり、Windows プラットフォームでは、*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 20 です。この値を変更するには、次のようにします。

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">20</xsl:variable>
```

注: 大きな Threshold 値を指定すると、パフォーマンスが低下する場合があります。このため、100 より大きな値を指定しないでください。

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か、担当者照会結果のサブセットのみをすぐに最新表示する場合に、この方式を使用します。
- **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 93. 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。 EngineAdministratorCannotBe ResolvedException
プロセス・リーダー	リーダーなし	リーダーなし

表 94. 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

表 94. 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 (続き)

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...**
タスクの潜在的スターター	だれでも潜在的スターターになる	だれでも潜在的スターターになる
タスクの潜在的所有者	だれでも潜在的所有者になる	管理者が潜在的所有者になる
タスク・エディター	エディターなし	エディターなし
タスク・リーダー	継承のみが適用される	継承のみが適用される
エスカレーション受信者	管理者がエスカレーション受信側になる	管理者がエスカレーション受信側になる

次の継承規則がインライン・タスクに適用されます。

- プロセス管理者が、すべてのインライン・タスク、それらのサブタスク、後続タスク、およびエスカレーションの管理者になります。
- プロセス・リーダーが、すべてのインライン・タスク、それらのサブタスク、後続タスク、およびエスカレーションのリーダーになります。
- タスク管理者が、これらすべてのタスクのすべてのサブタスク、後続タスク、およびエスカレーションの管理者になります。
- タスク・リーダーが、これらすべてのタスクのすべてのサブタスク、後続タスク、およびエスカレーションのリーダーになります。
- 任意のタスク・ロールのメンバーが、このタスクのエスカレーション、サブタスク、および後続のタスクのリーダーになります。
- エスカレーション受信側が、エスカレートしたタスクのリーダーになります。

表 95. 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	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
編集者	エディターなし	エディターなし
リーダー	継承のみが適用される	継承のみが適用される
エスカレーション受信者	管理者がエスカレーション受信側になる	管理者がエスカレーション受信側になる

次の継承規則がスタンドアロン・タスクに適用されます。

- タスク管理者が、これらすべてのタスクのすべてのサブタスク、後続タスク、およびエスカレーションの管理者になります。
- タスク・リーダーが、これらすべてのタスクのすべてのサブタスク、後続タスク、およびエスカレーションのリーダーになります。
- 任意のタスク・ロールのメンバーが、このタスクのエスカレーション、サブタスク、および後続のタスクのリーダーになります。
- エスカレーション受信側が、エスカレートしたタスクのリーダーになります。

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リーダー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リーダー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、WebSphere Application Server セキュリティーが使用可能になっていない可能性があることを示しています。要員の許可を使用するヒューマン・タスクやプロセスでは、セキュリティーを使用可能にすることとユーザー・レジストリーを構成することが必要です。次のステップを実行します。

1. WebSphere セキュリティーが使用可能になっていることを確認します。管理コンソールで、「**セキュリティー**」 → 「**グローバル・セキュリティー**」に移動して、「**グローバル・セキュリティーを使用可能に設定 (Enable global security)**」 のチェック・ボックスを選択してあることを確認します。
2. ユーザー・レジストリーが構成されていることを確認します。管理コンソールで、「**セキュリティー**」 → 「**ユーザー・レジストリー**」に移動して、「**アクティブ・ユーザー・レジストリー (Active user registry)**」属性にチェック・マークを付けます。
3. 停止している場合は、アクティビティーを再始動します。

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**」 → 「**アプリケーション・サーバー**」 → 「***server name***」、または「**サーバー**」 → 「**クラスター**」 → 「***cluster name***」 をクリックし、「**ビジネス・インテグレーション**」の下で「**Business Process Choreographer**」を展開して、「**Human Task Manager**」 → 「**構成**」をクリックします。
