BEA WebLogic Integration™

BPM ユーザーズ ガイド

リリース 7.0
マニュアルの日付 : 2002 年 6 月

著作権

限定的権利条項

商標または登録商標

WebLogic Integration BPM ユーザーズ ガイド

パート番号	日付	ソフトウェアのバージョン
なし	2002 年 6 月	7.0

目次

このマニュアルの内容

1. Business Process Management とサンプル ワークフローの紹介

2. WebLogic Integration Studio 入門

3. ワークフロー オブジェクトとプロパティ

4. ワークフローのノードの定義

5. ワークフローの作成

6. カスタム例外ハンドラの使い方

7. サンプル ワークフローの実行とモニタ

このマニュアルの内容

目的

このマニュアルは、WebLogic Integration で Business Process Management (BPM) を処理するためのチュートリアルです。WebLogic Integration Studio と Worklist アプリケーションを使用してチュートリアルを進め、そこで、3 つのワークフローのサンプルを使用して、単純な受注シナリオに基づいたワークフローの定義と実行プロセスを説明します。Order Processing Trigger、Order Processing、Order Fulfillment という 3 つのワークフローは、インストール先の *SAMPLES_HOME*`/integration/samples/bpm_tutorial` フォルダ内の `Tutorial.jar` パッケージ ファイルに収納されています。

このチュートリアルで説明する手順に従って、WebLogic Integration Business Process Management コンポーネントの基本概念の多くを習得し、Studio の機能を最大限に利用してください。ただし、このマニュアルはリファレンスでもなくユーザ ガイドでもないので、Studio や Worklist のすべての機能を説明しているわけではありません。このマニュアルで言及する特定のテーマに関する詳細については、『*WebLogic Integration Studio ユーザーズ ガイド*』と『*WebLogic Integration Worklist ユーザーズ ガイド*』を参照してください。

注意： Worklist クライアント アプリケーションは WebLogic Integration リリース 7.0 より非推奨となりました。代替機能に関する詳細については、『*WebLogic Integration リリース ノート*』を参照してください。

対象読者

このマニュアルは、ビジネス アナリスト、ワークフローの設計者など、WebLogic Integration Studio を使用してワークフローの定義および管理を行うユーザを対象としています。このマニュアルは、技術者と技術者ではないユーザの両方を同様に対象としています。

このマニュアルは、Java 2 Enterprise Edition (J2EE) プラットフォーム、Enterprise JavaBeans、BEA Weblogic Server、eXtensible Markup Language (XML)、XPath 言語についてある程度の知識があることを前提として書かれています。

構成

このマニュアルの内容は以下のとおりです。

第 1 章「Business Process Management とサンプル ワークフローの紹介」では、チュートリアルで使用するシナリオとサンプル ワークフローの概要について説明し、Business Process Management とワークフローの概念を紹介します。

第 2 章「WebLogic Integration Studio 入門」では、Studio のユーザ インタフェースを紹介し、チュートリアルのパッケージ ファイルをインポートする手順について説明します。

第 3 章「ワークフロー オブジェクトとプロパティ」では、Order Processing Trigger ワークフローを使用して、WebLogic Integration ワークフロー オブジェクトの使い方を例示し、あらかじめ定義されているワークフローの編集について実習します。

第 5 章「ワークフローの作成」では、Order Processing ワークフローの作成方法とセットアップの手順を習得します。

第 4 章「ワークフローのノードの定義」では、Order Processing ワークフローのすべてのオブジェクトを定義する手順を、始めから終わりまで順番に説明します。

第 6 章「カスタム例外ハンドラの使い方」では、Order Fulfillment ワークフローを使用して、Studio の拡張機能の 1 つを紹介します。

第 7 章「サンプル ワークフローの実行とモニタ」では、サンプル ワークフローを Worklist アプリケーションで実行し、同時にそのワークフローの進行を Studio でモニタする手順を説明します。

e-docs Web サイト

BEA 製品のドキュメントは、BEA Systems, Inc. の Web サイトで入手できます。BEA のホーム ページで [製品のドキュメント] をクリックするか、または「e-docs」という製品ドキュメント ページ（`http://edocs.beasys.co.jp/e-docs/index.html`）を直接表示してください。

このマニュアルの印刷方法

Web ブラウザの [ファイル | 印刷] オプションを使用すると、Web ブラウザからこのマニュアルを一度に 1 ファイルずつ印刷できます。

このマニュアルの PDF 版は、Web サイトで入手できます。WebLogic Integration PDF を Adobe Acrobat Reader で開くと、マニュアルの全体（または一部分）を書籍の形式で印刷できます。PDF を表示するには、WebLogic Integration ドキュメントのホーム ページを開き、[PDF 版] ボタンをクリックして、印刷するマニュアルを選択します。

Adobe Acrobat Reader がない場合は、Adobe の Web サイト（`http://www.adobe.co.jp/`）で無料で入手できます。

関連情報

次の WebLogic Integration マニュアルには、WebLogic Integration の BPM コンポーネントに関する詳細な情報が記載されています。

- *WebLogic Integration Studio ユーザーズ ガイド*
- *WebLogic Integration Worklist ユーザーズ ガイド*
- *BPM クライアント アプリケーション プログラミング ガイド*
- *WebLogic Integration BPM プラグイン プログラミング ガイド*

サポート情報

WebLogic Integration のドキュメントに関するユーザからのフィードバックは弊社にとって非常に重要です。質問や意見などがあれば、電子メールで **docsupport-jp@bea.com** までお送りください。寄せられた意見については、WebLogic Integration のドキュメントを作成および改訂する BEA の専門の担当者が直に目を通します。

電子メールのメッセージには、ご使用の WebLogic Integration のリリースをお書き添えください。

本バージョンの BEA WebLogic Integration について不明な点がある場合、または BEA WebLogic Integration のインストールおよび動作に問題がある場合は、BEA WebSupport (**websupport.bea.com/custsupp**) を通じて BEA カスタマ サポートまでお問い合わせください。カスタマ サポートへの連絡方法については、製品パッケージに同梱されているカスタマ サポート カードにも記載されています。

カスタマ サポートでは以下の情報をお尋ねしますので、お問い合わせの際はあらかじめご用意ください。

- お名前、電子メール アドレス、電話番号、ファクス番号
- 会社の名前と住所
- お使いの機種とコード番号
- 製品の名前とバージョン
- 問題の状況と表示されるエラー メッセージの内容

表記規則

このマニュアルでは、全体を通して以下の表記規則が使用されています。

表記法	適用
〔Ctrl〕+〔Tab〕	複数のキーを同時に押すことを示す。
斜体	強調または書籍のタイトルを示す。

等幅テキスト	コード サンプル、コマンドとそのオプション、データ構造体とそのメンバー、データ型、ディレクトリ、およびファイル名とその拡張子を示す。等幅テキストはキーボードから入力するテキストも示す。
	例
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

斜体の等幅テキスト	コード内の変数を示す。 *例* `String` ***`expr`***
すべて大文字のテキスト	デバイス名、環境変数、および論理演算子を示す。 *例* LPT1 SIGNON OR
{ }	構文の中で複数の選択肢を示す。実際には、この括弧は入力しない。

表記法	適用
[ ]	構文の中で任意指定の項目を示す。実際には、この括弧は入力しない。 *例* `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
|	構文の中で相互に排他的な選択肢を区切る。実際には、この記号は入力しない。
...	コマンドラインで以下のいずれかを示す。 ■ 引数を複数回繰り返すことができる。 ■ 任意指定の引数が省略されている。 ■ パラメータや値などの情報を追加入力できる。 実際には、この省略記号は入力しない。 *例* `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
. . .	コード サンプルまたは構文で項目が省略されていることを示す。 実際には、この省略記号は入力しない。

1 Business Process Management とサンプル ワークフローの紹介

このチュートリアルで紹介するサンプル ワークフローは `SAMPLES_HOME/integration/samples/bpm_tutorial` フォルダ内の `Tutorial.jar` ファイルに入っており、WebLogic Integration の Business Process Management (BPM) の最も一般的な機能について説明しています。これらのサンプル ワークフローは実際に使用されているものではありませんが、次に挙げる Business Process Management の 2 つのクライアント アプリケーションを実際に使用することで、ごく単純化されたシナリオを体験することができます。そのクライアント アプリケーションは、Studio (設計環境) と Worklist (実行環境) の 2 つです。

注意： Worklist クライアント アプリケーションは WebLogic Integration リリース 7.0 より非推奨となりました。代替機能に関する詳細については、『*WebLogic Integration リリース ノート*』を参照してください。

この概要では、サンプル ワークフローのコンテキストを説明し、全体像を理解した後で、チュートリアルの残りの各章でそれを組み合わせて完成できるようにします。特に、この節では以下のことについて説明します。

- 「シナリオの概要」では、基本的なビジネス ロジックによって、ワークフローの*機能的*要件を説明します。フローチャートを使用して関連するプロセスを例示します。
- 「BPM 環境の概要」では、ビジネス マネジメント コンポーネントの*アプリケーション*コンテキストについて説明し、ワークフローによって統合される多くのアプリケーション コンポーネント間のワークフローのロールを明確に位置付けられるようにします。
- 「WebLogic Integration におけるシナリオのモデル化」では、サンプル ワークフローで使用する外部コンポーネント、内部オブジェクト、インタフェースを紹介します。実際に使用可能なコンポーネントと、コンポーネントを実装、またはシミュレートする方法を例題の中で説明します。

- 最後に、「サンプル ワークフローを使った作業」と「チュートリアルの使い方」で、インストールしたサンプル ファイルの使い方と、チュートリアル自体に関する実用的な情報について説明します。

シナリオの概要

このサンプル ワークフローは、一般的な Web ベースでの受注販売のシナリオに、だいたい基づいています。いったん受注すると、承認、在庫確認、出荷、請求という段階を経ながら、ワークフローによって受注の実行と処理が制御されます。次に、このシナリオを構成するプロセスとデータについて説明します。

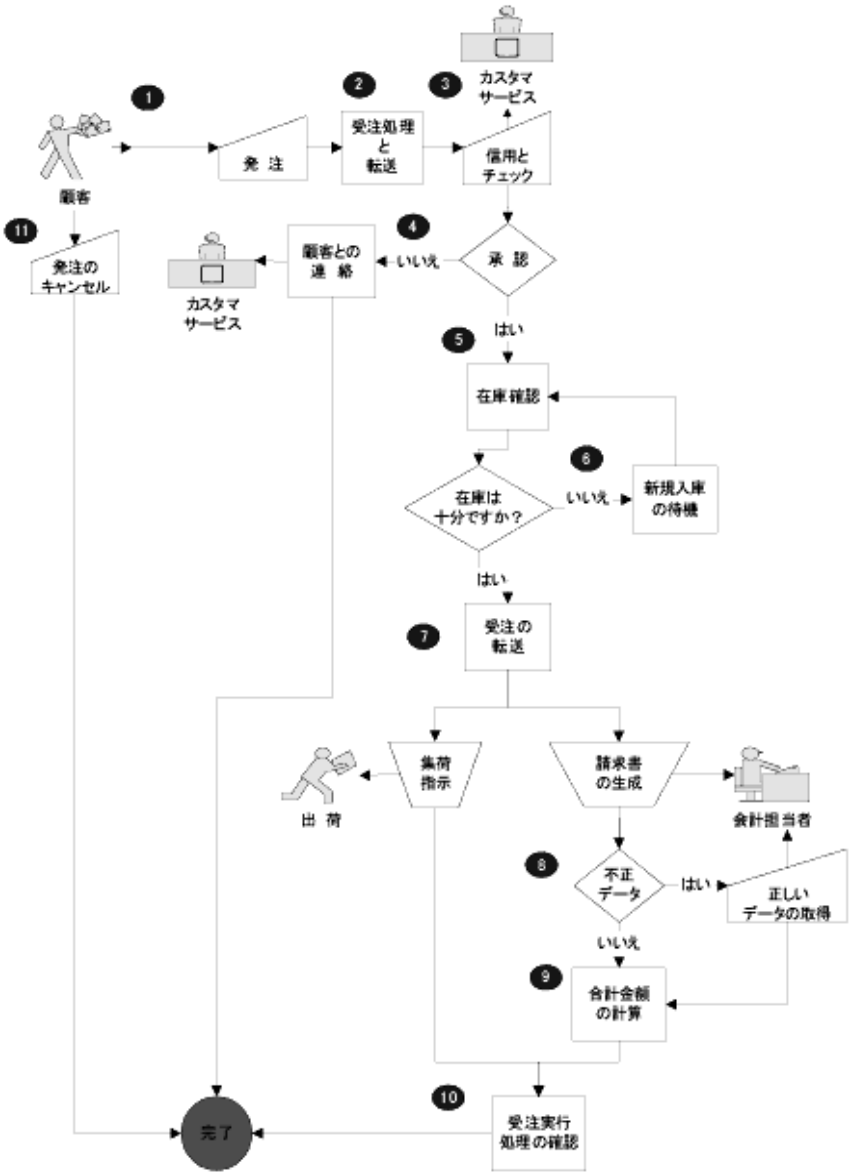
1. 顧客が CDExpress という音楽メディア小売店にブラウザ ベースの発注フォームなどのオンライン メカニズムを使用して商品を発注します。その顧客は、氏名、連絡用 E メール アドレス、出荷先の国または州名、希望商品（ID および商品）、発注数量を指定します。
2. 処理システムによってデータが読み取られ、ID 番号が付けられて受理されます。
3. その受注はカスタマサービスに転送され、顧客の信用情報がチェックされます。
4. 信用チェックができなかった場合、カスタマサービスが顧客に通知し、正しい信用情報を取得する必要があります。これ以降のプロセスは手動で行います。
5. 信用チェックが通ると、商品 ID を基に受注商品の現在の目録がデータベースでチェックされ、出荷可能な商品の数量と受注数量が比較されます。
6. 在庫が十分ではない場合は、次の新規入庫の到着までその受注は保留されます。新規入庫の在庫表が受領されると、その在庫が受注に対応可能であると確認できるまで、手順 5 が繰り返されます。
7. 十分な在庫がある場合、その受注は出荷業務を行う出荷担当者と、請求書の発行をシステムに指示する会計担当者に同時に転送されます。
8. 国または州の売上税を含め、請求書の合計金額の計算に必要な入力の処理中にエラーが発生した場合、請求プロセスを開始した会計担当者に通知され、正しい情報の入力が求められます。
9. 受注に対する合計金額が算出されます。

10. 次に、受注品が出荷されたことがシステムで確認され、顧客に E メールで通知されます。

11. 出荷前のトランザクションにおけるどの時点でも、顧客からの通知により受注をキャンセルできます。

このワークフローがどのようなものかを理解するための基本的なフローチャートは次の図のとおりです。各セクションにつけられた番号は、これまで説明してきた受注販売の各手順に対応しています。

図 1-1 受注販売のシナリオ

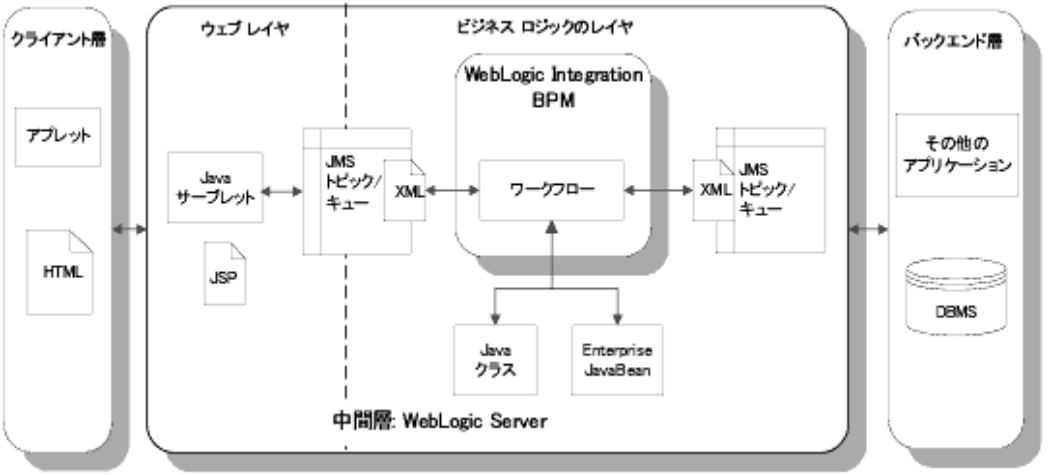


BPM 環境の概要

Studio のワークフローを使用して、たとえば、クライアントへの E メールの送信や手動で行う作業の Worklist ユーザーへの割り当てなどの一定のビジネス プロセスを直接実行できますが、ほとんどの場合は、*外部のソフトウェア コンポーネントを統合するワークフローを作成し、このような外部のプログラム モジュールの呼び出しシーケンスを制御するようにワークフローを定義します。したがって、ワークフローを作成する前に、Enterprise JavaBeans、Java クラスといった、インタフェースが必要なアプリケーションについて外部コンポーネントを識別する必要があります。

このシナリオに必要な外部コンポーネントの識別を容易にするため、はじめに WebLogic Server と Java 2 Enterprise Edition (J2EE) モデル ビュー コントローラ多層アプリケーションの中間層との関連における WebLogic Integration のロールを検討します。分散 J2EE/WebLogic Server アプリケーション内での、このシナリオの実行に必要なその他のコンポーネントとの関連における WebLogic Integration サーバを次の図に示します。

図 1-2　WebLogic Integration BPM の環境

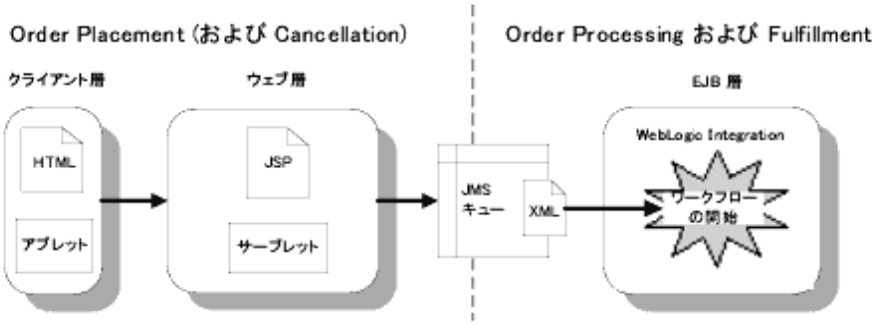


BPM は J2EE フレームワークのビジネス ロジック レイヤで図のように動作します。このことは、ワークフローが Java クラスや EJB (Enterprise JavaBeans) などのビジネス オブジェクトを直接統合する一方、他のコンポーネントやアプリケーションとの通信には、JMS (Java Message Service) の中間コードを通じて XML （eXtensible Markup Language) 言語のメッセージを交換することを意味します。

実際には、外部クライアントからのインプットの受信やフォーマットを行うフロントエンド アプリケーション全体は BPM ワークフローには表れません。Web コンポーネントと EJB レイヤとのやりとりが必要な場合に、ワークフローが処理を管理します。このように、ワークフローの開始は外部イベント、すなわち JMS トピック上へ該当する XML ドキュメントが到達することによって、通常、図のようにトリガされます。