2. 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、「**許可において大/小文字を無視 (Ignore case for authorization)**」オプションを選択したことが考えられます。この場合、オプションの選択を解除するか、LDAP グループ dn をすべて大文字で指定します。
- グループ・メンバーシップの変更がすぐに許可に反映されません。影響を受けたユーザーがまだログオンしていると、この状態が発生する場合があります。ユーザーのグループ・メンバーシップは、そのユーザーのログイン・セッションにキャッシュされ、デフォルトで 2 時間後に有効期限が切れます。ログイン・セッションの有効期限が切れるのを待つか (デフォルトは 2 時間)、アプリケーション・サーバーを再始動します。Human Task Manager 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。

Business Process Choreographer Explorer のトラブルシューティング

このトピックを参照して、Business Process Choreographer Explorer に関連した問題を解決します。

このタスクについて

以下の情報を使用して、Business Process Choreographer Explorer に関連した問題を解決します。

- ブラウザーを使って Business Process Choreographer Explorer にアクセスしようとすると、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` がサーバー上に実際にデプロイされ、実行されていることを確認します。
  - 管理コンソールのアプリケーションのページにある「デプロイメント記述子の表示」の下で、コンテキスト・ルートが Business Process Choreographer Explorer をセットアップしたときに使用したコンテキスト・ルートであることを確認します。
- Business Process Choreographer Explorer の使用中にエラー・メッセージが表示される場合は、エラー・ページで「**詳しくは、検索してください**」リンクをクリックします。 これにより、IBM 技術サポート・サイトでエラー・コードの検索を開始します。このサイトで提供される情報は英語のみです。Business Process Choreographer Explorer エラー・ページに表示されるエラー・メッセージ・コードをクリップボードにコピーします。エラー・コードの形式は `CWWBcnnnnc` で、`c` は文字を、`nnnn` は 4 桁の数値を示します。「WebSphere Process Server technical support」ページに進みます。エラー・コードを「**追加検索項目 (Additional search terms)**」フィールドに貼り付けて、「**実行 (Go)**」をクリックします。
- `StandardFaultException` を標準障害 `missingReply` (メッセージ CWWBE0071E) とともに取得した場合、これはプロセス・モデルに関する問題の症状です。この問題の解決方法について詳しくは、730 ページの『ビジネス・プロセスおよびヒューマン・タスクの管理のトラブルシューティング』を参照してください。
- Business Process Choreographer Explorer にログオンできるものの、一部の項目が表示されない場合、または特定の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。プロセス管理者またはタスク管理者ではないユーザー ID でログオンすると、すべての管理ビューおよびオプションは非表示または使用不可になります。
- WebSphere Integration Developer を使用して、ビジネス・プロセスに定義されている権限設定を確認または変更する。

- エラー・メッセージ CWWBU0001E: ″BFMConnection 関数の呼び出し時に通信エラーが発生しました。″ または ″HTMConnection 関数の呼び出し時に通信エラーが発生しました。″ このエラーは、ビジネス・プロセス・コンテナーまたはヒューマン・タスク・コンテナーが停止され、クライアントがサーバーに接続できなくなったことを示します。ビジネス・プロセス・コンテナーおよびヒューマン・タスク・コンテナーが実行中でアクセス可能であることを確認してください。ネストされた例外に、問題に関する詳細情報が含まれる場合があります。
- エラー・メッセージ WWBU0024E: 理由「命名例外」で、ローカルのビジネス・プロセス EJB への接続を確立できませんでした。(WWBU0024E: ″Could not establish a connection to local business process EJB with a reason ″Naming Exception″.) ビジネス・プロセス・コンテナーが実行していないときにユーザーがログオンしようとすると、このエラーがスローされます。アプリケーション BPEContainer_*InstallScope* が実行中であることを確認してください。ここで、*InstallScope* は cluster_name または hostname_servername です。

関連タスク

721 ページの『ビジネス・プロセスの実行のトラブルシューティング』
ここでは、ビジネス・プロセスの実行に関する共通の問題の解決方法について説明します。

Business Process Choreographer Observer のトラブルシューティング

Business Process Choreographer Observer 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