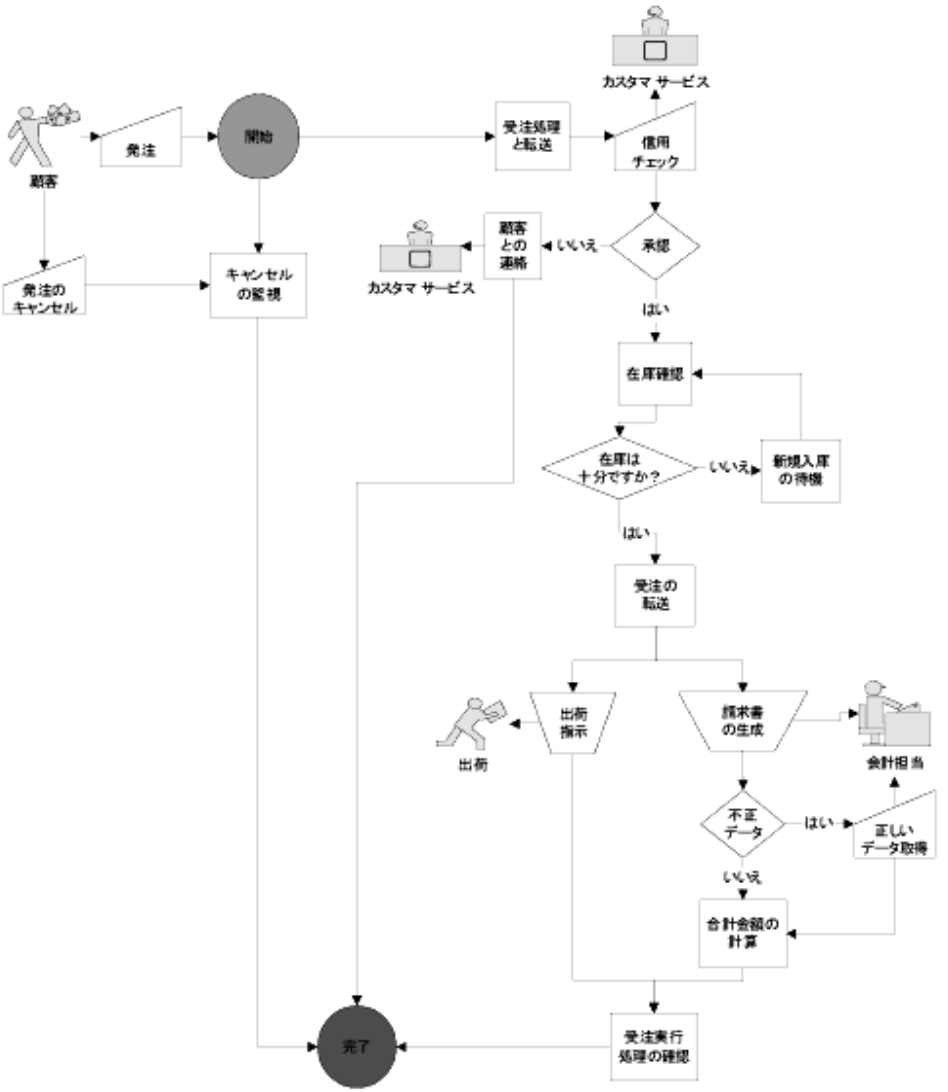
このフレームワークでは、通常 BPM ワークフローにこのシナリオの発注プロセスは含まれません。チュートリアルの目的上、このプロセスをシミュレートします（詳細については、「「サンプル ワークフローを使った作業」」を参照してください)。実際は、次の図で示すように、受注が処理プロセスに渡された時点が、ワークフローの開始となります。

図 1-3 受注販売のシナリオ：開始のメカニズム



シナリオのフローチャートを更新し、ワークフローの開始点や、キャンセル通知のインターセプトに必要な追加手順をより正確に表します。

図 1-4　受注販売のシナリオ：開始と終了



WebLogic Integration におけるシナリオのモデル化

シナリオの定義と WebLogic Integration BPM 環境の確認を行ったあと、実際のプロセスとシナリオ中のエンティティを、WebLogic Integration Studio が生成したワークフローのモデル構成にマップする方法を決める必要があります。マップ方法の決定には、以下の点について確認が必要です。

- アプリケーションのユーザ
- アプリケーションの機能を動作させるために、ワークフローからの起動が必要な外部コンポーネント
- シナリオの実行に必要な各種ワークフロー オブジェクト
- ワークフローと外部アプリケーションとのやりとりに必要なインタフェース

これらのモデル化の作業については、以下の節で説明します。

アプリケーション ユーザの定義

実際にワークフローをモデル化する前に、まず対象となる実際の担当者の観点からシステムを定義する必要があります。Studio は、用法のコンテキストを明確にする 3 つの構成要素、すなわち オーガニゼーション、ユーザ、ロールを定義します。これらの構成要素について次の節で解説し、それぞれがチュートリアルのシナリオでどのように使用されるかについても説明します。

オーガニゼーション

オーガニゼーションとは、企業やサイト ロケーションを表します。オーガニゼーションは、Studio 内で作成する Business Process Management 固有の構成要素であり、WebLogic Server で定義されるユーザまたはグループには対応しません。しかし、ユーザをオーガニゼーションに関連付けて、ロール（「ロール」を参照）をオーガニゼーション内に定義することができます。ワークフローは、単一のオーガニゼーションだけではなく複数のオーガニゼーションにも関連付けることができます。

このシナリオのオーガニゼーションは CDExpress という予め定義されたオーガニゼーションです。

ユーザ

ユーザとは、Studio と Worklist クライアント アプリケーションのユーザであり、WebLogic Server のセキュリティ レルムで定義されます。ユーザは Studio で作成され、少なくとも 1 つのデフォルトのオーガニゼーションに関連付けられる必要があります。ユーザはどのオーガニゼーションと関連付けられたワークフローにもアクセス可能ですが、実行可能なワークフローはユーザを割り当てたオーガニゼーション内のワークフローに限ります。ユーザはロールにも関連付けできます(「ロール」を参照)。

ここで使用する CDExpress というオーガニゼーションには、Studio で作成した、joe、mary、admin という 3 人のデフォルト ユーザがいます。

ユーザとパーミッションの詳細は、『*WebLogic Integration Studio ユーザーズ ガイド*』の「データ管理」の中の「ユーザの保守」を参照してください。

ロール

ロールとは、個人のユーザ アカウントとは無関係にユーザが実行できる汎用的なビジネス機能のことで、たとえば、部門、職務、役職などがロールとなる場合もあります。このシナリオでは、カスタマサービス担当者、会計、出荷がロールに相当します。

ロールはオーガニゼーション内に含まれるので、それぞれのオーガニゼーションは固有なロールのセットを持つことになります。Studio でロールを作成する際に、自動的に WebLogic Server グループを作成できても、WebLogic Server 上でコンフィグレーションされたグループにロールをマップする必要があります。

また、1 ユーザに対して 1 ロール、複数ユーザに対して 1 ロール、1 ユーザに対して複数のロールを関連付けできます。つまり、1 人のユーザが複数のロールを受け持ったり、あるいは 1 つのロールを数人で担う場合もあります。

CDExpress のオーガニゼーションには次のようなロール、マッピング、メンバー ユーザが存在し、それらはすべて予め定義されています。

表 1-1 CDExpress のデフォルトのロール、グループ、メンバー ユーザ

ロール	WebLogic Server グループへのマッピング	メンバー ユーザ
CustomerService	CustomerServiceCDE	admin
Accounting	AccountingCDE	admin joe
Shipping	ShippingCDE	admin mary

ロールとパーミッションの詳細は、『*WebLogic Integration Studio ユーザーズ ガイド*』の「データ管理」の中の「ロールを保守する」を参照してください。

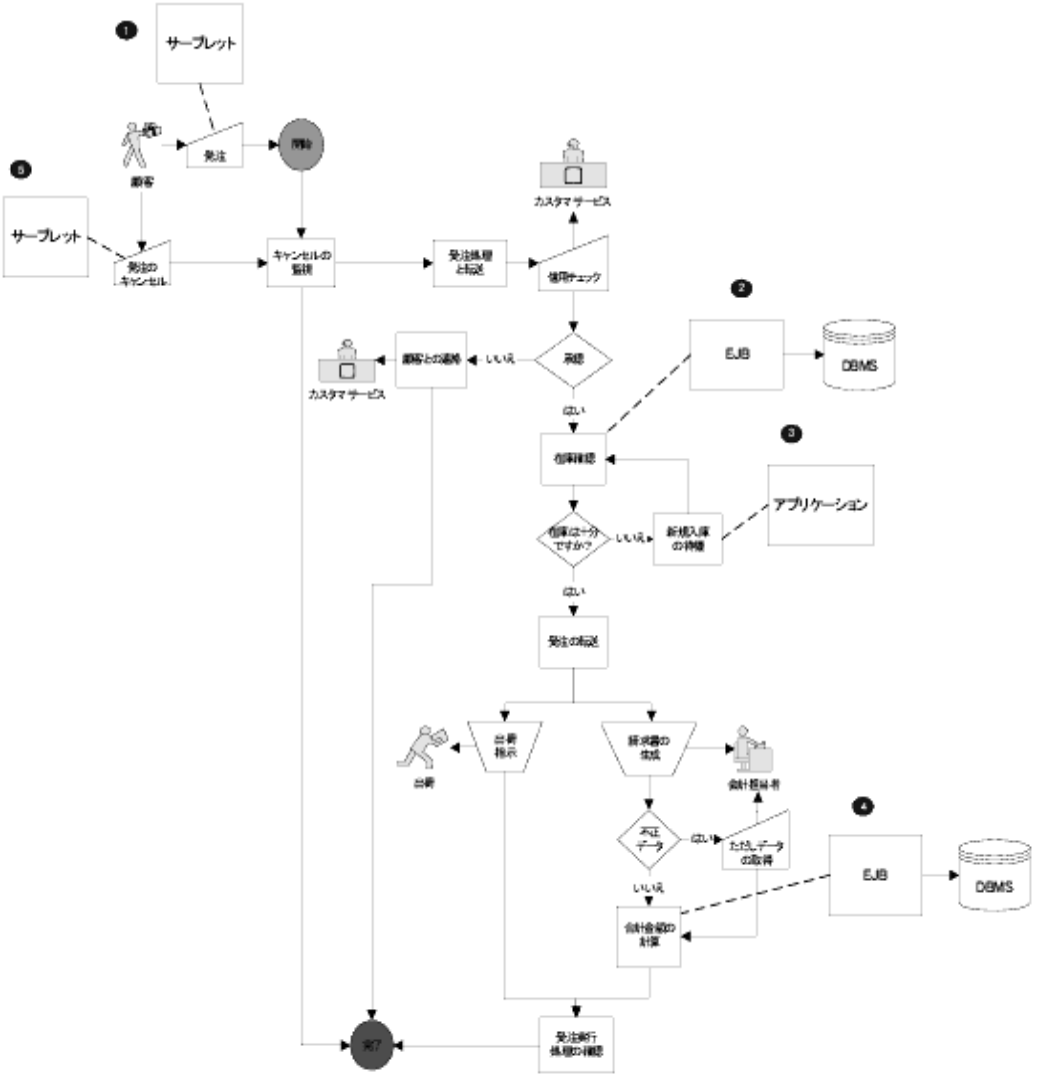
外部コンポーネントの指定

このシナリオでは、目的の作業を遂行するワークフロー内で統合の必要がある Web とバックエンドのコンポーネントを、以下のように指定できます。

- 顧客が使用するフロントエンド アプリケーションからの発注やキャンセル要求を処理するサーブレット
- 受注の合計金額を計算するメソッドと、データベース内の在庫をチェックするメソッドがある、セッション EJB
- 受注商品の新規入庫を報告するエンタープライズ アプリケーション

ここでは、ワークフローが外部コンポーネントとインタフェースをとる方法の詳細は省きますが（「ワークフロー インタフェースの定義」で説明）、この基本のワークフローが、どの時点で外部コンポーネントを必要とするかを次の図に示します。

図 1-5　受注販売のシナリオ：外部コンポーネント



サンプル ワークフローでは、他のコンポーネントに対するインタフェースがワークフローで定義されていますが、上の図の中で識別された EJB レベルのコンポーネントだけが組み込まれて実行されます。したがって、サンプル ワークフローでは外部コンポーネントを次のように表します。

1. プロセス全体を起動するフロントエンドのクライアント アプリケーションまたはサーブレットは供給されません。このモジュールは「サンプル ワークフローを使った作業」で説明する別のワークフローでシミュレートされます。
2. 在庫チェックのメソッドとして、WebLogic Integration の `lib` ディレクトリに、セッション EJB の `POBean.jar` が用意されています。実際のアプリケーションにおいても、EJB はデータベースをクエリします。このサンプルでは、商品 ID のデータベースを Bean クラスにハードコード化した値の配列によってシミュレートします。

   注意： `POBean` EJB は、WebLogic Integration サーバが起動したときに自動的に WebLogic Server 上にデプロイされます。
3. 実際のアプリケーションは在庫の報告には使用されず、そのアプリケーションに対するインタフェースがサンプル ワークフロー内に作成されます。詳細については、4-51 ページの「イベントの作成 : Wait for New Inventory イベントの定義」を参照してください。
4. 合計金額を計算するメソッドは、`POBean.jar` セッション EJB に用意されています。実際のアプリケーションにおいても、EJB はデータベースをクエリします。このサンプルでは、合衆国各州のデータベースを、Bean クラスにハードコード化した値の配列によってシミュレートします。
5. 発注のキャンセル プロセスのためのコンポーネントは用意されていませんが、そのようなコンポーネントに対するインタフェースがサンプル ワークフローに作成されます。詳細については、4-71 ページの「イベントの作成 : Watch for Cancellation イベントの定義」を参照してください。

ワークフロー オブジェクトの定義

アプリケーションで使用する外部コンポーネントが識別されると、ワークフローに必要なオブジェクトを定義して、シナリオ プロセスのモデル化を始めることができます。ワークフロー オブジェクトと、各オブジェクトがサンプル ワークフローにおいて果たす機能の例を以下の節で紹介します。

ノードと接続

ノードとは、ワークフローを構築し、ビジネス プロセスの境界を示す主要な単位です。接続によってノード間の移行が設定されます。次の表に示す 7 タイプのノードを使用して、ビジネス シナリオにおけるさまざまな要件を満たすことができます。

表 1-2 WebLogic Integration Studio のノード

機能要件	通常モデルのノード	Studio でのシェイプ
手動もしくはソフトウェアで行う作業を指定。	タスク ノード	
プロセスの境界設定	開始ノード	
	完了ノード	
作業の実行順序を指定 : 順次処理または並列処理	接続	
	AND ノード	
	OR ノード	
アクションの繰り返し、または前の手順への戻りを指定（ループ化）	接続	

表 1-2 WebLogic Integration Studio のノード

機能要件	通常モデルのノード	Studio でのシェイプ
判定結果により実行するアクションを指定	分岐ノード	
外部イベントまたは特別なイベントの結果を基に実行するアクションを指定	イベント ノード	

通常 1 つのノードの完了が次のノードの起因となり、そのフローが特定の *アクション*でオーバーライドされない限り、連続的にアクティブとなります（詳細は「アクション」を参照）。

イベントは、ワークフロー中で待機状態で機能する特殊なタイプのノードです。イベントは前のノードの完了により *アクティブ*になりますが、イベントがリスンするようコンフィグレーションされた XML ドキュメントの到着後にイベントが非同期で *トリガ*されます。XML メッセージは、外部 JMS キューを通じて外部アプリケーションから受け取るか、内部 JMS キューを通じて別のワークフローから受け取ります。

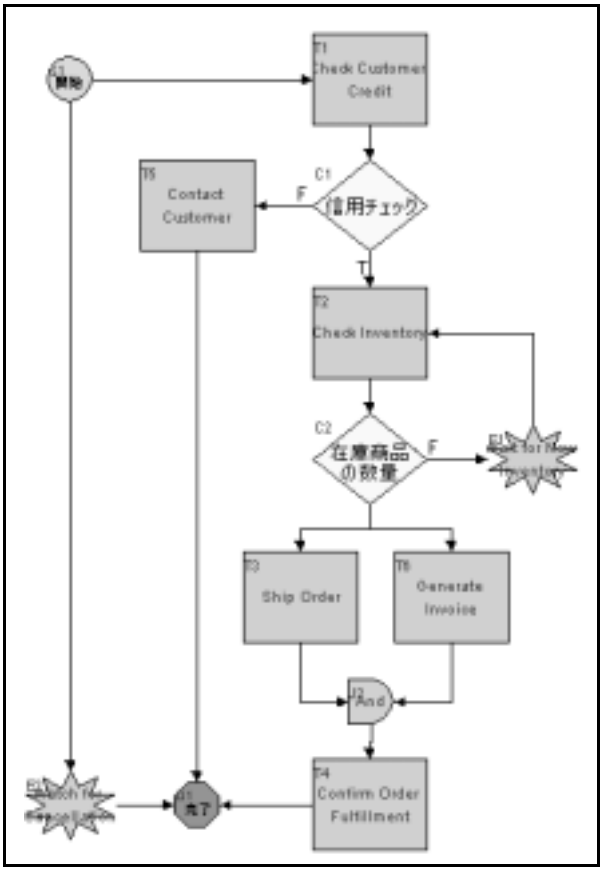
注意： イベント ノードは、JMS 上の XML 以外のメカニズムを使用できるプラグイン定義イベントからも、トリガできます。プラグインのプログラミングの詳細は、『*WebLogic Integration BPM プラグイン プログラミング ガイド*』を参照してください。

このサンプル ワークフローでは、起動、タスク、完了などの必須ノード以外にも以下のノードを使用します。

- 実行するアクションを指定する分岐ノード。信用チェックの結果と在庫状況を基に判定します。
- イベント ノード。新規入庫を待ち、顧客の発注キャンセルを監視します。
- AND ノード。受注完了通知を送信する前に、出荷と請求の両タスクが完了していることを確認します。

したがって、このシナリオを次の図のように表すことができます。

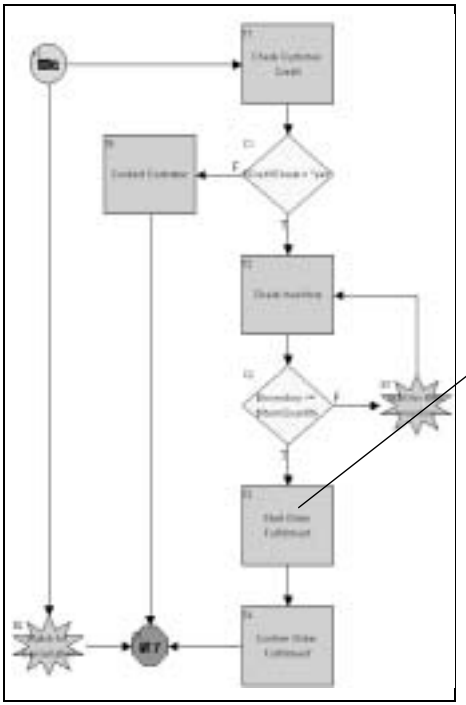
図 1-6　受注販売のシナリオ：Studio ワークフロー



上記のサンプル ワークフローにおいて、この単一のワークフローは、次の図のように、メイン ワークフローとサブ ワークフローの 2 つに分割されています。

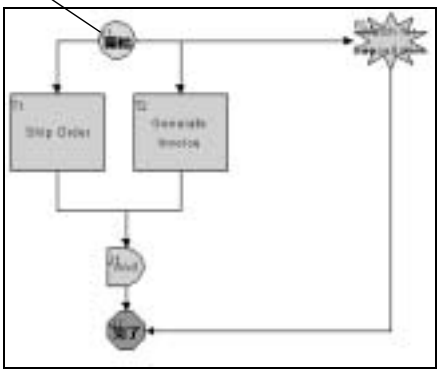
図 1-7 受注販売のシナリオ：メイン ワークフローとサブ ワークフロー

メイン ワークフロー : Order Processing



タスク ノードによってサブ ワークフローが呼び出される。

サブ ワークフロー : Order Fulfillment



図で示すように、サブ ワークフローを呼び出すために追加ノードがメイン ワークフローで使用されています。

第 5 章「ワークフローの作成」でノードを引き出し、それを第 4 章「ワークフローのノードの定義」で定義します。

アクション

アクションはワークフローの構築に使用する作業の基本単位です。Studio には 28 の種類のアクションがあり、WorkList ユーザに対する手動タスクの割り当て、プログラム全体の流れの制御、外部コンポーネントの統合および情報のやり取りなど、それぞれが異なる動作をします。

注意： デフォルトのアクション カタログを、プラグインを使用して拡張できます。プラグインのプログラミングの詳細は、『*WebLogic Integration BPM プラグイン プログラミング ガイド*』を参照してください。

アクションはワークフローで見える形では表現されず、ノード（通常はタスク ノード）で指定されます。ただし、起動、判定、イベント、例外ハンドラ（「例外ハンドラ」を参照）、さらに他のアクションにおいてもアクションを指定できます。

サンプル ワークフローでは次のようなアクションを使用することがあります。

- Worklist ユーザに対する承認タスクと出荷タスクの割り当て
- 在庫チェックと受注合計金額の計算を行う POBean EJB の呼び出し
- 顧客への E メール送信

このチュートリアルではさまざまなアクションを使用します。

変数

変数とは、ワークフローが値を格納できるコンテナであり、その値とは、テキスト、数値、日付、さらに Java オブジェクトや EJB への参照なども格納できます。ワークフローのテンプレートを定義する場合、変数は幾つでも作成できます。ワークフローの有効期間中は変数の値を変更できます。

一般的に、変数は実行時のワークフローによって収集される実際のデータ項目を表します。変数には、*ビジネス オペレーション*（「ワークフロー インタフェースの定義」を参照）から返される値、XML ドキュメントから抽出される値、ワークフローのアクションにより明示的に設定される値などがあります。また、変数はワークフローによって、分岐ノードにおける条件の評価や、Worklist ユーザの XML メッセージへの応答結果の格納など、さまざまな目的で使用できます。

サンプル ワークフローでは、以下のようなさまざまな機能の変数を使用します。

- 顧客名、ID、電子メール、住所、電話番号、国または州名、商品 ID、商品名、数量の表示
- 承認の要求に対する Worklist ユーザの応答結果の格納
- 在庫確認プロシージャの結果の格納

変数は第 5 章「ワークフローの作成」で作成します。

例外ハンドラ

WebLogic Integration では、すべてのワークフロー *インスタンス*または実行中のワークフローによって、デフォルトで使用されるシステム例外ハンドラが用意されています。実行中に例外が発生した場合、システム例外ハンドラはアクティブなトランザクションをロールバックし、再び例外を送出します。

ワークフロー用のカスタム例外ハンドラも設定できます。ロールバックやトランザクションのコミット時に機能する例外ハンドラに対して、特定のアクションを定義します。

第 6 章「カスタム例外ハンドラの使い方」において、`POBean` の `calculate()` メソッドに、無効なデータが渡される可能性に対して対処する例外ハンドラの定義と起動を行います。

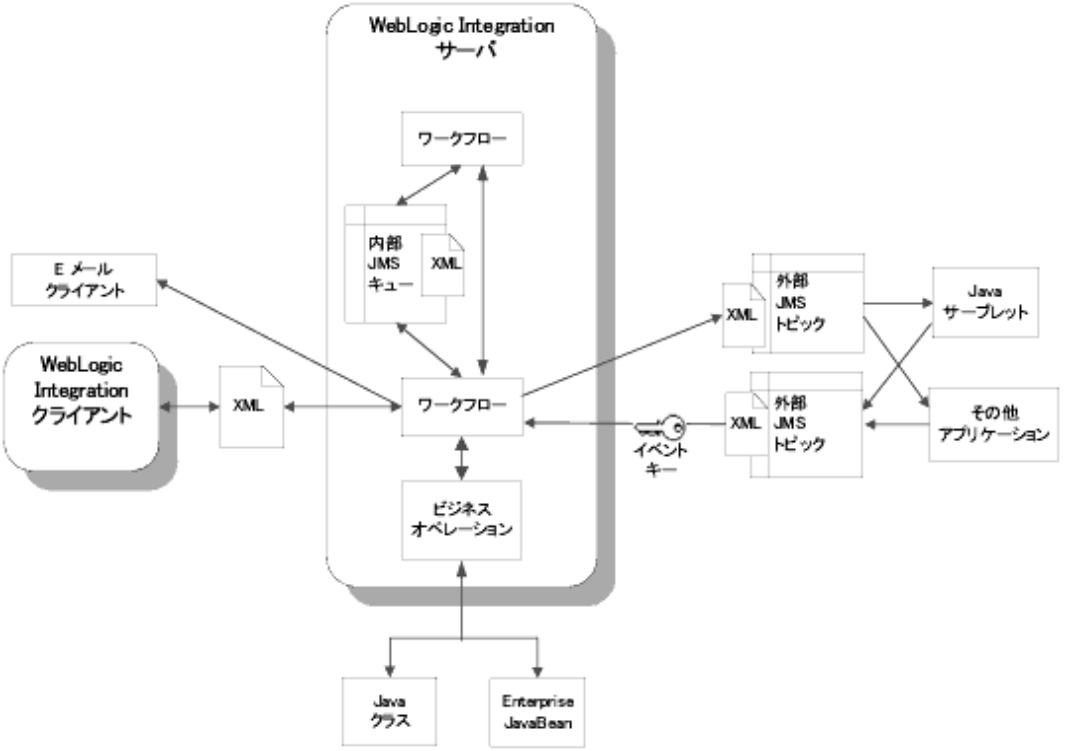
ワークフロー インタフェースの定義

このシナリオとワークフローの設計では、ワークフローは以下に述べる仮想のコンポ - ネントや実際のコンポーネント、またアプリケーションともインタフェースする必要があります。

- カスタマサービス、会計、出荷の各担当者が使用する Worklist クライアント
- 在庫チェック メソッドと金額計算メソッドを持つ EJB
- 顧客がやり取りに使用するフロントエンド Web アプリケーション内の、サーブレットなどのサーバ側コンポーネント
- バックエンド在庫レポート アプリケーション
- 確認メッセージの送信先の E メール クライアント

ワークフローが外部コンポーネントとインタフェースをとるためのさまざまな方式を、次の図に示しています。

図 1-8　外部コンポーネントと外部アプリケーションとのインタフェース



これらのインタフェースについて、以下の節で詳しく説明します。

ビジネス オペレーション

Java のクラスまたは Enterprise JavaBean のメソッドをワークフローから呼び出すには、*ビジネス オペレーション*を作成します。ビジネス オペレーションを実行すると、ワークフローの変数の値を入力パラメータとして EJB または Java オブジェクトのメソッドに渡すことができます。パラメータが渡されると、これらのメソッドは値をワークフローの変数に返します。

サンプル ワークフローでは、ビジネス オペレーションを以下の目的で使用します。

- 在庫チェック タスクの実行のための、`POBean` の `checkInventory()` メソッドの呼び出し
- 合計金額の計算タスク実行のための、`POBean` の `calculate()` メソッドの呼び出し
- 実行時の WebLogic Integration サーバ上に Bean のインスタンスを作成する `POBean` の `create()` メソッドの呼び出し

第 5 章「ワークフローの作成」で、ビジネス オペレーションを定義し、実行します。

XML ドキュメント

WebLogic Integration のワークフローは、サーブレットやバックエンド アプリケーションといった異なるタイプのサーバ側コンポーネントと非同期方式でやりとりを行うため、JMS トピックまたはキューにパブリッシュされる XML ドキュメント、または JMS キューから受け取った XML ドキュメントを通じてデータを共有します。

実際に、複数のワークフロー間で非同期で情報交換する場合にも、WebLogic Integration サーバは、内部トピックにポストされた XML ドキュメントを使用します。また、ワークフローと Worklist クライアント間の通信は、WebLogic Integration アプリケーション プログラミング インタフェース（API）を介して行われますが、多くの場合、リクエストとその応答は XML ドキュメントの形式をとります。

注意： WebLogic Integration API の詳細は、『*BPM クライアント アプリケーション プログラミング ガイド*』と『*BEA WebLogic Integration Javadoc*』を参照してください。

このように、これらのワークフローでは、多くの外部コンポーネントとの通信、また内部コンポーネントとの通信においても、XML メッセージを作成して、それを Worklist クライアントのユーザや、外部や内部のトピックまたはキューに送信する必要があります。XML 要素と属性は、実行時に値が使用されるワークフロー変数に対応します。サンプル ワークフローでは、送信する XML ドキュメントを次の目的に使用します。

- 受注処理アプリケーションをトリガするフロントエンド Web アプリケーションによって送信される XML ドキュメントの代用
- Worklist ユーザに対する信用チェック結果の要求
- Worklist ユーザに対する受注商品出荷先の国または州名の正確なデータの要求

XML ドキュメントは、その XML ドキュメントの参照に使用するアクションによっては、特定のワークフローに埋め込むことができます。一方、そのドキュメントを、XML ドキュメントや文書型定義（DTD）、そしてワークフロー定義外のその他の XML エンティティを格納する XML リポジトリへエクスポートしたり、インポートしたりすることもできるので、1 つのワークフロー内で作成したドキュメントにアクセスして別のワークフローに再利用することができます。XML ドキュメントの作成、リポジトリへのエクスポート、リポジトリからのインポートは 3-21 ページの「XML リポジトリの使い方 : Neworder XML ドキュメントのエクスポート」と 4-10 ページの「XPath 文を定義する」で行います。

イベント キー

ワークフロー イベントが受信した XML ドキュメントをリスンするよう定義する場合、イベントが想定するドキュメントのドキュメント タイプまたはルート要素を指定します。受信 XML ドキュメントをフィルタ処理し、特定の要素または属性値を持つ XML ドキュメントに限りイベントをトリガできるようにするには、イベント キーを指定します。イベント キーは、ドキュメント タイプ、および XPath 式により XML ドキュメントから抽出される要素の値を識別します。イベント キーは 1 つのテーブルに格納され、実行時に XML ドキュメントから返された値がサーバのイベント プロセッサによってイベント ノードで指定されたキーの値と比較されます。一致した場合にイベントがトリガされます。このメカニズムについては、4-51 ページの「イベントの作成 : Wait for New Inventory イベントの定義」で詳しく説明します。

ワークフローで使用するイベントの例には、以下のような目的で送信される XML メッセージを認識するためのイベント キーがあります。

- 顧客による発注のキャンセル通知
- 新規入庫の報告

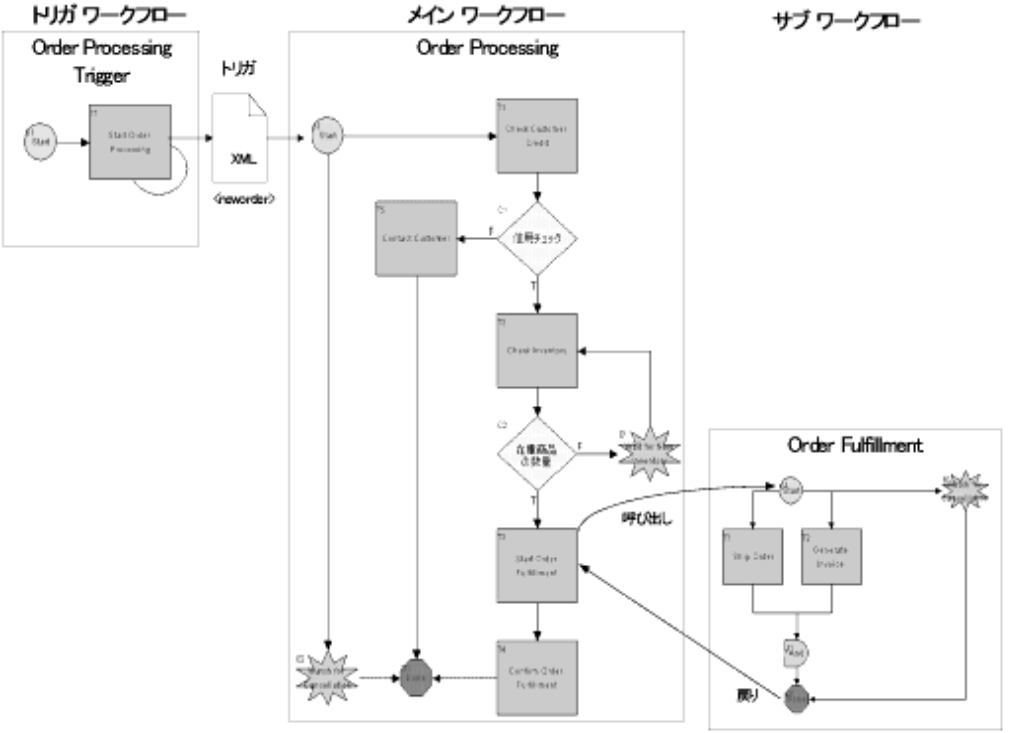
第 5 章「ワークフローの作成」で、イベント キーと XPath 式を定義します。

サンプル ワークフローを使った作業

Business Process Management に使用するサンプル ワークフローでは、シナリオは次の 3 つのワークフローとして実行されます。すなわち、メイン ワークフロー、サブ ワークフロー、*トリガ*ワークフローの 3 つです。トリガ ワークフローはチュートリアルとともに使用されます。同じシナリオを単純に 1 つのワークフローとして実行することもできますが、長く複雑なワークフローの構築には 2 つのワークフローにプロセスを分割する方式がよいでしょう。

3 つのサンプル ワークフローの関係を、次の図に表します。

図 1-9 3 つのサンプル ワークフローの関係



各ワークフローについてはチュートリアルの各節で詳しく説明しますが、各ワークフローの概要をこの節で説明します。

- メイン ワークフロー : Order Processing

  Order Processing がメイン ワークフローです。このワークフローは、最初の顧客リクエスト処理から、在庫チェックを行い、タスクの管理をサブ ワークフローに渡すまでのタスクを定義します。Order Processing が再びプログラムの制御を回復すると、確認通知を送信し、プロセス全体が完了します。

- サブ ワークフロー : Order Fulfillment

  Order Fulfillment は、在庫チェック後に Order Processing ワークフローから呼び出されるサブワークフローです。このサブワークフローは、出荷タスクと請求タスクを定義します。これらのタスクが完了すると、プログラムの制御はメイン ワークフローに戻されます。

- トリガ ワークフロー : Order Processing Trigger

  Order Processing Trigger は、顧客からの発注を受け付ける Web アプリケーションをシミュレートするために組み込まれています。その結果、Worklist からワークフローを実行できるようになります。

このチュートリアルで説明する 3 つのワークフローの完全な使用可能バージョンは、*SAMPLES_HOME*/integration/samples/bpm_tutorial フォルダの Tutorial.jar パッケージに用意されています。このパッケージには、テンプレート、テンプレート定義、ビジネス オペレーション、ワークフローの実行に必要なイベント キーが入っています。

このチュートリアルの手順に従うことをお勧めしますが、チュートリアルを使用せずにデモンストレーションとしてワークフローを実行したい場合は次のようにします。

1. 2-2 ページの「WebLogic Integration Studio の起動」の手順に従って Studio にログオンします。
2. Tutorial.jar ワークフロー パッケージ全体をインポートし、ワークフローをアクティブにします。2-7 ページの「ワークフロー オブジェクトのインポート : チュートリアル パッケージ ファイルのインポート」で説明する手順のステップ 5 のオプション 1 を選択します。
3. 第 7 章「サンプル ワークフローの実行とモニタ」の手順に従って Worklist にログオンし、ワークフローを実行します。

チュートリアルの使い方

このチュートリアルは、Studio と Worklist アプリケーションの、多くの一般機能の使い方を学習できるように編成されています。最初の 4 つの各節では、Studio とワークフローの設計の異なる側面が 1 つのサンプル ワークフローを使って強調され、最後の節ではアクションのワークフロー全体を示します。

第 2 章「WebLogic Integration Studio 入門」では、Studio にログオンし、Order Processing Trigger と Order Fulfillment のワークフローをインポートして Studio インタフェースのさまざまな側面を調べます。この節で説明するトピックは次のとおりです。

- Studio の起動とログオン
- ワークフロー オブジェクトの階層構造
- ワークフロー パッケージのインポート
- フォルダ ツリーの使い方
- Interface View の使い方

第 3 章「ワークフロー オブジェクトとプロパティ」で、Order Processing Trigger ワークフローを使用してワークフローのプロパティを調べ、ワークフローに埋め込まれている XML メッセージを編集します。この節で説明するトピックは以下のとおりです。

- ワークフロー オブジェクトのプロパティ
- ワークフローの式
- ポストする XML メッセージの表示と編集
- XML リポジトリへのドキュメントのエクスポート
- ワークフローの保存

第 5 章「ワークフローの作成」で、Order Processing ワークフローを次の手順で新しく作成します。

- ワークフロー テンプレートの作成
- テンプレート定義の作成

- 変数の作成
- ワークフロー ラベルの追加
- テンプレート定義のアクティブ化

第 4 章「ワークフローのノードの定義」で、変数、アクション、その他のワークフロー オブジェクトの追加と定義を行い、Order Processing ワークフローのノードを定義します。この節で説明するトピックは以下のとおりです。

- イベントによってトリガされる開始ノードの作成
- ワークフローの式の作成
- Worklist ユーザとのやり取り
- 分岐ノードによる条件の評価
- ビジネス オペレーションの作成と実行
- イベントとイベント キーの作成
- サブ ワークフローの呼び出し
- クライアントへの E メール送信

第 6 章「カスタム例外ハンドラの使い方」では、Order Fulfillment ワークフローを使用して、WebLogic Integration 例外ハンドラの機能と、他の Worklist ユーザとのやり取りについて説明します。

最後に、第 7 章「サンプル ワークフローの実行とモニタ」で、Worklist アプリケーションにログオンし、サンプル ワークフローを実行しながら、同時に Studio で Worklist アプリケーションをモニタします。

このチュートリアルで説明されない Studio の機能には、以下のようなものがあります。

- タイマートリガ機能
- タイマーイベント機能
- カレンダー機能
- ユーザとロールのパーミッション設定機能
- ビジネス オペレーションを使用する Entity Beans と Java オブジェクトの呼び出し機能

- 実行プログラムの呼び出し機能
- XML イベントのポスト用の JMS メッセージ機能
- 監査機能
- 作業負荷と統計報告によるワークフロー モニタ機能
- プラグイン機能

これらの機能の使い方については、『*WebLogic Integration Studio ユーザーズ ガイド*』を参照してください。

2 WebLogic Integration Studio 入門

この節では、Studio クライアント アプリケーションについて理解を深めます。このアプリケーションを使用してワークフローの設計を行うために必要な、以下の作業について説明します。

- Studio の起動とログオン
- ワークフロー オブジェクトの階層構造
- ワークフローのインポート
- テンプレート定義の表示
- インタフェース ビュー の使い方
- フォルダ ツリーの使い方
- オブジェクト プロパティの表示

WebLogic Integration Studio の起動

WebLogic Integration Studio を起動する手順は、次のとおりです。

1. 以下のいずれか 1 つを実行します。
   - Windows システムで、[スタート | プログラム | BEA WebLogic E-Business Platform | WebLogic Integration 7.0 | Studio] を選択します。
   - UNIX システムで、`WLI_HOME/bin` ディレクトリに移動し、コマンド プロンプトに次のように入力して Studio 起動スクリプトを実行します。

     ```
     sh studio.sh
     ```

   [WebLogic Integration へのログオン] ダイアログ ボックスが、[Business Process Management Studio] アプリケーション ウィンドウの前面に表示されます。

図 2-1 [WebLogic Integration へのログオン] ダイアログ ボックス

2. ユーザ名とパスワードを該当するフィールドに入力します。認証済みのユーザとして設定されている場合は自分のユーザ名とパスワードを入力できます。認証されていない場合は、サーバのインストール時に CDExpress に対してデフォルト値として与えられたユーザ名を使用します。使用できるユーザ名を次の表に示します。

表 2-1 CDExpress のユーザとパスワード

ユーザ名	Password
joe	password
mary	password
admin	security

注意： ユーザ名とパスワードは大文字と小文字の区別が必要です。表中のユーザ名とパスワードは必ず小文字で入力してください。

3. [サーバ URL] フィールドで、WebLogic Integration サーバを稼働させるシステムを次のように指定します。

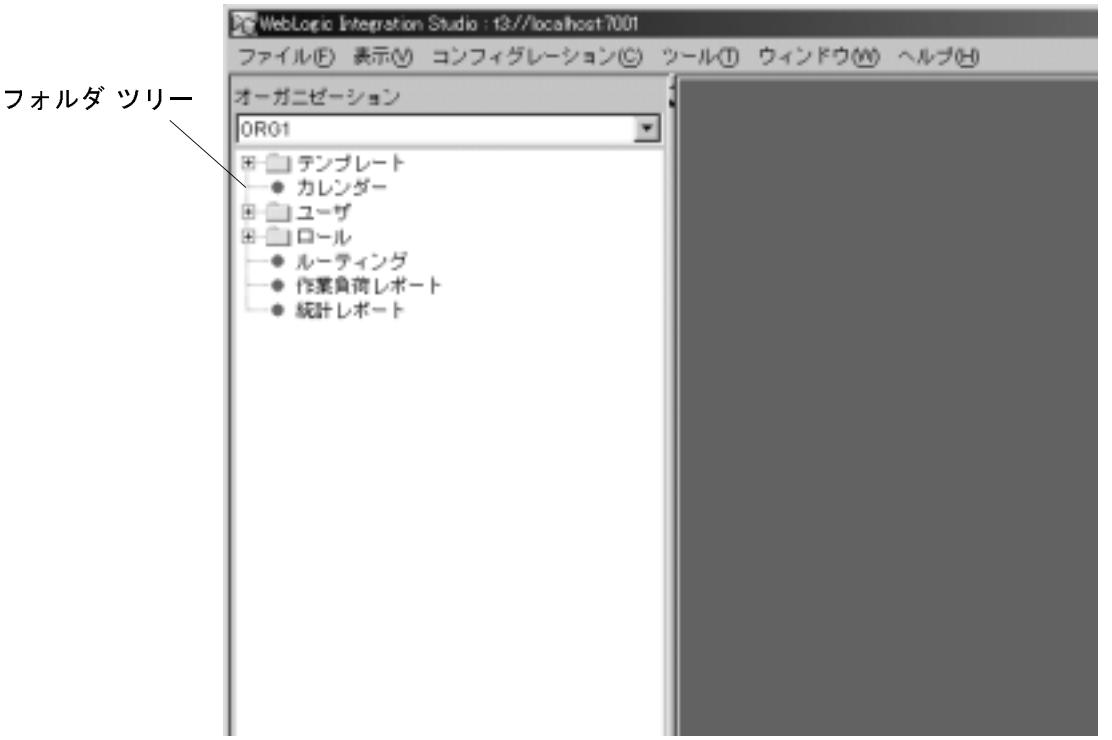
   `t3://host:7001`

   この *host* とは、コンピュータ名または WebLogic Integration サーバ実行中のシステムの IP アドレスです。サーバを Studio アプリケーションと同じコンピュータ上で実行している場合は、`localhost` を指定します。