Business Process Choreographer Observer の「ようこそ」ページにイベントが何も表示されない。

理由および解決方法

Business Process Choreographer Observer データベースにイベントが何も含まれていないか、イベントがまだ変換されていません。

理由	解決方法
イベントは正しく変換されるが、Business Process Choreographer Observer に表示されない。	イベントを受信した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 Observer にイベントがまったく表示されない場合は、Business Process Choreographer Observer およびイベント・コレクターが同じデータ・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**」 → 「**エンタープライズ・アプリケーション**」をクリックした後、BPCObserver アプリケーションを選択し、「**リソース参照 (Resource references)**」をクリックします。 2. モジュールの「**ターゲット・リソースの JNDI 名 (Target Resource JNDI Name)**」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。
サーバーで CEI サービスが使用可能になっていないため、イベントがまったく受信されない。	管理コンソールで「**アプリケーション・サーバー**」 → 「***server***」 → 「**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。
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。	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 ロギングの使用可能化については、『Business Process Choreographer Observer のロギング可能化』を参照してください。
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。	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。
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。	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	解決方法
一部のイベントが変換されずに消去されたため、(全部ではなく) 一部のイベントが変換される。	Event Collector が受信したイベントの数が Business Process Choreographer Observer に表示される数より少ない場合は、以下のいずれかを実行します。 • 必要な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。イベントがイベント・セレクター・ストリングに一致しても、変換プログラムに認識されない場合、そのイベントは消去されます。 • 一部のイベントの配信が遅すぎる場合は、受信したイベントをプロセス・インスタンスに関連付けることができません。これを防ぐには、BPCEventTransformerToleranceTime の設定値を増加させます。詳しくは、『構成パラメーターの変更』を参照してください。
イベント変換プログラムが起動されていない。	Business Process Choreographer Event Collector を再始動して、イベント変換プログラムを起動してください。Event Collector のしきい値設定を低くすることもできますが、新しいイベントを作成して Event Collector を再起動する必要があります。Business Process Choreographer Event Collector の構成設定の変更について詳しくは、『構成パラメーターの変更』を参照してください。
イベントが生成され、CBE ブラウザーに表示されるが、Business Process Choreographer Observer には表示されない。理由は、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である。	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 (Event service)**」 → 「**イベント・サービス (Event services)**」 → 「**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。
Business Process Choreographer Event Collector の構成設定が不適切なため、Observer でデータを表示することができない。	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、『構成パラメーターの変更』を参照してください。
BFMEvents イベント・グループを定義する必要がある。	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 (Event service)**」 → 「**イベント・サービス (Event services)**」 → 「**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、グループ BFMEvents が存在することを確認します。 • グループが存在しない場合は、イベント・コレクター・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。

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 と Business Process Choreographer Observer のデプロイメントと構成を確認してください。構成設定を調べるには、setupEventCollector および setupObserver 構成スクリプトを呼び出します。Business Process Choreographer Event Collector の構成設定の変更方法についての詳細は、294 ページの『Business Process Choreographer Observer の構成パラメーターの変更』を参照してください。
- 次のように選択して、管理コンソールで Observer コンポーネントのトレース機能を使用可能にします。**「ロギングおよびトレース」 → server1 → 「診断トレース・サービス」 → 「ログ詳細レベルの変更」**。`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 BPCObserver アプリケーションを再始動します。

プロセス関連およびタスク関連の監査証跡情報の使用

ビジネス・プロセスおよびヒューマン・タスクのイベント・タイプとデータベース構造について説明します。

始める前に

ビジネス・プロセス・コンテナーとタスク・コンテナーの一方または両方で、ロギングを使用可能にする必要があります。

このタスクについて

ロギングが使用可能にされている場合、ビジネス・プロセスまたはヒューマン・タスクのナビゲーション・ステップごとに、情報が監査ログまたは Common Event Infrastructure (CEI) ログに書き込まれます。CEI について詳しくは、「*WebSphere Process Server* モニター」PDF を参照してください。 以降のトピックでは、ビジネス・プロセスおよびヒューマン・タスクのイベント・タイプとデータベース構造について説明します。

ビジネス・プロセスの監査イベント・タイプ

ここでは、ビジネス・プロセスの処理中に監査ログに書き込むことができるイベントのタイプについて説明します。

イベントをログに記録するには、次の条件を満たす必要があります。

- ビジネス・プロセス・コンテナー用の対応する監査ロギング・タイプが使用可能になっている
- プロセス・モデルの対応するエンティティー用にイベントが必ず使用可能になっている

次の表に、ビジネス・プロセスの実行中に発生する可能性がある監査イベントのコード一覧を示します。

表 96. プロセス・インスタンス・イベント

監査イベント	イベント・コード
PROCESS_STARTED	21000
PROCESS_SUSPENDED	21001
PROCESS_RESUMED	21002
PROCESS_COMPLETED	21004
PROCESS_TERMINATED	21005
PROCESS_RESTARTED	21019
PROCESS_DELETED	21020
PROCESS_FAILED	42001
PROCESS_COMPENSATING	42003
PROCESS_COMPENSATED	42004
PROCESS_TERMINATING	42009
PROCESS_FAILING	42010
PROCESS_CORRELATION_SET_INITIALIZED	42027
PROCESS_COMPENSATION_INDOUBT	42030
PROCESS_WORKITEM_DELETED	42041
PROCESS_WORKITEM_CREATED	42042
PROCESS_COMPENSATION_FAILED	42046
PROCESS_EVENT_RECEIVED	42047
PROCESS_EVENT_ESCALATED	42049
PROCESS_WORKITEM_TRANSFERRED	42056
PROCESS_PARTNER_CHANGED	42058

表 97. アクティビティー・イベント

監査イベント	イベント・コード
ACTIVITY_READY	21006
ACTIVITY_STARTED	21007
ACTIVITY_COMPLETED	21011
ACTIVITY_CLAIM_CANCELED	21021
ACTIVITY_CLAIMED	21022
ACTIVITY_TERMINATED	21027
ACTIVITY_FAILED	21080
ACTIVITY_EXPIRED	21081
ACTIVITY_LOOPED	42002
ACTIVITY_SKIPPED	42005
ACTIVITY_TERMINATING	42008
ACTIVITY_FAILING	42011
ACTIVITY_OUTPUT_MESSAGE_SET	42012
ACTIVITY_FAULT_MESSAGE_SET	42013
ACTIVITY_STOPPED	42015
ACTIVITY_FORCE_RETRIED	42031

表 97. アクティビティー・イベント (続き)

監査イベント	イベント・コード
ACTIVITY_FORCE_COMPLETED	42032
ACTIVITY_UNDO_STARTED	42033
ACTIVITY_UNDO_SKIPPED	42034
ACTIVITY_UNDO_COMPLETED	42035
ACTIVITY_MESSAGE_RECEIVED	42036
ACTIVITY_LOOP_CONDITION_TRUE	42037
ACTIVITY_LOOP_CONDITION_FALSE	42038
ACTIVITY_WORKITEM_DELETED	42039
ACTIVITY_WORKITEM_CREATED	42040
ACTIVITY_ESCALATED	42050
ACTIVITY_WORKITEM_REFRESHED	42054
ACTIVITY_WORKITEM_TRANSFERRED	42055