4. [OK] をクリックします。Studio のメイン ウィンドウが表示されます。このウィンドウは 2 つのペインで縦に分割されています。右ペインは空白のワークスペースです。左ペインにはオーガニゼーション名の下にフォルダ ツリーが表示され、そのオーガニゼーションにユーザ名が関連付けられています。

5. [オーガニゼーション] ドロップダウン リストから、[CDExpress] を選択します。

図 2-2 [WebLogic Integration Studio] アプリケーション ウィンドウ



ワークフロー オブジェクトの関係

Studio では、データとプロセス オブジェクトは別の階層レベルに表示されます。これらはユーザが作成する以下のオブジェクトのコンテナとして考えることができます。

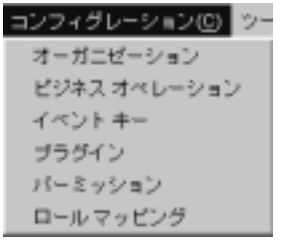
- グローバル オブジェクト
- テンプレートとテンプレート定義
- オーガニゼーション

この節では、この 3 つのレベルについて説明します。

グローバル オブジェクト

グローバル オブジェクトは階層構造の最上位のレベルにあり、すべてのワークフローとオーガニゼーションが利用できます。コンフィグレーション オブジェクトをフォルダ ツリーに表示したり、[コンフィグレーション] メニューからアクセスしたりできます。オブジェクトには、カレンダー、ユーザ、ビジネス オペレーションやイベント キーなどのようなワークフロー *テンプレート*（下記参照）インタフェース、およびユーザのパーミッションが含まれます。

図 2-3 [コンフィグレーション] メニュー



テンプレートとテンプレート定義

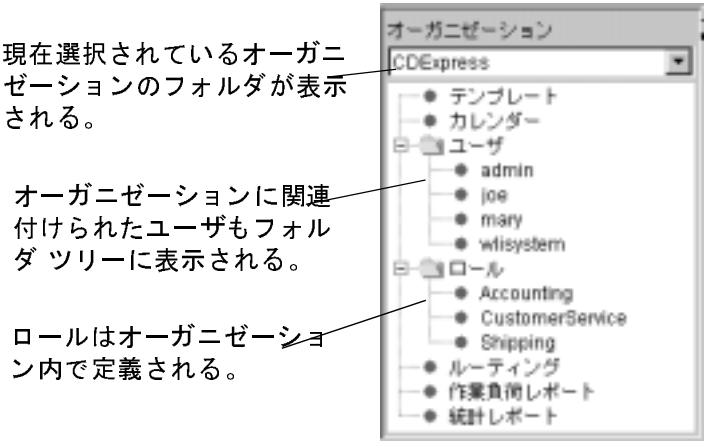
テンプレートは複数のワークフロー定義を格納できるフォルダです。1 つのテンプレートを複数のオーガニゼーションで共有できます。テンプレートには自由に名前を付けられますが、その名前は一意のものである必要があります。

テンプレート フォルダ内に、ノード、アクション、変数や例外ハンドラなどのオブジェクトを含む実際のワークフローであるテンプレート定義を作成します。テンプレート定義には、有効開始日時と終了日時に基づいて自動的にラベルが割り当てられ、また複合的なテンプレート定義はさまざまなバージョンのワークフローに対応します。

オーガニゼーション

オーガニゼーションは、別のレベルの構造を表します。オーガニゼーションは、ユーザ、ロール、他のワークフローなど、そのオーガニゼーション内で作成されたすべてのワークフローがアクセスできるオブジェクトがあります。Studio のインタフェース画面では、左ペインにあるフォルダ ツリーは、現在選択されているオーガニゼーションのオブジェクトを示しています。

図 2-4 [オーガニゼーション] フォルダ ツリー



ユーザとテンプレートを複数のオーガニゼーションに関連付けることができますが、カレントのオーガニゼーションに関連付けられているユーザやテンプレートは [ユーザ] と [テンプレート] フォルダにそれぞれ一覧で表示されます。

一方、ロール、ルーティング、レポートなどの他のオブジェクトは、実際にはオーガニゼーション内で定義されます。

カレンダーとルーティングの詳細は、『*WebLogic Integration Studio ユーザーズ ガイド*』の「データの管理」を参照してください。レポートの詳細は、『*WebLogic Integration Studio ユーザーズ ガイド*』の「ワークフローのモニタリング」を参照してください。

これで、テンプレートとテンプレート定義をインポートし、ほかのワークフローのオブジェクトを表示できるようになります。

ワークフロー オブジェクトのインポート：チュートリアル パッケージ ファイルのインポート

ワークフロー オブジェクトをバイナリ形式で格納している Java アーカイブ (JAR) パッケージ ファイルに対し、すべての依存関係も含めたワークフロー パッケージ全体をインポート、またはエクスポートできます。Import/Export Package 機能を使用すると、以下のワークフロー オブジェクトをインポートできます。

- テンプレート
- テンプレート定義
- イベント キー
- ビジネス オペレーション
- XML リポジトリのコンテンツ
- カレンダー

注意： テンプレート定義も同様に、XML ファイルに対してインポートまたはエクスポートできます。

インポート / エクスポートの詳細は、『*WebLogic Integration Studio ユーザーズ ガイド*』の「ワークフロー パッケージのインポートとエクスポート」を参照してください。

Tutorial.jar ワークフロー オブジェクトのインポート

`tutorial.jar` パッケージには以下のオブジェクトが含まれています。

- 3 タイプのワークフロー テンプレート : Order Processing Trigger、Order Processing、Order Fulfillment

- 3 つのテンプレート定義 : リスト内の各テンプレートに対して 1 定義
- 2 つのイベント キー : `cancelledorder` と `newinventory`。前者は、Order Processing ワークフローと Order Fulfillment ワークフロー内に定義されたイベントによって使用されます。後者は、Order Processing ワークフロー内のイベントによってのみ使用されます。
- 3 つのビジネス オペレーション : Check Inventory、Calculate Total Price、Create OrderBean。1 つ目は Order Processing ワークフロー内で定義したタスクが参照し、2 つ目は Order Fulfillment ワークフロー内で定義したタスクが使用し、3 つ目は両方のワークフローが使用します。

ワークフローをインポートするための方法には、次の 2 通りがあります。1 つはチュートリアルで推奨されているとおり、最初から Order Processing ワークフローを作成する方法、もう 1 つは設計手順を無視して単にワークフローを実行する方法です。

サンプル ワークフローをインポートする手順は次のとおりです。

1. [ツール（T）｜パッケージをインポート] を選択します。[インポート] ウィザードの [ファイルを選択] ダイアログ ボックスが表示されます。

図 2-5　[ファイルを選択] ダイアログ ボックス

2. [参照] ボタンをクリックし、*SAMPLES_HOME*/integration/samples/bpm_tutorial ディレクトリに移動し、Tutorial.jar ファイルを開きます。
3. [次へ] をクリックします。[インポートするコンポーネントを選択] ダイアログ ボックスが表示されます。このダイアログ ボックスには CDExpress に設定済みの対象オーガニゼーションと、デフォルトで選択されたインポート ファイル内のすべてのワークフロー オブジェクトが表示されています。

図 2-6 [インポートするコンポーネントを選択] ダイアログ ボックス

4. [インポート後にワークフローをアクティブ化] チェック ボックスを選択します。

   注意 : ワークフローを実行する前にアクティブ化する必要があります。

5. 以下のオプションのどちらかを選択します。

   オプション 1 : チュートリアルに進まない場合は、手順 9 に進み、テンプレート、テンプレート定義、ビジネス オペレーション、イベント キーから構成されるパッケージ全体をインポートします。

   オプション 2 : チュートリアルに進む場合は、手順 6 に進み、以下のオブジェクトのみをインポートします。

   - Order Processing Trigger のテンプレートとテンプレート定義
   - Order Fulfillment のテンプレートとテンプレート定義

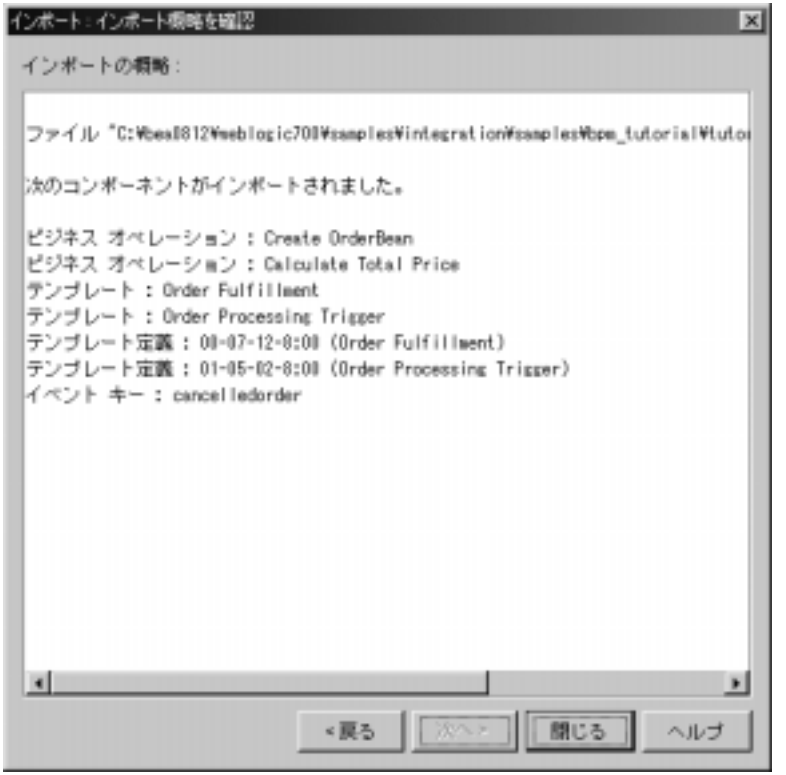
- Calculate Total Price ビジネス オペレーション
- Create OrderBean ビジネス オペレーション
- cancelledorder イベント キー

6. [テンプレート] チェック ボックスを展開し、[Order Processing] チェック ボックスの選択を解除します。
7. [ビジネス オペレーション] チェック ボックスを展開し、[Check Inventory] チェック ボックスの選択を解除します。
8. [イベント キー] チェック ボックスを展開し、[newinventory] イベント キーの選択を解除します。

   ダイアログ ボックスは次のように表示されます。

9. [インポート] ボタンをクリックします。インポートされたオブジェクトの概要を示す [インポート概略を確認] ダイアログ ボックスが表示されます。オプション 2 に従った場合は、このダイアログ ボックスは次のように表示されます。

図 2-7 [インポート概略を確認] ダイアログ ボックス



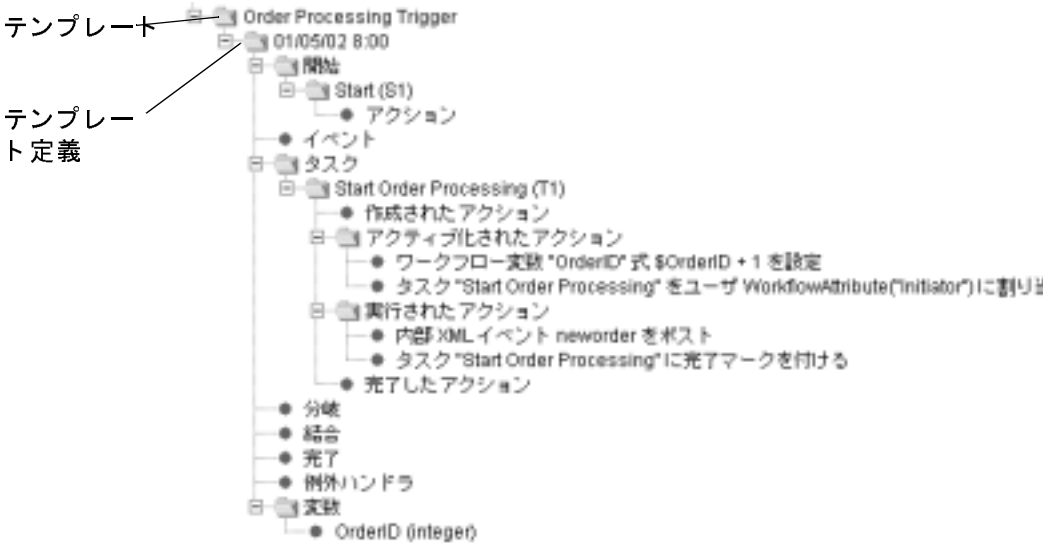
10. [閉じる] ボタンをクリックします。

これで、インポートされたテンプレートとテンプレート定義がフォルダ ツリーに表示されます。

フォルダ ツリーの使い方 : Order Processing Trigger のテンプレート内容の表示

テンプレート定義のフォルダ ツリーは、実際に開かなくても表示できます。該当するテンプレート定義フォルダのすべてのフォルダを展開するだけです。Order Processing Trigger テンプレート定義のフォルダ ツリー構造を、次の図に示します。

図 2-8　Order Processing Trigger ワークフロー : フォルダ ツリー



フォルダ ツリーでオブジェクトを右クリックし、表示されるポップアップ メニューから [プロパティ] を選択すると、オブジェクトのプロパティを表示できます。テンプレート定義がクローズ状態の場合は、[プロパティ] ダイアログ ボックスは読み取り専用です。編集を行うには、最初にテンプレート定義を開く必要があります。

注意： 変数と例外ハンドラのプロパティについては、フォルダ ツリーからのみアクセスできます。

ここで、インポートしたワークフローを、[ワークフロー設計] ウィンドウ内の [インタフェース ビュー] に表示します。フォルダ ツリーの最後の 2 つのビューを確認するには、テンプレート定義を開く必要があります。

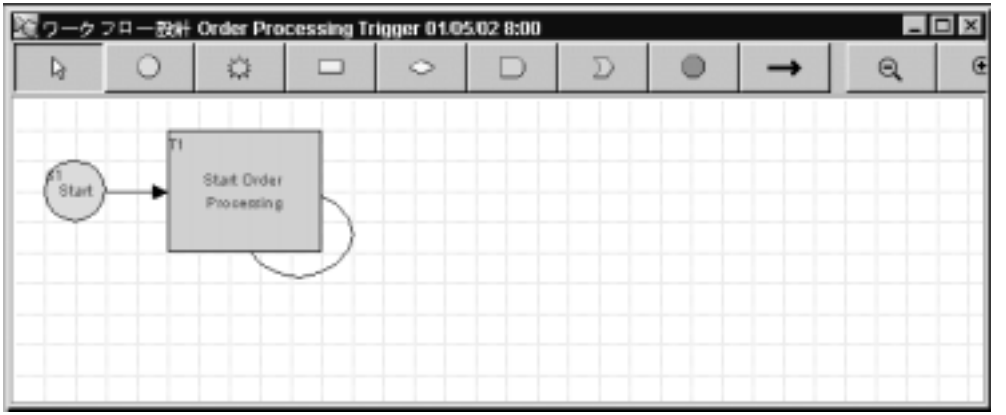
ワークフローの表示：Order Processing Trigger のテンプレート定義の表示

ワークフローのグラフィック表示を確認するには、ワークフローのテンプレート定義を開く必要があります。

Order Processing Trigger のテンプレート定義を開く手順は次のとおりです。

1. フォルダ ツリーで、Order Processing Trigger のテンプレートを展開します。
2. テンプレート定義を右クリックし、表示されるポップアップ メニューから [開く] を選択します。[ワークフロー設計] ウィンドウの右ペインに、ワークフローが表示されます。

図 2-9 Order Processing Trigger テンプレート定義 : [ワークフロー設計] ウィンドウ



該当するオブジェクトをダブルクリックすると、[ワークフロー設計] ウィンドウ内の任意のオブジェクトのプロパティを表示して編集できます。

インタフェース ビューの使い方

インタフェース ビューで、各ノードに関連する受信 / 発信 XML ドキュメント、ビジネス オペレーション、サブワークフロー、プラグインをグラフィック表示し、ワークフローについての付加情報を得ることができます。[インタフェース ビュー] アイコンと、そのアイコンが表すオブジェクトを次の表に示します。

表 2-2 [インタフェース ビュー] アイコン

アイコン	オブジェクト
	ビジネス オペレーション

表 2-2 [インタフェース ビュー] アイコン

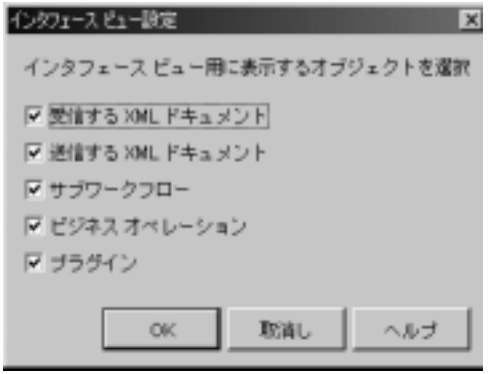
アイコン	オブジェクト
	プラグイン
	サブワークフロー
	着信 XML ドキュメント
	発信 XML ドキュメント

ワークフローの インタフェース ビューを使用するには、まずプリファレンスを設定し、次にツールバーの切り替えボタンを使用してインタフェース ビューのオン / オフを切り替えます。

インタフェース ビュー設定は次の手順で実行します。

1. [表示（V） | インタフェース ビュー] を選択します。[インタフェース ビュー設定] ダイアログ ボックスが表示されます。

図 2-10 [インタフェース ビュー設定] ダイアログ ボックス

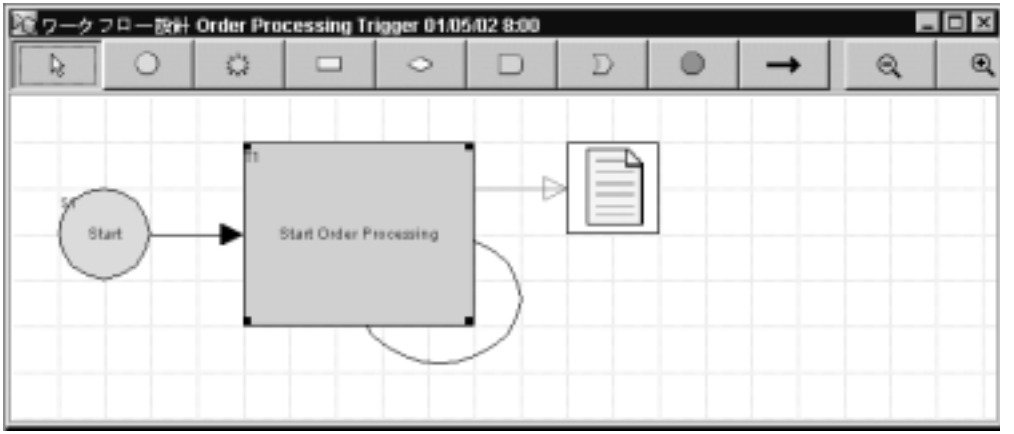


2. すべてのチェック ボックスをチェックし、[OK] をクリックします。
3. [ワークフロー設計] ウィンドウで、ツールバーの一番右端にスクロールし、[インタフェースの表示 / 非表示] ボタン  をクリックします。

   参考： ツールバーの非アクティブな領域をクリックし、デスクトップ上にドラッグすることで、ツールバーを任意の位置に変えることができます。また、[表示 | ルック & フィール | Metal] を選択して、作図領域内に全体が含まれるツールバーを表示できます。