ACTIVITY_PARALLEL_BRANCHES_STARTED	42057

表 98. 変数関連イベント

監査イベント	イベント・コード
VARIABLE_UPDATED	21090

表 99. 制御リンク・イベント

監査イベント	イベント・コード
LINK_EVALUATED_TO_TRUE	21034
LINK_EVALUATED_TO_FALSE	42000

表 100. プロセス・テンプレート・イベント

監査イベント	イベント・コード
PROCESS_INSTALLED	42006
PROCESS_UNINSTALLED	42007

表 101. スコープ・インスタンス・イベント

監査イベント	イベント・コード
SCOPE_STARTED	42020
SCOPE_SKIPPED	42021
SCOPE_FAILED	42022
SCOPE_FAILING	42023
SCOPE_TERMINATED	42024
SCOPE_COMPLETED	42026
SCOPE_COMPENSATING	42043
SCOPE_COMPENSATED	42044
SCOPE_COMPENSATION_FAILED	42045
SCOPE_EVENT_RECEIVED	42048

表 101. スコープ・インスタンス・イベント (続き)

監査イベント	イベント・コード
SCOPE_EVENT_ESCALATED	42051

ヒューマン・タスクの監査イベント・タイプ

ここでは、ヒューマン・タスクの処理中に監査ログに書き込むことができるイベントのタイプについて説明します。

イベントをログに記録するには、次の条件を満たす必要があります。

- ヒューマン・タスク・コンテナー用の対応する監査ロギング・タイプが使用可能になっている
- タスク・モデルの対応するエンティティー用にイベントが必ず使用可能になっている

次の表に、ヒューマン・タスクの実行中に発生する可能性がある監査イベントのコード一覧を示します。

表 102. タスク・インスタンス・イベント

監査イベント	イベント・コード
TASK_CREATED	51001
TASK_DELETED	51002
TASK_STARTED	51003
TASK_COMPLETED	51004
TASK_CLAIM_CANCELLED	51005
TASK_CLAIMED	51006
TASK_TERMINATED	51007
TASK_FAILED	51008
TASK_EXPIRED	51009
TASK_WAITING_FOR_SUBTASK	51010
TASK_SUBTASKS_COMPLETED	51011
TASK_RESTARTED	51012
TASK_SUSPENDED	51013
TASK_RESUMED	51014
TASK_COMPLETED_WITH_FOLLOW_ON	51015
TASK_UPDATED	51101
TASK_OUTPUT_MESSAGE_UPDATED	51103
TASK_FAULT_MESSAGE_UPDATED	51104
TASK_WORKITEM_DELETED	51201
TASK_WORKITEM_CREATED	51202
TASK_WORKITEM_TRANSFERRED	51204
TASK_WORKITEM_REFRESHED	51205

表 103. タスク・テンプレート・イベント

監査イベント	イベント・コード
TASK_TEMPLATE_INSTALLED	52001
TASK_TEMPLATE_UNINSTALLED	52002

表 104. エスカレーション・インスタンス・イベント

監査イベント	イベント・コード
ESCALATION_FIRED	53001
ESCALATION_WORKITEM_DELETED	53201
ESCALATION_WORKITEM_CREATED	53202
ESCALATION_WORKITEM_TRANSFERRED	53204
ESCALATION_WORKITEM_REFRESHED	53205

ビジネス・プロセス用監査証跡データベース・ビューの構造

AUDIT_LOG_B データベース・ビューは、ビジネス・プロセスについての監査ログ情報を提供します。

監査証跡の内容を読み取るには、データベース表とビューの読み取りをサポートする SQL または他の管理ツールを使用します。

監査イベントは、プロセス・エンティティーと関連があります。監査イベントのタイプは、そのイベントが参照するエンティティーによって異なります。監査イベントには、次のタイプがあります。

- プロセス・テンプレート・イベント (PTE)
- プロセス・インスタンス・イベント (PIE)
- アクティビティー・インスタンス・イベント (AIE)
- 変数関連イベント (VAR)
- 制御リンク・イベント (CLE)
- スコープ関連イベント (SIE)

監査イベント・タイプのコード一覧については、 744 ページの『ビジネス・プロセスの監査イベント・タイプ』を参照してください。

以下の表に、AUDIT_LOG_B 監査証跡ビューの構造を示します。表では、列名とイベント・タイプをリストし、列について簡単に説明しています。

インライン・タスクは、AUDIT_LOG_B 監査証跡ビューに記録され、TASK_LOG 監査証跡ビューには記録されません。例えば、インライン参加タスクを要求すると ACTIVITY_CLAIMED イベントが生成されますが、タスク関連イベントは生成されません。

表 105. 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。

表 105. 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
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 105. 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 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。

表 105. 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
CONTROL_LINK_ NAME					x		現行リンク・イベントに関連したリンクの名前。
PRINCIPAL		x	x	x	x	x	プリンシパルの名前。 PROCESS_DELETED イベントの場合には設定されません。
VARIABLE_DATA				x			`variable updated`イベント用の変数のデータ。
EXCEPTION_TEXT		x	x			x	アクティビティーまたはプロセスが失敗する原因となった例外メッセージ。次の場合に適用されます。 PROCESS_FAILED ACTIVITY_FAILED SCOPE_FAILED
DESCRIPTION		x	x	x	x	x	潜在的に解決される可能性のある置換変数を含むアクティビティーまたはプロセスの説明。
CORR_SET_INFO		x					プロセスの開始時に初期化された相関セットのストリング表現。 processCorrelationSetInitialized イベント (42027) により出力されます。
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 105. 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・ネーム・スペース '}' ポート・タイプ名 ':' オペレーション名。このフィールドは、「onAlarm」イベントの場合には設定されません。

ヒューマン・タスク用監査証跡データベース・ビューの構造

TASK_AUDIT_LOG データベース・ビューは、ヒューマン・タスクについての監査ログ情報を提供します。

インライン・タスクは AUDIT_LOG_B ビューに記録されます。他のすべてのタスク・タイプは TASK_AUDIT_LOG ビューに記録されます。

監査証跡の内容を読み取るには、データベース表とビューの読み取りをサポートする SQL または他の管理ツールを使用します。

監査イベントは、タスク・エンティティーと関連があります。監査イベントのタイプは、そのイベントが参照するエンティティーによって異なります。監査イベントには、次のタイプがあります。

- タスク・インスタンス・イベント (TIE)
- タスク・テンプレート・イベント (TTE)
- エスカレーション・インスタンス・イベント (EIE)

以下の表に、TASK_AUDIT_LOG 監査証跡ビューの構造を示します。表では、列名とイベント・タイプをリストし、列について簡単に説明しています。

インライン・タスクは、AUDIT_LOG_B 監査証跡ビューに記録され、TASK_AUDIT_LOG 監査証跡ビューには記録されません。例えば、インライン参加タスクを要求すると ACTIVITY_CLAIMED イベントが生成されますが、タスク関連イベントは生成されません。

表 106. TASK_AUDIT_LOG 監査証跡ビューの構造

名前	TIE	TTE	EIE	説明
ALID	x	x	x	監査ログ・エントリーの ID。
AUDIT_EVENT	x	x	x	発生したイベントのタイプ。監査イベント・コードの一覧については、747 ページの『ヒューマン・タスクの監査イベント・タイプ』を参照してください。
CONTAINMENT_ CTX_ID	x	x		収容コンテキストの ID (ACOID、PTID、または PIID など)。
DESCRIPTION	x		x	解決された記述ストリング。記述のプレースホルダーは現行値によって置き換えられます。影響を受けたすべての言語は、XML 文書としてフォーマット設定されて、この列に一緒に記録されます。 create-like イベントのプレースホルダーを含む記述を持つか、update-like イベント用に明示的に更新された言語のみが記録されます。
ESIID			x	現行イベントに関連したエスカレーション・インスタンスの ID。
ESTID			x	現行イベントに関連したエスカレーション・テンプレートの ID。
EVENT_TIME	x	x	x	イベントが発生したときの時刻 (協定世界時 (UTC) 形式)。
FAULT_NAME	x			障害メッセージの名前。この属性は、次のイベントの場合に適用されます。 TASK_FAILED TASK_FAULT_MESSAGE_UPDATED
FAULT_NAME_SPACE	x			障害メッセージ・タイプのネーム・スペース。この属性は、次のイベントの場合に適用されます。 TASK_FAILED TASK_FAULT_MESSAGE_UPDATED
FOLLOW_ON_TKIID	x			後続のタスク・インスタンスの ID。

表 106. TASK_AUDIT_LOG 監査証跡ビューの構造 (続き)

名前	TIE	TTE	EIE	説明
MESSAGE_DATA	x			新規に作成または更新された入力、出力、または障害メッセージの内容。
NAME	x	x	x	イベントに関連付けられたタスク・インスタンス、タスク・テンプレート、またはエスカレーション・インスタンスの名前。
NAMESPACE	x	x		イベントに関連付けられたタスク・インスタンス、タスク・テンプレート、またはエスカレーション・インスタンスのネーム・スペース。
NEW_USER				転送または作成された作業項目の新規所有者。USERS フィールドによってこの値が使用可能になる場合、この値は null になることがあります。フィールド USERS も参照してください。この属性は次のイベントに適用されます。
	x			TASK_WORKITEM_CREATED
	x			TASK_WORKITEM_TRANSFERRED
			x	ESCALATION_WORKITEM_CREATED
			x	ESCALATION_WORKITEM_TRANSFERRED
OLD_USER				転送済み作業項目の前の所有者。この属性は、次のイベントの場合に適用されます。