   これで、ワークフローは次のように表示されます。

図 2-11 Order Processing Trigger ワークフロー：インタフェース ビュー



インタフェース ビュー内で、ビジネス オペレーションや発信 XML ドキュメントなどの、インタフェース オブジェクトの読み込み専用 [プロパティ] ダイアログ ボックスにもアクセスすることができます。インタフェース オブジェクトの [プロパティ] ダイアログ ボックスを表示するには、[ワークフロー設計] ウィンドウ内のインタフェース アイコンをダブルクリックします。

注意： 着信 XML ドキュメントはワークフロー オブジェクトではないので、そのインタフェース アイコンをダブルクリックしても無効です。着信 XML ドキュメントのプロパティに関する基本情報を得るには、該当するイベントまたは開始ノードをポイントし、その上でダブルクリックします。

チュートリアルの次の節では、その他のタイプのワークフロー オブジェクトとそのプロパティについて説明します。

3 ワークフロー オブジェクトとプロパティ

注意： Worklist クライアント アプリケーションは WebLogic Integration リリース 7.0 より非推奨となりました。代替機能に関する詳細については、『*WebLogic Integration リリース ノート*』を参照してください。

この節では、Order Processing Trigger ワークフローを説明しながらチュートリアルを進めます。ワークフローの概要を説明した後にワークフローのオブジェクトを詳しく調べて、以下の項目について理解を深めます。

- テンプレート定義プロパティ
- 変数のプロパティ
- 開始ノードのプロパティ
- タスク ノードのプロパティ
- ワークフローの式

最後に、Studio のいくつかの機能を使って慣れるために、以下を実行します。

- ポストする XML ドキュメントの編集
- リポジトリへの XML ドキュメントのエクスポート
- テンプレート定義の保存

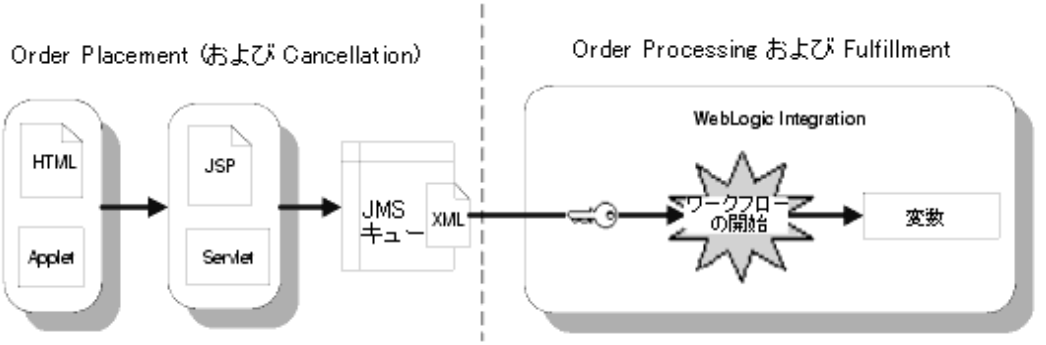
チュートリアルのこの節では、以下の手順が完了しているものとします。

- 2-2 ページの「WebLogic Integration Studio の起動」の説明に従って、Studio にログオンしていること。
- 2-7 ページの「ワークフロー オブジェクトのインポート : チュートリアル パッケージ ファイルのインポート」と 2-14 ページの「ワークフローの表示 : Order Processing Trigger のテンプレート定義の表示」の説明に従って、Order Processing Trigger のテンプレート定義をインポートして開いていること。

Order Processing Trigger ワークフロー

Order Processing Trigger ワークフローは、Order Processing ワークフローを通常トリガするプロセスをシミュレートします。実際のアプリケーションでは、多くの場合、これは XML ドキュメントを JMS キューに送信するサーブレットによって実行されます。次の図に示すように、XML メッセージがキューに着信することにより Order Processing ワークフローの開始がトリガされます。

図 3-1 Order Processing 開始のシミュレーション



このチュートリアルでは、Worklist ユーザに XML ドキュメントをポストするプロセスを開始させて、このプロセスをシミュレートします。Order Processing Trigger ワークフローの詳細を次の図に示します。図で番号が振られている手順については、図の下の表で説明します。この図は、実際のプロセスと実装を、サンプル ワークフロー内の実際の実装にマップしています。

図 3-2　Order Processing Trigger ワークフロー：詳細表示

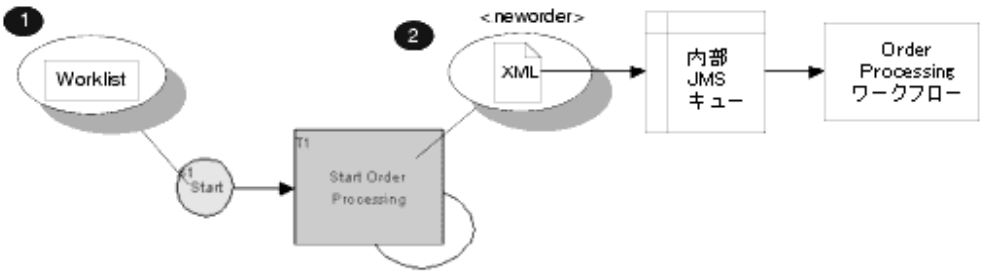


表 3-1 Order Processing Trigger ワークフローの概要

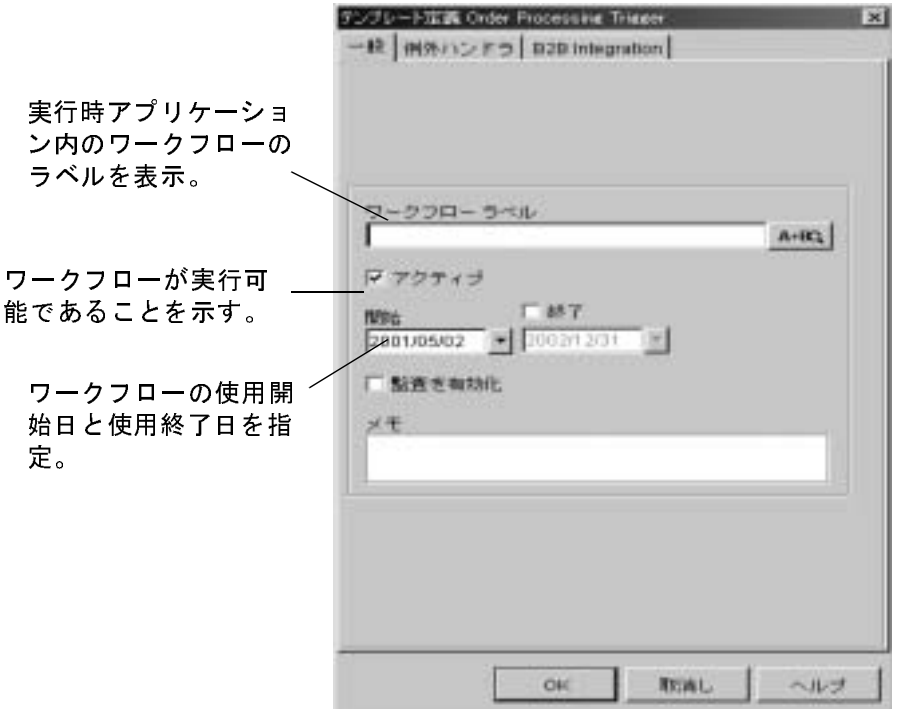
プロセス	実装
顧客が CDExpress という音楽メディア小売店にブラウザ ベースの発注フォームなどのオンライン メカニズムを使用して商品を発注します。その顧客は、氏名、連絡用 E メール アドレス、出荷先の国または州名、希望商品（ID および商品）、発注数量を指定します。 実際のアプリケーションでは、ワークフローに渡される外部 JMS キューに対して XML ドキュメントを転送するサーブレットにより受注が処理される。	1. 開始ノードが手動に設定されているので、ワークフローは Worklist ユーザによって開始される。 2. 最初のタスクは、ワークフローを開始する Worklist ユーザに割り当てられる。タスクが実行されると、<neworder> XML ドキュメントが内部 JMS キューにポストされる。XML ドキュメントには、受注のステータス、受注 ID、顧客 ID、顧客名、E メール、住所、電話番号、国名、商品名、数量が含まれる。

注意： このワークフローには完了ノードがありません。つまり、このワークフローは無限に繰り返されることを意味します。後ほどチュートリアルで説明するように、ワークフローを実行する場合、このメカニズムを使用すると、Worklist アプリケーションのタスク一覧ウィンドウ内からワークフローを開始できるようになります。

テンプレート定義のプロパティ

Order Processing Trigger テンプレート定義を開いた状態でテンプレート定義のプロパティを表示するには、フォルダ ツリーか、[ワークフロー設計] ウィンドウ内の任意の場所を右クリックし、表示されるポップアップ メニューから [プロパティ] を選択します。[テンプレート定義 Order Processing Trigger] ダイアログ ボックスが表示されます。

図 3-3 [テンプレート定義 Order Processing Trigger] ダイアログ ボックス



複数のテンプレート定義で同じ有効開始日付と終了日付を指定できますが、アクティブな定義にはそれぞれユニークな日付または時刻を指定する必要があります。実行時において、あるワークフローに複数のテンプレート定義が存在する場合、処理エンジンは、その有効開始日時 / 終了日時を基に*現在*に最も近いアクティブなテンプレート定義を選択します。

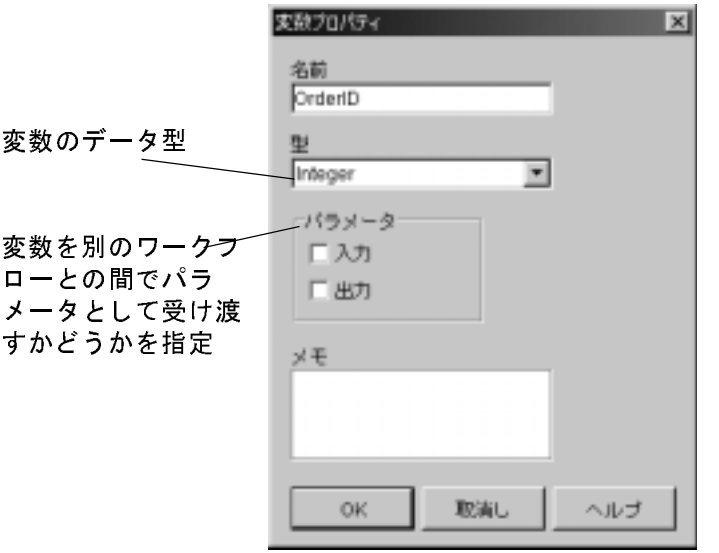
この Order Processing Trigger テンプレート定義をインポート プロセス中にアクティブ化したため、ダイアログ ボックスでは、すでにアクティブ化された状態で表示されます。この図のテンプレート定義のワークフローは、2001 年 5 月 2 日から有効であり、使用期限は指定されていません。

このワークフローにはラベルを使用していません。

変数プロパティ : OrderID 変数の表示

変数プロパティを表示するには、フォルダ ツリーに移動し、[変数] フォルダの下の OrderID 変数をダブルクリックします。[変数プロパティ] ダイアログ ボックスが表示されます。

図 3-4　OrderID 変数 : [変数プロパティ] ダイアログ ボックス



変数は、ほかのテンプレート定義、XML ドキュメント、EJB および Java クラス メソッドにパラメータとして渡すことができます。入力パラメータは、変数がその値を呼出し側、つまり親ワークフローから受け取ることを示します。出力パラメータは、呼び出し元のワークフローに返された値を変数が持つことを示します。

Order Processing Trigger テンプレートでは、OrderID 変数には入力パラメータまたは出力パラメータのマークが付いていません。図 3-4 を参照してください。後で説明しますが、Order Processing と Order Fulfillment ワークフロー テンプレートでは、変数にはほかのワークフローから渡されたことを示す *input* パラメータのマークが付いています。OrderID 変数は、ワークフローの各インスタンスで受注番号のトラッキングに使用されるため、Order Processing ワークフローと Order Fulfillment ワークフローの両方に渡されます。

変数はスコープ内でグローバルです。つまりワークフロー全体に適用されます。現在のワークフローで変数が使用されているノードやアクションの概要を表示することができます。

OrderID 変数の使い方を表示するには、フォルダ ツリーに移動し、[変数] フォルダの下の OrderID 変数を右クリックして表示されるポップアップ メニューから [使用] を選択します。[変数の使用場所] ウィンドウが表示されます。

図 3-5　OrderID 変数 : [変数の使用場所] ウィンドウ

上の図に示されているように、OrderID 変数は、Start Order Processing タスクのワークフロー変数を設定 というアクションで使用されています。このアクションについては、「ワークフローの式 : ワークフロー変数を設定 と ユーザにタスクを割り当て アクションの表示」で詳しく説明します。

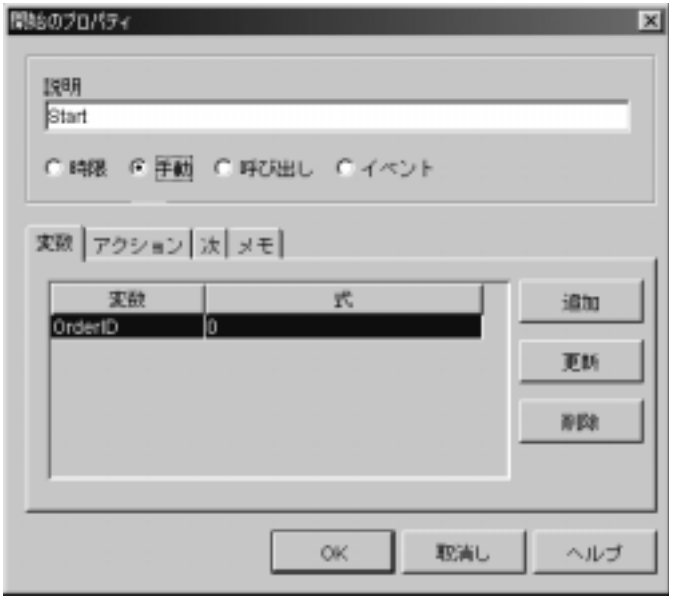
変数は、5-11 ページの「ワークフロー ラベルの追加」と第 4 章「ワークフローのノードの定義」全体を通して作成します。

開始ノードのプロパティ

Order Processing Trigger には開始ノードが 1 つしかないため、開始ノードにはノード名がありません。ただし、複数の開始ノードを含むワークフローを定義し、並列実行する複数の処理パスを規定することも可能です。また、開始ノードを使用して、任意のワークフロー変数を初期化することもできます。

開始ノードのプロパティを表示するには、[ワークフロー設計] ウィンドウに移動して [開始] ノードをダブルクリックします。[開始のプロパティ] ダイアログ ボックスが表示されます。

図 3-6 [開始のプロパティ] ダイアログ ボックス



ダイアログ ボックスに表示されるように、ワークフローの開始方法には、以下の 4 種類があります。

- 時限 ー ワークフローは、指定日の指定時刻に自動的に開始します。このチュートリアルではタイマーでの開始は使用しません。
- 手動 ー ワークフローは Worklist ユーザが開始します。Order Processing Trigger ワークフローを手動による開始に定義し、チュートリアル サンプル全体を起動できるようにします。
- 呼び出し ー ワークフローが別のワークフローに呼び出されます。サンプルの Order Fulfillment ワークフローは、呼び出しによって開始します。
- イベント ー ワークフローは XML ドキュメントの JMS キューの着信によってトリガされます。Order Processing ワークフローのイベントトリガ開始ノードを第 4 章「ワークフローのノードの定義」で作成し、その節の後半

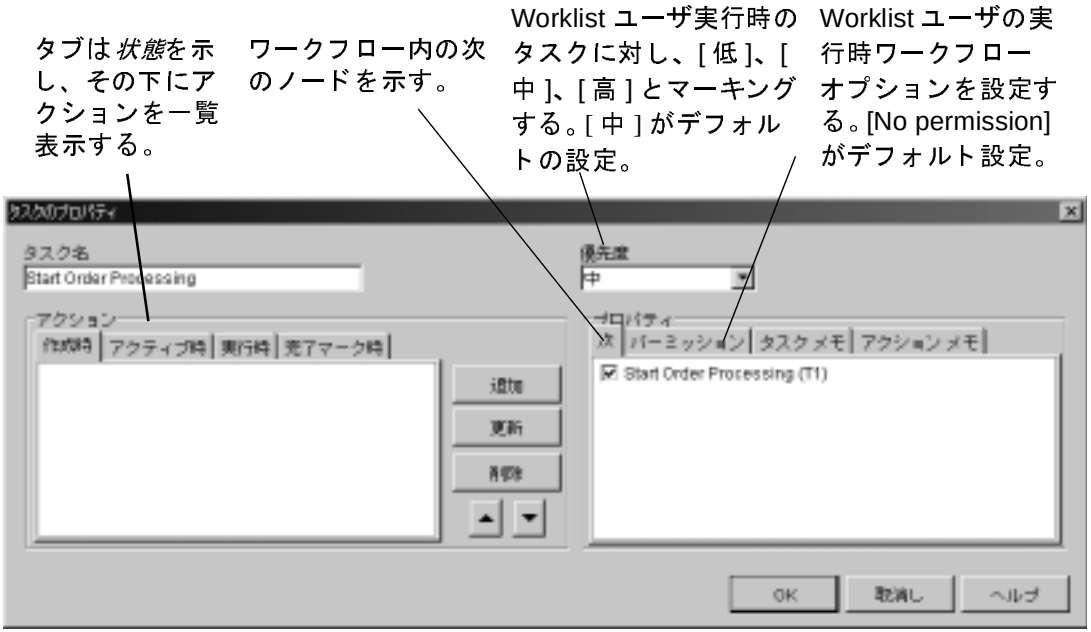
で、このワークフローのトリガとして動作する XML ドキュメントを使用します。

更に、この開始ノード内では、ワークフロー実行のたびに加算される OrderID 変数が初期化され、0 (ゼロ) になります。この変数の詳細は、3-5 ページの「変数プロパティ : OrderID 変数の表示」を参照してください。

タスク ノードのプロパティ : Start Order Processing タスク ノードの表示

タスク ノードのプロパティを表示するには、[ワークフロー設計] ウィンドウに移動して Start Order Processing タスク ノードをダブルクリックします。[タスクのプロパティ] ダイアログ ボックスが表示されます。

図 3-7　Start Order Processing タスク : [タスクのプロパティ] ダイアログ ボックス



[次] タブの下で、このタスクのサクセサがそのタスク自身であり（[ワークフロー設計] ウィンドウ内でループ矢印によって指示）、リスト内に [完了] ノードはありません。つまり、ワークフローは実際には終了することがなく、Start Order Processing タスクが実行されるたびに、このタスク自身の別のインスタンスを作成します。第 7 章「サンプル ワークフローの実行とモニタ」では、ワークフローを実行する時にこのプロセスが動作する様子を示します。

タスクの状態とアクション

ダイアログ ボックスの左側のタブは、実行時の各*状態* を示します。所定のタスクに対して状態を指定するタブに割り当てられたアクションは、指定の状態になったときに実行されます。以下の 4 種類の状態があります。

- 作成時 ー このタブ内に含まれるアクションは、ワークフローの開始直後に実行されます。

- アクティブ時 ー フローがこの特定のタスクに到達した場合に、ここにリストされたアクションが実行されます。一般的には、自動起動するアクションのすべてをこのタブの下に指定します。
- 実行時 ー ここに含まれるアクションは、あるアクションが Worklist 内で手動で実行された場合、または実行タスクの API でプログラムにより実行された場合、あるいは Execute Task アクションを使用して実行された場合、のいずれかの方法で実行された場合のみ実行されます。これ以外の場合は、[実行時] タブに指定されたアクションは実行されません。
- 完了マーク ー タスクを、タスクに完了マークを付ける アクションまたは対応する API が、完了とマークします。通常、このタブは、タスクに完了マークを付ける アクションがそのタスク内に指定されてはいませんが、フロー内のほかのノードによって指定されている場合に使用されます。一般に、監査情報をログに書き込むなどのクリーンアップ アクションの指定に使用されます。

タスクの状態の詳細は、『*WebLogic Integration Studio ユーザーズ ガイド*』の「ワークフロー テンプレートの定義」の中の「タスク プロパティの定義」を参照してください。

アクションは、次のようなカテゴリにグループ化されます。

- *タスク*アクション ー 特定のタスクの動作を指定
- *ワークフロー*アクション ー ワークフロー全体の動作を指定
- *統合*アクション ー 外部コンポーネントに対するインタフェースの起動
- *例外処理*アクション ー カスタム例外ハンドラを呼び出し
- *その他*のアクション ー E メールの送信や監査エントリの作成など、その他の動作

[タスクのプロパティ] ダイアログ ボックス (またはフォルダ ツリー) の表示で分かるように、ワークフロー変数を設定 タスクには以下のアクションが含まれています。

- アクティブ時
  - ワークフロー変数を設定 ー ワークフローの OrderID 変数を設定する *ワークフロー*アクションです。詳細については、「ワークフローの式 : ワークフロー変数を設定 と ユーザにタスクを割り当て アクションの表示」を参照してください。

- ユーザにタスクを割り当て ー 手動タスクをワークフローのユーザに割り当てる*タスク*アクションです。詳細については、「ワークフローの式：ワークフロー変数を設定 と ユーザにタスクを割り当て アクションの表示」を参照してください。

■ 実行時

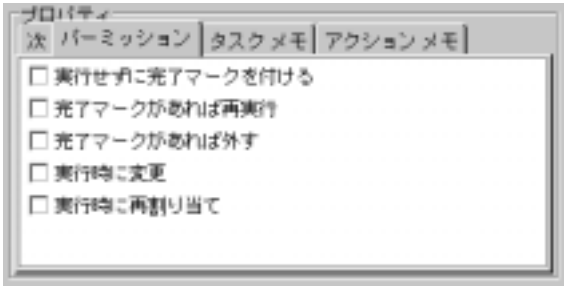
- Post initial XML event ー Order Processing ワークフローで使用される XML メッセージをパブリッシュする*統合*アクション。詳細については、「内部 XML イベントのポスティング：Neworder XML ドキュメントの編集」を参照してください。
- タスクに完了マークを付ける ー 次のノードにフローの進行を許可する*タスク*アクションです。すべてのタスク ノードに対して、フローの進行を指示するこのアクションを指定する必要があります。

第 4 章「ワークフローのノードの定義」で、アクションの定義方法を詳しく説明します。

タスク パーミッションについて

最後に、[パーミッション] タブを使用すると、Worklist ユーザがこのタスクに対して実行できるアクションのタイプを設定できます。デフォルトでは何も設定されません。このシナリオ用のデフォルト設定を、次の図に示します。

図 3-8　Start Order Processing タスク : [パーミッション] タブ



タスク パーミッションの詳細は、*WebLogic Integration Studio ユーザーズ ガイド* の「ワークフロー テンプレートの定義」の「タスク パーミッションの定義」と、『*WebLogic Integration Worklist ユーザーズ ガイド*』の「タスクの処理」を参照してください。

ワークフローの式：ワークフロー変数を設定 と ユーザにタスクを割り当て アクションの表示

[タスクのプロパティ] ダイアログ ボックスの [アクション] リストの下にあるスクロール バーを使用して、[アクティブ時] タブ内の全アクション定義を表示します。アクション定義を以下に示します。

- ワークフロー変数 “OrderID” を式に設定します。$OrderID + 1
- “Start Order Processing” タスクをユーザ WorkflowAttribute ("Initiator") に割り当てます。

これらのアクションの定義の詳細を表示するには、各アクションをダブルクリックします。各アクションのプロパティのダイアログ ボックスが表示されます。

図 3-9 [ワークフロー変数を設定] ダイアログ ボックス

図 3-10 [ユーザにタスクを割り当て] ダイアログ ボックス

このボタンは、このフィールドにワークフロー式を入力する必要があることを示す。このボタンを使用して Expression Builder を起動することもできる。4-31 ページの「タスクとワークフローのコメントの追加：Contact Customer タスクの定義」では、Expression Builder を使用する。

図に示すように、この 2 つのアクションは式を使用して定義します。ノードをワークフローの基本単位、アクションをノードの基本単位と考えれば、式はアクションの基本単位と考えることができます。式とは、値の計算と入力を行うために実行時に評価される文です。式には、変数、定数、リテラル、演算子、定義済み関数を組み合わせることができます。ユーザが定義する最も一般的なタイプの式では、XML ドキュメントから値を抽出する XPath 関数を使用します。

ユーザが知っておく必要のあるワークフロー式の言語の基本的なルールを以下に示します。

- 文字列リテラルはクォーテーション (" ") で囲みます。
- 変数名には、ドル記号 ($) またはコロン (:) を前に付けます。
- 異なる要素のタイプを、プラス記号 (+) 演算子で連結できます。

このように、最初のアクションの式では、OrderID 変数の値に対しワークフローの実行のたびに 1 が加算されるように指定します。第 7 章「サンプル ワークフローの実行とモニタ」では、ワークフロー実行時のアクション内でこのプロセスを確認します。

2 番目のアクションの式は、組み込み*関数*を使用して Start Order Processing タスクが、ワークフローを開始する Worklist ユーザに割り当てられるように指定します。

第 4 章「ワークフローのノードの定義」で、Expression Builder と XPath Wizard を使用して式を作成します。

ワークフロー式の言語と関数の詳細は、『*WebLogic Integration Studio ユーザーズガイド*』の「ワークフロー式の使い方」を参照してください。

内部 XML イベントのポスティング : Neworder XML ドキュメントの編集

XML イベントをポストアクションを表示する手順は次のとおりです。

1. [Start Order Processing タスク プロパティ] ダイアログ ボックスで、[実行時] タブを選択します。
2. XML イベントをポストアクションを選択し、[更新] をクリックします。定義済みの XML ドキュメントが表示されている [XML イベントをポスト] ダイアログ ボックスが表示されます。

図 3-11 [XML イベントをポスト] ダイアログ ボックス

XML イベントをポストする場合、以下のようにさまざまなオプションを指定できます。

- Destination － 内部 JMS キュー、外部 JMS トピックまたはキュー。デフォルトのポスト先は内部キューです。
- Transaction Mode － トランザクションのコミット時、または直ちにメッセージを送信します。デフォルトのトランザクション モードは、トランザクションのコミット時です。
- XML Message － XML 変数内に格納されたドキュメント、またはアクションのダイアログ ボックスで、XML エディタだけを使用して作成したドキュメント。デフォルトの設定は、新規ドキュメントの作成です。

このサンプルでは、デフォルトの設定を使用します。プロダクション環境によっては、種々の JMS メッセージや XML イベントのポスト用のオプションの利用をお勧めします。XML イベントをポスト アクションのオプション全体の詳細は、『*WebLogic Integration Studio ユーザーズ ガイド*』の「アクションの定義」の「JMS トピックまたはキューに対する XML メッセージのポスト」を参照してください。

XML ドキュメントの構造を編集する

このアクションによってポストされる XML ドキュメントは、サーブレットによってワークフローに送信される XML ドキュメントをシミュレートします。このドキュメントには、受注に関する情報が含まれ、以下に示すような構造となっています。

コード リスト 3-1 Neworder XML ドキュメント

```
<neworder>
   <id>$OrderID</id>
   <shiptostate>KK</shiptostate>
   <status>new</status>
   <customer>
       <id>6831</id>
       <name>John Doe</name>
       <address>3126 Blue Street Anytown CA 96822</address>
       <phone>408 534 9567</phone>
       <email>jdoe@bea.com</email>
   </customer>
   <item>
       <id>236</id>
       <name>CD storage rack</name>
<quantity>2</quantity>
   </item>
</neworder>
```

この XML ドキュメント内の各要素の値は、Order Processing ワークフローの開始ノードによって抽出され、ワークフローの各変数内に格納されます。これらの値は、Order Processing ワークフローと Order Fulfillment ワークフロー全体を通して使用されます。変数、ワークフロー内での変数の使用法、対応する XML 要素などについて、その使用方法を、次の表で説明します。

表 3-2 Neworder の XML 要素

XML 要素	対応する ワークフロー変数	ワークフローにおける使用法
`<status>`	OrderStatus	ワークフローのさまざまなポイントで変更できる。
`<id>`	OrderID	ワークフローから値を受け取り、Worklist 内ではワークフロー ラベルとなってワークフローが実行されるたびに加算される。
`<shiptostate>`	ShipToState	`POBean` の `calculate()` メソッドに渡される。初期値は例外ハンドラによって捕らえられる。
`<customer>` `<id>`	CustomerID	信用チェック タスクに使用される。
`<customer>` `<name>`	CustomerName	カスタマ通知タスクに使用され、さらに出荷タスク用に Order Fulfillment ワークフローに渡される。
`<customer>` `<address>`	CustomerAddress	出荷タスク用に Order Fulfillment ワークフローに渡される。
`<customer>` `<phone>`	CustomerPhone	カスタマ通知タスクで使用される。
`<customer>` `<email>`	CustomerEmail	受注確認タスクで使用される場合がある。
`<item>` `<id>`	ItemID	`checkInventory()` メソッドと `calculate()` メソッドに渡される。
`<item>` `<name>`	ItemName	使用されない。
`<item>` `<quantity>`	ItemQuantity	`checkInventory()` メソッドと `calculate()` メソッドに渡される。

各要素には、デフォルト値が入力されています。<shiptostate> 要素に、誤った国名の省略形 KK が入力されています。これは無効なデータが Order Fulfillment ワークフローの POBean EJB に渡される場合をシミュレートして、カスタマ例外ハンドラを呼び出すためです (詳細は、第 6 章「カスタム例外ハンドラの使い方」を参照してください)。

4-64 ページの「電子メール メッセージの送信 : Confirm Order Fulfillment タスクの定義」で、電子メールを送信するアクションを定義します。アクションの定義をする場合、メール アドレスにはこの XML イベントをポスト アクションによって送信された XML ドキュメント内の値を使用します。第 7 章「サンプル ワークフローの実行とモニタ」でワークフローを実行する際に、プロセスの最後に受注の確認用に送信される E メールを受信したい場合には、XML ドキュメントを編集して、実際に使用しているユーザのアドレスを指定できます。

注意 : このプロシージャが実行時に動作するようにするには、ユーザの E メール サーバを、ユーザのサーバ ドメインに適切にコンフィグレーションする必要があります。詳細については、『*WebLogic Integration の起動、停止およびカスタマイズ*』の「WebLogic Integration のカスタマイズ」の「メール セッション プロパティのカスタマイズ」を参照してください。

ユーザ自身のデータを指定するには、XML ドキュメントを次の手順で編集します。

1. [XML イベントをポスト] ダイアログ ボックスの [XML ドキュメント構造] 領域内で、Element ノードを展開してその値を表示します。スクロール バーを使用してほかのフィールドを表示します。

図 3-12 Neworder XML ドキュメントの構造

2. 顧客の [e-mail] 要素値フィールドに、ユーザのメール アドレスをクォーテーション マークで囲んで入力します。〔Enter〕を押して値を入力します。

   注意： 4-64 ページの「電子メール メッセージの送信 : Confirm Order Fulfillment タスクの定義」で電子メール メッセージを送信 アクションを指定する場合は、必ず有効なメール アドレスを入力してください。

3. また、ほかの [customer] の値と [item] の属性値を編集し、そこにユーザ独自の情報も入力できます。

   注意： [shiptostate] 要素値フィールドの間違っているデータは編集しないでください。この値は、実行時に Order Fulfillment ワークフローで例外ハンドラを呼び出すために使用します。

4. 完了したら、[OK] をクリックして変更を保存し、[XML イベントをポスト] ダイアログ ボックスを終了します。

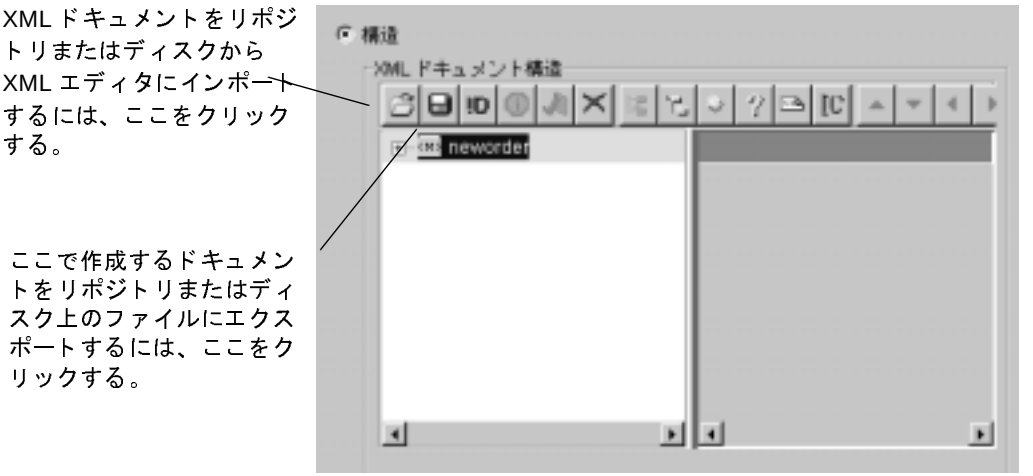
XML リポジトリの使い方 : Neworder XML ドキュメントのエクスポート

Studio で XML ドキュメントの作成や編集を行った場合、その結果をディスク上のファイルに保存するか、XML リポジトリにエクスポートできます。リポジトリは、XML ドキュメント、DTD、スキーマ、スタイル シート、ほかのエンティティをその後のアクセスに備えて格納し、他のノードやワークフローまたはオーガニゼーションで再利用するために定義済みの XML ドキュメントをエクスポートまたはインポートする便利な方法を説明します。

Order Processing ワークフローは、Start Order Processing タスクの XML ドキュメントに格納されているデータを参照する XML ドキュメントをリポジトリに格納することによって、4-10 ページの「XPath 文を定義する」の、ドキュメントからデータを抽出するための XPath 式の定義タスクを簡素化します。

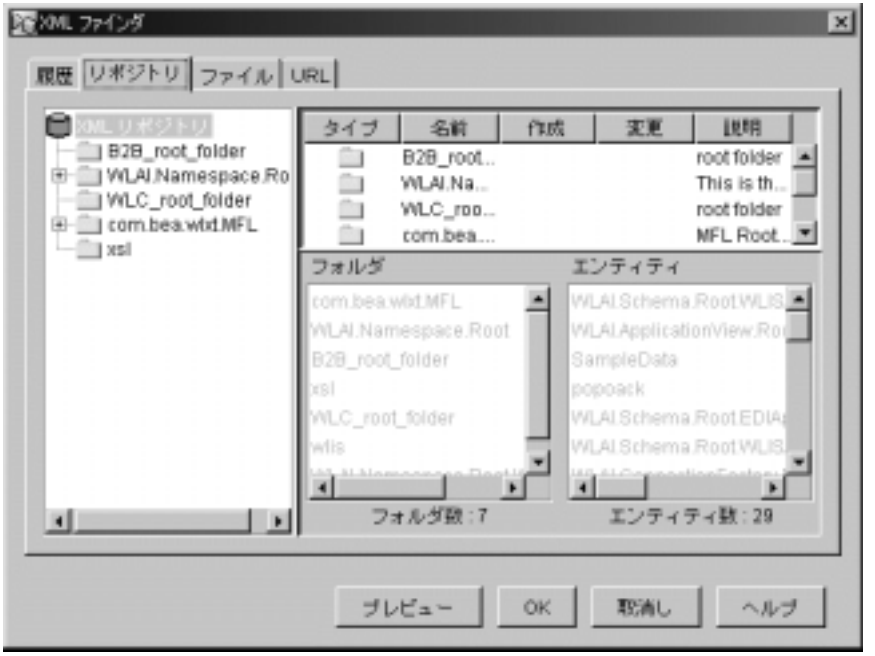
ドキュメントをリポジトリへ保存する手順は次のとおりです。

1. [タスクのプロパティ] ダイアログ ボックスの [実行時] タブの中にある Post internal XML Event アクションをダブルクリックし、[XML イベントをポスト] ダイアログ ボックスを表示します。



2. ツール バーにある [エクスポート] ボタン  をクリックします。[XML ファインダ] ダイアログ ボックスが表示されます。
3. [リポジトリ] タブを選択します。

図 3-13 [XML ファインダ] ダイアログ ボックス



4. XML ドキュメントを保存する新しいフォルダを作成するには、[XML リポジトリ] フォルダを右クリックして、表示されるポップアップ メニューから [フォルダを追加] を選択します。[フォルダを追加] ダイアログ ボックスが表示されます。

図 3-14 XML ファインダ [フォルダを追加] ダイアログ ボックス

5. [名前] フィールドに、たとえば `Tutorial` のように新しいフォルダの名前を入力し、[OK] をクリックします。これで、[リポジトリ] ウィンドウに新しいフォルダが表示されます。
6. [リポジトリ] ウィンドウで新しいフォルダを右クリックし、表示されるポップアップ メニューから [エンティティを追加] を選択します。[エンティティを追加] ダイアログ ボックスが表示されます。

図 3-15　XML ファインダ [エンティティを追加] ダイアログ ボックス

7. [タイプ] ドロップダウン リストから、[XML ドキュメント] を選択します。
8. [名前] フィールドに、たとえば `Neworder` のようなドキュメント名を入力し、[OK] をクリックします。これで新しいドキュメント用の空のコンテナが作成され、フォルダのエンティティ リスト内に表示されます。

図 3-16　[XML ファインダ] ダイアログ ボックス

9. 新しいドキュメントを選択し、[OK] をクリックして [XML ファインダ] ダイアログ ボックスを終了します。このコンテナに XML エディタで定義した XML ドキュメントの内容が格納されます。
10. [OK] をクリックし、[XML イベントをポスト] ダイアログ ボックスを終了します。

11. [OK] をクリックし、[タスクのプロパティ] ダイアログ ボックスを終了します。

テンプレート定義の保存

テンプレート定義を変更すると、フォルダ ツリー内および [ワークフロー設計] ウィンドウのタイトル バー内のテンプレート定義ラベルの前にアスタリスクが付きます。このアスタリスクは、テンプレート定義に未保存の変更が含まれていることを示します。

ワークフローを保存する手順は次のとおりです。

1. フォルダ ツリー内の Order Processing Trigger テンプレートを展開し、ワークフローのテンプレート定義をダブルクリックします。
2. ポップアップ メニューから [保存] を選択します。

これで、チュートリアルの次の節で説明される Order Processing ワークフローの作成と定義が可能となります。

4　ワークフローのノードの定義

注意： Worklist クライアント アプリケーションは WebLogic Integration リリース 7.0 より非推奨となりました。代替機能に関する詳細については、『*WebLogic Integration リリース ノート*』を参照してください。

この章では、ワークフローのノードの詳しい概要に続いて、Order Processing ワークフローの各ワークフロー オブジェクトを定義するための手順をノードごとに説明します。この章の手順に従えば、次の操作方法が習得できます。

- イベント トリガ開始ノードの作成
- XPath Wizard を使用した XPath の作成
- XML ドキュメントの作成と Worklist クライアントへの送信、および Worklist DTD による応答処理
- タスクやワークフローに対するコメントの追加
- イベントとイベント キーの定義
- 分岐の作成と定義
- EJB メソッドを呼び出すビジネス オペレーションのコンフィグレーションと実行
- プラグイン アクションの使い方
- サブワークフローの呼び出し
- 電子メール メッセージの送信

この章で説明する手順を行う前に、以下のことが完了していることを確認してください。

- 2-7 ページの「Tutorial.jar ワークフロー オブジェクトのインポート」の説明に従って、Create OrderBean のビジネス オペレーションのインポートが完了していること。

- 第 5 章「ワークフローの作成」の説明に従って、Order Processing ワークフロー テンプレートとテンプレート定義の作成、作図、アクティブ化が完了していること。
- （省略可能）3-15 ページの「内部 XML イベントのポスティング : Neworder XML ドキュメントの編集」の説明に従って、有効な 電子メール アドレスを、Order Processing Trigger ワークフロー内の XML イベントをポスト アクションに指定済みであること。

Order Processing ワークフロー ノードの概要

このワークフローの詳細を、次の図に示します。図中で番号が振られている手順については、図の下にある表で説明します。この図では、実際のプロセスをワークフロー 実装にマップしています。