	x			TASK_WORKITEM_TRANSFERRED
	x			TASK_WORKITEM_DELETED
			x	ESCALATION_WORKITEM_TRANSFERRED
			x	ESCALATION_WORKITEM_DELETED
PARENT_CONTEXT_ID	x			タスクの親コンテキスト (例えば、アクティビティー・テンプレートやタスク・インスタンス) の ID。サブタスクおよび後続タスクの場合にのみ設定されます。
PARENT_TASK_NAME	x			親タスク・インスタンスまたはテンプレートの名前。サブタスクおよび後続タスクの場合にのみ設定されます。
PARENT_TASK_NAMESP	x			親タスク・インスタンスまたはテンプレートのネーム・スペース。サブタスクおよび後続タスクの場合にのみ設定されます。
PARENT_TKIID	x			親タスク・インスタンスの ID。
PRINCIPAL	x	x	x	出したリクエストによってイベントがトリガーされたプリンシパルの名前。
TASK_KIND	x	x		タスクの種類。考えられる値は次のとおりです。 KIND_HUMAN 101 KIND_ORIGINATING 103 KIND_PARTICIPATING 105 KIND_ADMINISTRATIVE 106

表 106. TASK_AUDIT_LOG 監査証跡ビューの構造 (続き)

名前	TIE	TTE	EIE	説明
TASK_STATE	x			タスクまたはタスク・テンプレートの状態。タスク・テンプレートの場合に考えられる値は次のとおりです。 STATE_STARTED 1 STATE_STOPPED 2 タスク・インスタンスの場合に考えられる値は次のとおりです。 '1' :'STATE_INACTIVE' '2' :'STATE_READY' '3' :'STATE_RUNNING' '5' :'STATE_FINISHED' '6' :'STATE_FAILED' '7' :'STATE_TERMINATED' '8' :'STATE_CLAIMED' '12' :'STATE_EXPIRED' '101':'FORWARDED'
TKIID	x		x	タスク・インスタンスの ID。
TKTID	x	x		タスク・テンプレートの ID。
TOP_TKIID	x			トップ・タスク・インスタンスの ID。
USERS	x		x	タスクまたはエスカレーション作業項目に割り当てられた新規ユーザー ID。NEW_USER フィールドによってこの値が使用可能になる場合、この値は null になることがあります。この属性が適用されるイベントのリストについては、フィールド NEW_USER を参照してください。
VALID_FROM		x		現行イベントに関連したタスク・テンプレートの有効開始日付。

表 106. TASK_AUDIT_LOG 監査証跡ビューの構造 (続き)

名前	TIE	TTE	EIE	説明
WORK_ITEM_REASON	x		x	作業項目の割り当て理由。考えられる値は次のとおりです。 POTENTIAL_OWNER 1 EDITOR 2 READER 3 OWNER 4 POTENTIAL_STARTER 5 STARTER 6 ADMINISTRATOR 7 POTENTIAL_SENDER 8 ORIGINATOR 9 ESCALATION_RECEIVER 10 POTENTIAL_INSTANCE_CREATOR 11 理由は、作業項目に関連したすべてのイベントの場合に設定されます。例えば、ESCALATION_RECEIVER はエスカレーション作業項目に関連したイベントの場合に設定される一方で、他の理由はタスク作業項目関連のイベントに適用されます。

第 8 部 付録

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-8711
東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物にも、次のように、著作権表示を入れていただく必要があります。「(C) (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 (c) Copyright IBM Corp. _年を入れる_. All rights reserved.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

プログラミング・インターフェース情報は、プログラムを使用してアプリケーション・ソフトウェアを作成する際に役立ちます。

一般使用プログラミング・インターフェースにより、お客様はこのプログラム・ツール・サービスを含むアプリケーション・ソフトウェアを書くことができます。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

警告: 診断、修正、調整情報は、変更される場合がありますので、プログラミング・インターフェースとしては使用しないでください。

商標

IBM、IBM logo、AIX、Cloudscape、DB2、DB2 Connect、DB2 Universal Database、i5/OS、Informix、iSeries、MQSeries、WebSphere、および z/OS は、International Business Machines Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

Microsoft および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

この製品には、Eclipse Project (http://www.eclipse.org) により開発されたソフトウェアが含まれています。

IBM WebSphere Process Server for Multiplatforms バージョン 6.1.0

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12