図 4-1　Order Processing のノード：詳細表示

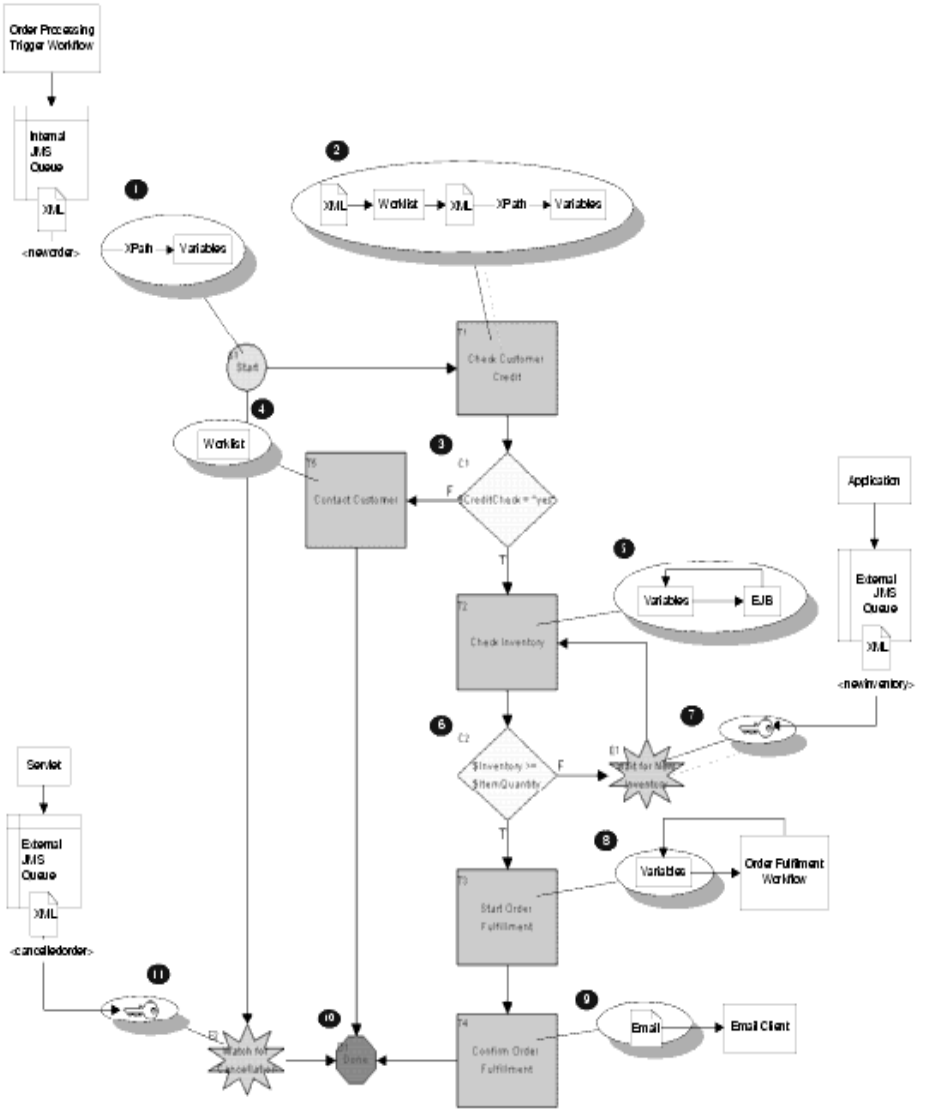


表 4-1 Order Processing ワークフローの概要

プロセス	実装
処理システムによってデータが読み取られ、ID 番号が付けられて受理されます。 実際の実装では、イベント型の開始ノードは、サーブレットが外部 JMS キューにポストした XML メッセージに反応するように設定されている。	1. イベントトリガ開始ノードは、Order Processing Trigger ワークフローによって内部 JMS に送信された <neworder> XML メッセージをリスンするよう設定される。XPath 文は XML ドキュメントからデータを抽出し、ワークフロー変数を登録する。
その受注はカスタマサービスに転送され、顧客の信用情報がチェックされます。	2. 信用チェックの合否確認（はい または いいえ）を担当者に求める XML メッセージが admin ユーザに送信される。応答は XML メッセージで返され、XPath 文によってワークフロー変数に読み込まれる。
	3. 分岐ノードで、応答が はい か いいえ のいずれであるかを評価する。
信用チェックができなかった場合、顧客サービス担当者は顧客に通知して、正しい信用情報を取得する。これ以降のプロセスは手動で行う。	4. 顧客との連絡タスクが CustomerService ロールに割り当てられ、ワークフローは [完了] ノードに進む。
信用チェックが通ると、商品 ID を基に受注商品の現在の目録がデータベースでチェックされ、出荷可能な商品の数量と受注数量が比較されます。	5. 商品 ID が EJB 内のメソッドに渡される。EJB 内のメソッドによってデータベースがチェックされ、受注商品の現在の在庫が変数に返される。
	6. 分岐によって、返された在庫の値が、受注数量以上あるかどうかが評価される。受注数量に足りない場合は、受注ステータスは待機に設定される。

表 4-1 Order Processing ワークフローの概要 （続き）

プロセス	実装
在庫が十分ではない場合は、次の新規入庫の到着までその受注は保留されます。新規入庫の在庫表が受領されると、その在庫が受注に対応可能であると確認できるまで、手順 5 が繰り返されます。 実際の実装においては、イベントは、在庫報告アプリケーションから外部 JMS キューにポストされる XML メッセージに反応するように設定される。	7. イベントとイベント キーは、該当する商品 ID を含む仮想の XML `<checkinventory>` メッセージに反応するように設定される。イベントがトリガされると、フローは在庫チェック タスク ノードにループバックされる。
在庫が十分な場合は、受注が 2 人の担当者に同時に転送される。担当者の 1 人は出荷を担当し、もう 1 人は受注に対する請求書生成をシステムに指示する。	8. Order Fulfillment ワークフローが呼び出され、変数が渡される。次にそのワークフロー内で実行された演算結果が返される。
次に、受注品が出荷されたことがシステムで確認され、顧客に 電子メールで通知されます。	9. Order Fulfillment ワークフローの終了時、および Watch for Cancellation イベントのキャンセル時には、受注ステータスが完了に設定される。また、電子メールが顧客に送信される場合もある。
	10. ワークフローが終了する。
出荷前のトランザクションにおけるどの時点でも、顧客からの通知により受注をキャンセルできます。 実際の実装においては、イベントは、サーブレットが外部 JMS キューにポストした XML ドキュメントに反応するように設定される。	11. イベントとイベント キーは、関連する受注 ID を含む仮想の `<cancelledotder>` XML メッセージに反応するように設定される。イベントがトリガされると、ワークフローは完了ノードに進み、終了する。

イベントトリガによる開始の作成

Order Processing ワークフローは、Order Processing Trigger ワークフローがポストした XML イベントによってトリガされます。ワークフローが開始すると、まず、ワークフロー変数を Order Processing Trigger ワークフローがポストした XML ドキュメントの値に初期化します。

そのため、ノードの作成には、次の作業が必要になります。

- XML ドキュメント内のデータに対応する変数の作成
- ワークフローをトリガするイベントの設定
- XML ドキュメントからデータを抽出し、そのデータを変数に割り当てる XPath 式の定義

変数を作成する

開始ノードによって XML ドキュメントから値が抽出されます。そのため、対応する変数を設定して抽出した値をその中に格納できるようにする必要があります。変数、ワークフロー内での用法、対応する XML 要素を次の表に示します。変数の値をほかのワークフローに渡すときには、*input* パラメータを使用します。一方、*output* パラメータは、現在のワークフローが他のワークフローから変数の値を受け取るときに使用します。

表 4-2 開始ノード変数

変数	データ型	パラメータタイプ	ワークフローでの使い方	対応する XML 要素
OrderStatus	文字列	Input	ワークフローのさまざまなポイントで変更できる。	`<status>`
OrderID	整数	Input	Worklist 内にワークフロー ラベルとして表示し、ワークフローの実行のたびに加算する。	`<id>`
ShipToState	文字列	Input、Output	Order Fulfillment ワークフローによって `POBean` の `calculate()` メソッドに渡される。初期値は例外ハンドラが検出し、修正のため Worklist ユーザに渡される。 ユーザから取得した修正後の値が Order Processing ワークフローに返される。	`<shiptostate>`

表 4-2 開始ノード変数 （続き）

変数	データ型	パラメータタイプ	ワークフローでの使い方	対応する XML 要素
CustomerID	文字列	なし	Worklist ユーザが信用チェックタスクで顧客を識別するために使用する。	`<customer>` `<id>`
CustomerName	文字列	Input	Worklist ユーザが顧客通知 タスクで顧客を識別するために使用する。 出荷タスク用に Order Fulfillment ワークフローに渡される。	`<customer>` `<name>`
CustomerAddress	文字列	Input	出荷タスク用に Order Fulfillment ワークフローに渡される。	`<customer>` `<address>`
CustomerPhone	文字列	なし	Worklist ユーザへ顧客の連絡情報を提供する顧客通知タスクで使用される。	`<customer>` `<phone>`
CustomerEmail	文字列	なし	受注確認タスクで使用される。	`<customer>` `<email>`
ItemID	整数	Input	`checkInventory()` メソッドと `calculate()` メソッドに渡される。	`<item>` `<id>`
ItemName	文字列	なし	使用されない。	`<item>` `<name>`
ItemQuantity	整数	Input	`CheckInventory()` メソッドの結果と比較するために分岐ノードで使用され、Order Fulfillment ワークフローによって `calculate()` メソッドに渡される。	`<item>` `<quantity>`

変数を作成する手順は、次のとおりです。

1. Order Processing のテンプレート定義フォルダを展開し、[変数] を右クリックして、表示されるポップアップ メニューから 変数を作成 を選択します。[変数プロパティ] ダイアログ ボックスが表示されます。

図 4-2 [変数プロパティ] ダイアログ ボックス

2. 表 4-2 に従って [名前] フィールドに変数名を入力します。

   注意： 変数名にはスペースは入れません。

3. 表 4-2 に従って、[タイプ] ドロップダウン リストから変数に対応するデータ型を選択します。

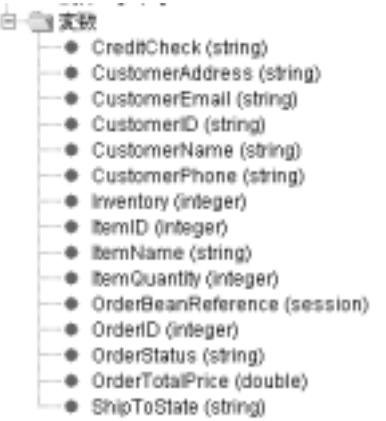
4. 表 4-2 に従って、必要な [パラメータ] のチェック ボックスをチェックします。

5. [OK] をクリックします。新しい変数がフォルダ ツリーの [変数] フォルダ内に表示されます。

6. 表 4-2 にリストした変数をすべて作成するまで、手順 1 から 5 までを繰り返す。

   注意： OrderID 変数は、5-11 ページの「ワークフロー ラベルの追加」ですでに作成済みのため、ここで作成する必要はありません。

終了すると、[変数] フォルダ ツリーは次のようになります。



開始ノード イベントを定義する

正しい XML ドキュメントによって確実にワークフローをトリガするには、Order Processing Trigger にポストされた XML ドキュメントのルート要素、つまり <neworder> に対応する開始ノードのルート要素を指定します。

開始ノードは、XML ドキュメントの特定のインスタンスに関連付けられるのではなく、ルート要素 <neworder> を持つすべての受信 XML ドキュメントによりトリガされるため、ワークフロー内で定義する他のイベントの場合のようにイベント キーを設定する必要はありません (イベントの作成 : Watch for Cancellation イベントの定義」に例がありますので、参照してください)。

イベントをセットアップする手順は、次のとおりです。

1. Order Processing ワークフロー図で開始ノードをダブルクリックして [開始のプロパティ] ダイアログ ボックスを表示します。
2. [イベント] オプションを選択して、ドロップダウンリストから [XML イベント] を選択します。

図 4-3 [開始のプロパティ] ダイアログ ボックスの イベント

3. [ドキュメント タイプ / ルート要素] フィールドに `neworder` と入力します。
4. [開始オーガニゼーション] ドロップダウン リストから、「CDExpress」を選択します。

XPath 文を定義する

XML ドキュメントから値を抽出し、その値をワークフローの変数に挿入するには、データベース上のクエリ言語の文と同じように機能する XPath 言語の文を使用します。XPath は多くの機能を実現する多彩な言語ですが、もっともよく使用する関数は、要素または属性の値を返す `text()` です。ここで使用する XPath の `text()` 関数文の一般的形式は、次のとおりです。

- 要素の値

  `XPath("/root element/child element/text()")`

- 属性の値

  XPath("/*root element*/@*attribute*/text()")

XML ドキュメントから値を抽出するには、次の表に示すような XPath 文の設定が必要です。

表 4-3 開始ノード： XPath 式

変数	対応する XML 要素	値の抽出に必要な XPath 式
OrderStatus	<status>	XPath("/neworder/status/text()")
OrderID	<id>	XPath("/neworder/id/text()")
ShipToState	<shiptostate>	XPath("/neworder/shiptostate/text()")
CustomerID	<customer> <id>	XPath("/neworder/customer/id/text()")
CustomerName	<customer> <name>	XPath("/neworder/customer/name/text()")
CustomerAddress	<customer> <address>	XPath("/neworder/customer/address/text()")
CustomerPhone	<customer> <phone>	XPath("/neworder/customer/phone/text()")
CustomerEmail	<customer> <email>	XPath("/neworder/customer/email/text()")
ItemID	<item> <id>	XPath("/neworder/item/id/text()")
ItemName	<item> <name>	XPath("/neworder/item/name/text()")
ItemQuantity	<item> <quantity>	XPath("/neworder/item/quantity/text()")

Studio 内で XPath 文を作成するには、以下の 3 つのいずれかの方法を使用します。順に有効性が高くなります。

- 式を手動で入力する－この方法では有効性チェックが行われません。したがって、XPath 文が無効でも拒否されないため、エラーを自分で探す必要があります。
- Expression Builder を使用して式の一部を作成する－この方法でも式の大部分を手動で入力する必要がありますが、無効な文を入力しようとすると、エラーメッセージが表示されて正しく入力するよう求められます。
- XPath Wizard を使用して、格納されている XML エンティティから自動的に式を生成する－この方法を使用すると、実際の XML エンティティのセクションを選択し、それを式でテストできます。この方法は、ここで挙げた 3 つの方法の中で、もっとも高度な支援と有効性チェックが得られます。

次の手順では、XPath Wizard を使用して、3-21 ページの「XML リポジトリの使い方 : Neworder XML ドキュメントのエクスポート」でエクスポートした `Neworder` ドキュメントをロードします。

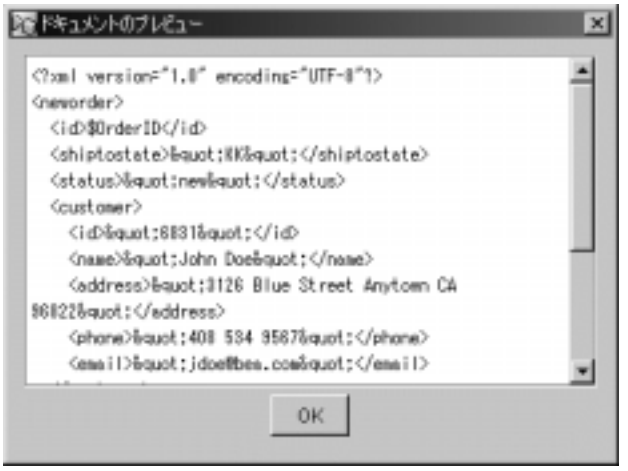
XPath Wizard を使用して XPath 式を定義する手順は、次のとおりです。

1. [開始のプロパティ] ダイアログ ボックス下部の [変数] タブを選択し、[追加] ボタンをクリックします。[ワークフロー変数の割り当て] ダイアログ ボックスが表示されます。
2. [式] フィールドの隣にある  ボタンをクリックします。[Expression Builder] ダイアログ ボックスが表示されます。
3. [XPath Wizard] をクリックします。XPath Wizard が表示されます。
4. [サンプル] タブを選択します。
5. [XML、DTD、XSD、XSL、または MFL ドキュメントを開く] ボタン  をクリックします。[XML ファインダ] ダイアログ ボックスが表示されます。
6. [リポジトリ] タブを選択します。
7. 3-21 ページの「XML リポジトリの使い方 : Neworder XML ドキュメントのエクスポート」で XML エンティティを保存したフォルダ (たとえば、`Tutorial`) をダブルクリックします。

図 4-4 [XML ファインダ] ダイアログ ボックス : [リポジトリ] タブ

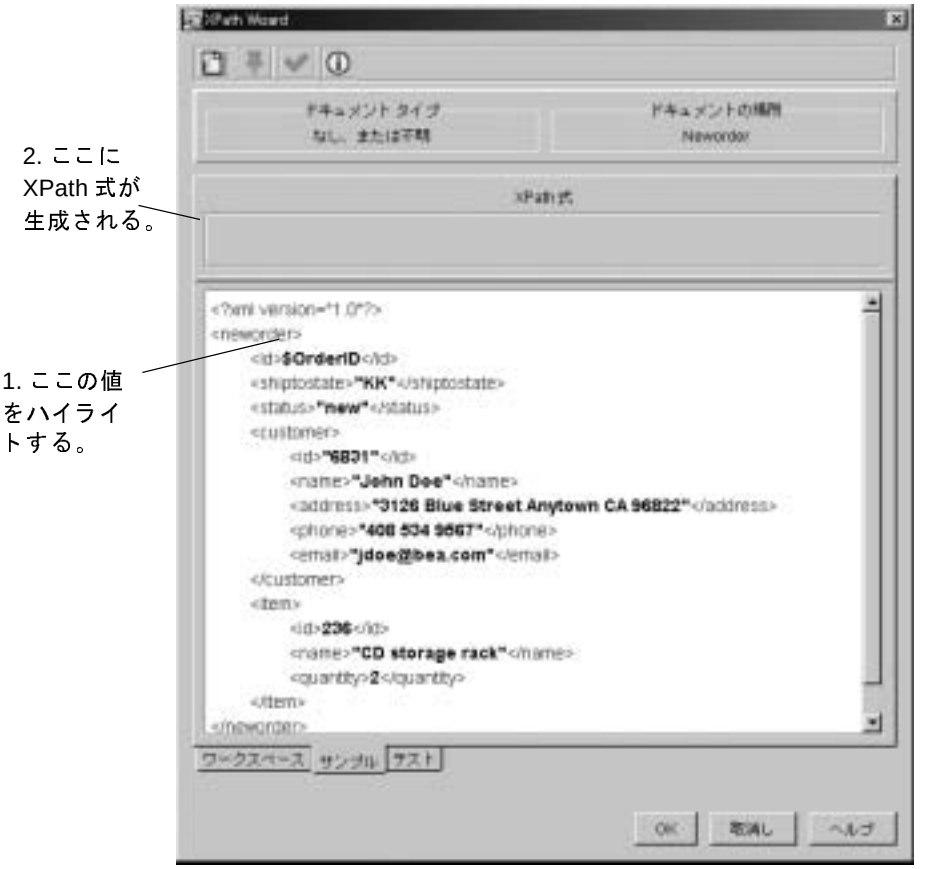
8. [Neworder] ドキュメントを選択し、[プレビュー] をクリックして、ドキュメントのプレビューを表示します。[ドキュメントのプレビュー] ウィンドウが表示され、3-21 ページの「XML リポジトリの使い方 : Neworder XML ドキュメントのエクスポート」で保存した XML ドキュメントの実際のテキスト バージョンが表示されます。

図 4-5 [ドキュメントのプレビュー] ウィンドウ



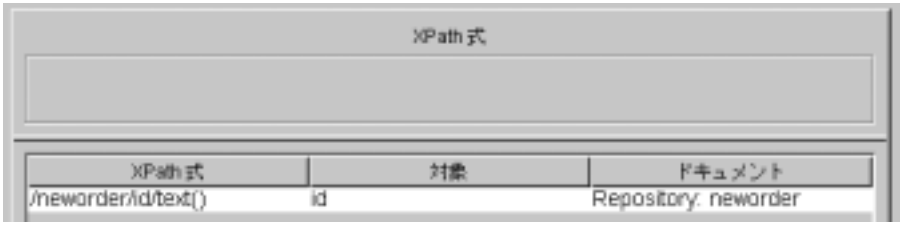
9. [OK] をクリックして [ドキュメントのプレビュー] ウィンドウを閉じます。
10. [OK] をクリックし、XML Finder を終了します。XML ドキュメントが、XPath Wizard の [サンプル] ウィンドウに表示されます。

図 4-6　XPath Wizard の [サンプル] タブ



11. 次のいずれかの方法を行います。
    - 最初の要素の値をクリックしてハイライトします。その XPath 文が [Xpath 式] フィールドに自動的に生成されます。
    - [テスト] タブを選択して [Xpath 式] フィールドに XPath 文を入力し、[XPath 式をテスト] ボタン  をクリックして、ドキュメント内で [XPath Wizard] によって選択された値が、取り込もうとした値かどうかをチェックして、式の有効性を検証します。
12. [ワークスペースに式を固定] ボタン  をクリックします。[ワークスペース] 内に式が表示されます ([ワークスペース] タブを選択して表示)。

図 4-7 XPath Wizard の [ワークスペース] タブ



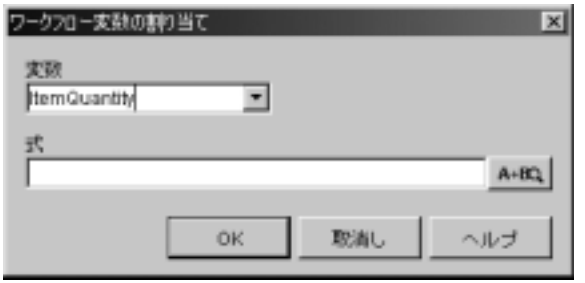
13. 表 4-3 内の XML 要素全部に対して手順 10 と 11 を繰り返します。すべて終了すると、[ワークスペース] は次のようになります。

14. [OK] をクリックして、XPath Wizard を終了します。Expression Builder を終了するには [OK] をクリックし、[ワークフロー変数の割り当て] ダイアログボックスを終了するには [取消し] をクリックします。

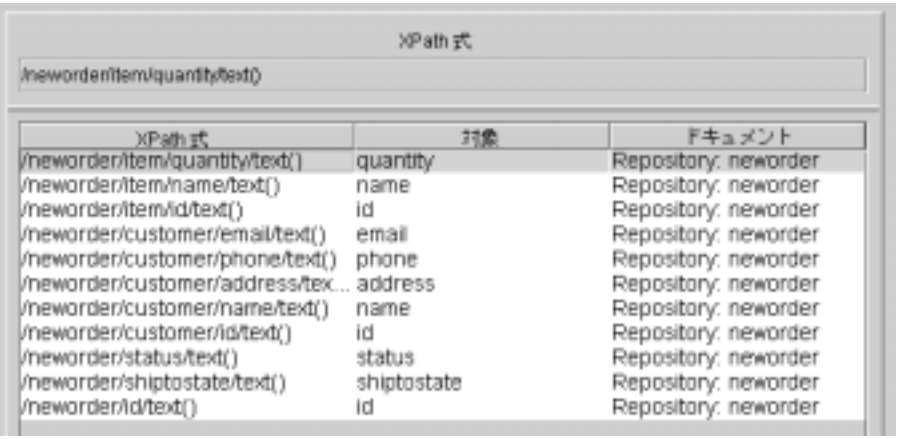
XPath 式を [開始のプロパティ] ダイアログ ボックスの 変数 リストに追加する手順は、次のとおりです。

1. [開始のプロパティ] ダイアログ ボックス下部の [変数] タブを選択し、[追加] ボタンをクリックします。[ワークフロー変数の割り当て] ダイアログ ボックスが表示されます。

図 4-8 [ワークフロー変数の割り当て] ダイアログ ボックス

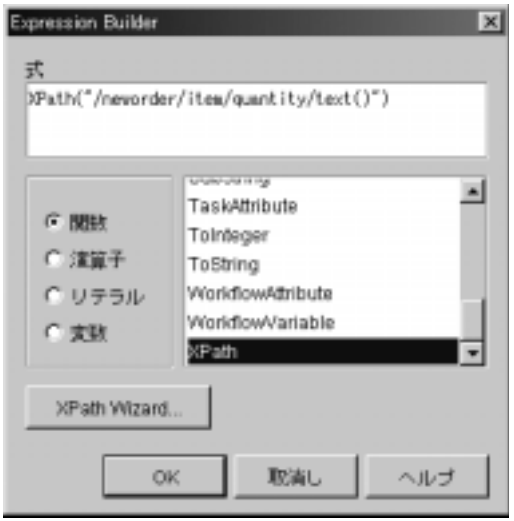


2. [変数] ドロップダウン リストから、変数を選択します。
3. A+B ボタンをクリックし、[Expression Builder] ダイアログ ボックスを開きます。
4. [XPath Wizard] をクリックします。
5. [XPath Wizard] ダイアログ ボックスで、[ワークスペース] タブを選択します。



6. 表 4-3 内のリストから変数に対応する XPath 式を選択し、[OK] をクリックします。これで、[Expression Builder] ダイアログ ボックスの [式] ウィンドウに、この式が表示されます。

図 4-9 [Expression Builder] ダイアログ ボックス



7. [OK] をクリックします。[ワークフロー変数の割り当て] ダイアログ ボックスの [式] フィールドにこの式が表示されます。

8. [OK] をクリックします。新しい XPath 式が [開始のプロパティ] ダイアログ ボックスの [変数] タブに表示されます。

図 4-10 [開始のプロパティ] ダイアログ ボックスの [変数] タブ

9. 表 4-3 にある変数と式のすべてに対して、手順 1 から 8 までを繰り返します。
10. すべて終了したら、[OK] をクリックして [開始のプロパティ] ダイアログ ボックスを終了します。

XML メッセージの Worklist ユーザへの送信 : Check Customer Credit タスクの定義

このタスクによって、顧客の信用チェック作業を Worklist 内のユーザ admin に割り当てられ、次に XML メッセージがユーザ admin に送信されて、承認の確認が求められます。

タスクの割り当てに必要な作業は、次のとおりです。

- ユーザにタスクを割り当て アクションの追加と定義
- XML をクライアントに送信 アクションの追加と定義

タスクをユーザに割り当てる

ここでは、Check Customer Credit タスクを、ユーザ admin に割り当てる方法を説明します。

このノードにアクションを追加する前に、テンプレート定義を作成したときにすべての T1 タスク ノードに定義した、デフォルトのアクションとパーミッションを削除する必要があります。

デフォルトのアクションを削除する手順は、次のとおりです。

1. ワークフロー設計領域で、[Check Customer Credit] タスク ノードをダブルクリックして、[タスク プロパティ] ダイアログ ボックスを表示します。
2. [アクティブ時] タブを選択し、デフォルトの (“Initiator”) タスクをユーザ WorkflowAttribute (“Initiator”) に割り当てアクションを選択して [削除] ボタンをクリックします。確認を求めるダイアログ ボックスが表示されたら、[はい] をクリックします。
3. [実行時] タブを選択し、デフォルトの タスクに完了マークを付ける アクションを選択して [削除] ボタンをクリックします。確認を求めるダイアログ ボックスが表示されたら、[はい] をクリックします。

ユーザにタスクを割り当て アクションの追加と定義の手順は、次のとおりです。

1. [タスク プロパティ] ダイアログ ボックスで、[アクティブ時] タブを選択し、[追加] をクリックして [アクションを追加] ダイアログ ボックスを開きます。
2. [タスク アクション] フォルダを展開し、ユーザにタスクを割り当て を選択して [OK] をクリックします。[ユーザにタスクを割り当て] ダイアログ ボックスが表示されます。

図 4-11 [ユーザにタスクを割り当て] ダイアログ ボックス

3. ダイアログボックスの [割り当てるタスク] リストから [Check Customer Credit] タスクを選択します。
4. [ユーザ] ドロップダウン リストから [admin] を選択し、[OK] をクリックします。タスクをユーザに割り当て が [タスク プロパティ] ダイアログ ボックス内の [アクティブ時] タブに表示されます。

XML メッセージをクライアントに送信する

クライアント アプリケーションのユーザと対話する方法の 1 つは、クライアント アプリケーションに送信する XML ドキュメントを定義することです。この XML ドキュメントを識別して適切に応答し、またワークフロー変数の登録に必要な情報と共に XML ドキュメントをサーバに送信するよう、アプリケーションをプログラムする必要があります。

デフォルトでは、Worklist アプリケーションは、8 タイプの要求および応答 Document Type Definitions (DTD) からなるデフォルト セットの DTD を認識します。このセットは、WebLogic Integration のインストール ディレクトリの `WLI_HOME/docs/apidocs/com/bea/wlpi/common/doc-files` で表示できます。デフォルト セットは次の表のとおりです。

注意： クライアント アプリケーションによって認識される XML ドキュメントのタイプを拡張するカスタム コードや、API を通じてクライアント アプリケーションと通信するカスタム コードを開発することもできます。詳細については、『*BPM クライアント アプリケーション プログラミング ガイド*』を参照してください。

表 4-4 Worklist の DTD

要求 DTD	説明	対応する応答 DTD
`ClientMsgBoxReq.dtd`	ユーザ向けのボタン付きのメッセージ ボックスを表示する。	`ClientMsgBoxResp.dtd`
`ClientSetVarsReq.dtd`	ユーザ向けのデータ入力フィールド付きのメッセージ ボックスを表示する。	`ClientSetVarsResp.dtd`
`ClientCallPgmReq.dtd`	クライアント コンピュータのプログラムを呼び出す	`ClientCallPgmResp.dtd`
`ClientCallAddInReq.dtd`	Worklist アプリケーションに対するカスタム アドインを呼び出す。	`ClientCallAddInResp.dtd`

ここで使用する Check Customer Credit タスクには、[はい] ボタン、[いいえ] ボタンのついたメッセージ ボックスを送信する `ClientMsgBoxReq.dtd` を使用します。DTD の構造は次のとおりです。

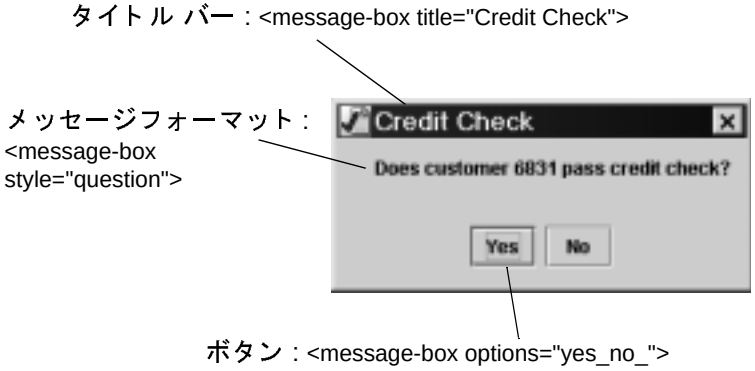
コード リスト 4-1 ClientMsgBoxReq XML ドキュメントの構造

```
<message-box title="text"
 style="{plain|information|question|warning|error}"
 options="{ok|ok_cancel|yes_no|yes_no_cancel}">
text
   <actionid>provided by default</actionid>
</message-box>
```

注意 : アクション ID の要素と値は、XML をクライアントに送信 アクションの作成時に、システムによって指定されます。

送信されるメッセージ ボックスに対応するタグと値を次の図に示します。

図 4-12 [Credit Check] メッセージ ボックス



ここでは送信メッセージのテキストで顧客 ID を使用するため、実行時に対応する値を渡す CustomerID 変数を使用します。

ユーザのボタン選択の結果を取り込むには、コード リスト 4-2 で示すように、`ClientMsgBoxResp.dtd` から、特に `<message-box option>` 属性から、データを抽出する必要があります。

コード リスト 4-2 ClientMsgBoxReq XML ドキュメントの構造

```
<message-box option="{ok|yes|no|cancel}" />
```

そのため、アクション全体を定義するには、次の操作が必要です。

- XML をクライアントに送信アクションを追加し、XML メッセージに対するユーザの応答を格納する変数の作成
- XML ドキュメント構造体の作成
- メッセージ ボックスに対するユーザの応答を抽出するための XPath 式を持つコールバック変数の指定
- タスクに完了マークを付ける コールバック アクションの追加と定義

XML をクライアントに送信 アクションを追加する

Check Credit タスクへのユーザの入力結果を格納するために、CreditCheck 変数を作成する必要があります。

変数は [XML をクライアントに送信] ダイアログ ボックスで作成できるため、XML をクライアントに送信 アクションを追加し、同時に CreditCheck 変数を作成します。

XML をクライアントに送信 アクションを追加し、CreditCheck 変数を作成する手順は次のとおりです。

1. [タスク プロパティ] ダイアログ ボックスで、[実行時] タブを選択し、[追加] をクリックします。[アクションを追加] ダイアログ ボックスが表示されます。
2. [統合アクション] フォルダを展開し、[XML をクライアントに送信] を選択して [OK] をクリックします。デフォルトのルートと actionid 要素が表示された [XML をクライアントに送信] ダイアログ ボックスが表示されます。

3. ルート要素の隣の値フィールドをダブルクリックして、[式] ボタン  を表示します。
4. [式] ボタンをクリックし、Expression Builder を起動します。
5. Expression Builder の中で、[変数] オプションを選択します。
6. 変数リストをスクロールして New を表示し、New をダブルクリックして [変数プロパティ] ダイアログ ボックスを開きます。
7. 変数を次の表に従って作成します。[OK] をクリックし、[変数プロパティ] ダイアログ ボックスを終了します。

表 4-5 Check Customer Credit タスク変数

変数	データ型	パラメータタイプ	ワークフローでの使い方	対応する XML 要素または属性
CreditCheck	文字列	なし	Worklist ユーザの応答結果を格納する。	`<message-box option>`

8. [OK] をクリックし、[Expression Builder] を終了します。

XML ドキュメント構造体を定義する

message-box XML ドキュメント構造体を定義する手順は、次のとおりです。

1. [XML をクライアントに送信] ダイアログ ボックスで、ルート要素を展開します。

   警告： `actionid` 要素、その名前および値を、編集、削除または移動しないでください。アクション ID は、実行時にアクション インスタンスを識別するために使用されます。

2. XML ドキュメントの作成を始めるには、ルート要素をダブルクリックし、既存のテキストに上書きして `message-box` と入力します。〔Enter〕を押します。

   注意：  アイコンが表示されている場合は、フィールド内に有効な式が入力されていないことを示します。フィールドに値を入力すると、このアイコンは消えます。

3. [message-box] 要素の隣の値フィールドをダブルクリックして、タイプするか、Expression Builder を使用して次のように入力します。

   ```
   “Does customer ” + $CustomerID + “ pass credit check?”
   ```

   文字列リテラルと変数を分離するために、必ず引用符の内部にスペースを挿入してください。

4. [message-box] 要素を選択し、[属性を追加 (Ctrl+A)] ボタン  をクリックして最初の属性を追加します。

5. [属性] フィールドで、既存のテキストをダブルクリックし、属性名 `options` を上書きして追加します。

6. [オプション] 属性の隣の値フィールドで、既存のテキストをダブルクリックして、テキストに上書きして `"yes_no"` と入力します。

7. [message-box] 要素を選択し、[属性を追加 (Ctrl+A)] ボタンをクリックして 2 番目の属性を追加します。

8. [属性] フィールドで、テキストをダブルクリックして属性名 `style` をテキストに上書きして追加します。

9. [オプション] 属性の隣の値フィールドで、テキストをダブルクリックして `"question"` とテキストに上書きして入力します。

10. [message-box] 要素を選択し、[属性を追加 (Ctrl+A)] ボタンをクリックして 3 番目の属性を追加します。
11. [属性] フィールドで、テキストをダブルクリックして属性名 `title` とテキストに上書きして追加します。〔Enter〕を押します。
12. [style] 属性の隣の値フィールドで、テキストをダブルクリックして属性名 `"Credit Check"` を上書きして入力し、〔Enter〕を押します。

    これで、完成した XML ドキュメント構造体は、次のようになります。

コールバック変数を割り当てる

message box に対するユーザの応答結果を、Worklist アプリケーションによって返された XML ドキュメントから抽出するには、XPath 式をセットアップして CreditCheck 変数を登録する必要があります。

コールバック変数を割り当てる手順は、次のとおりです。

1. [XML をクライアントに送信] ダイアログ ボックス下部の [コールバック変数] タブを選択します。
2. [追加] ボタンをクリックします。[ワークフロー変数の割り当て] ダイアログ ボックスが表示されます。
3. [変数] ドロップダウン リストから [CreditCheck] を選択します。

4. [Event Expression] フィールドに次の表の XPath 文を入力します。

表 4-6 [Check Customer Credit] タスク ノードの XPath 式

変数	対応する XML 要素	値の抽出に必要な XPath 式
CreditCheck	<message-box option>	XPath("message-box/@option/text()")

5. [OK] をクリックして終了します。[コールバック変数] タブに新しい変数割り当て文が表示されます。

参考： また、ClientMsgBoxResp.dtd ファイルを XPath Wizard にロードしても、XPath 式をビルドする XML ドキュメントを自動的に生成できます。XPath Wizard を使用する手順については、4-10 ページの「XPath 文を定義する」を参照してください。

タスク終了をマークする

タスク ノードが message box に対するユーザの応答を確実に待ってから次のノードに移行するようにするので、コールバック アクションとしての タスクに完了マークを付ける を、[タスク プロパティ] ダイアログ ボックスに追加するのではなく、[XML をクライアントに送信] ダイアログ ボックスに指定します。タスクに完了マークを付ける アクションをここに配置することで、ワークフローは Worklist クライアントからの応答を待って、次のノードに進むようになります。

タスクに完了マークを付ける コールバック アクションを追加する手順は、次のとおりです。

1. [XML をクライアントに送信] ダイアログ ボックス下部の [コールバック アクション] タブを選択します。
2. [追加] ボタンをクリックして [アクションを追加] ダイアログ ボックスを表示します。
3. [タスク アクション] フォルダを展開し、[タスクに完了マークを付ける] を選択して [OK] をクリックします。[タスクに完了マークを付ける] ダイアログ ボックスが表示されます。

図 4-13 [タスクに完了マークを付ける] ダイアログ ボックス

4. [完了マークを付けるタスク] リストから [Check Customer Credit] を選択し、[OK] をクリックします。タスクに完了マークを付ける アクションが [コールバック アクション] タブに表示されます。
5. [OK] をクリックし、[XML をクライアントに送信] ダイアログ ボックスを終了します。XML をクライアントに送信 アクションが [タスク プロパティ] ダイアログ ボックスの [実行時] タブに表示されます。
6. [OK] をクリックし、[タスク プロパティ] ダイアログ ボックスを終了します。

等価テスト：Check Credit 分岐の定義

サンプル ワークフローにおける最初の分岐では、Check Credit タスクでの Worklist ユーザの応答が はい か いいえ のどちらであるかによって、顧客が信用チェックに合格したかどうかをチェックします。

分岐ノードの名前は、評価する条件によって決まり、この条件を式として指定する必要があります。一般的に、条件は、変数の値を定数や別の変数などのほかの要素と比較します。デフォルトでは、分岐には 1=1 の条件（常に true）が割り当てられています。

信用チェックの問い合わせに対する Worklist ユーザの応答をチェックするために、定数との等価テストを作成します。つまり、変数 CreditCheck に格納されたユーザからの応答が、"yes" と等価かどうかをテストします。条件によって true と評価された場合、フローは Check Inventory タスクに進み、false と評価された場合は、Contact Customer タスクに進みます。

C1 分岐ノードを定義する手順は、次のとおりです。

1. [ワークフロー設計] ウィンドウ内で、[C1] 分岐シェイプをダブルクリックして [分岐のプロパティ] ダイアログ ボックスを表示します。
2. [条件] フィールドに `$CreditCheck="yes"` と入力します。
3. [OK] をクリックして、ダイアログ ボックスを閉じます。

タスクとワークフローのコメントの追加： Contact Customer タスクの定義

信用チェックの分岐が false となった場合、フローは Contact Customer タスクに進んで終了します。このノードは、変数 OrderStatus を 取消し に設定し、それを表すワークフロー コメントを追加します。ワークフローが実行されると、このコメントが実行中のアプリケーションに表示されます。

顧客への連絡タスクは特定のユーザではなく、CustomerService ロールに割り当てられます。このため、そのロールに関連付けられているすべてのユーザが、そのタスクを実行できます。また、XML メッセージを通じて Worklist ユーザと通信するのではなく、実行時に [Worklist] ウィンドウにタスクと共に表示されるタスク コメントを設定します。

注意： ワークフローおよびタスクのコメントは最大 254 文字まで。式の変数の長さにより、コメントの長さは実行時に決まる。254 文字の限度を超えるコメントは超過部分が実行時に警告無く短縮される。

[Contact Customer] タスク ノードを定義する手順は、次のとおりです。

- ワークフロー変数を設定 アクションの追加と定義
- ワークフロー コメントを設定 アクションの追加と定義
- タスク コメントを設定 アクションの追加と定義
- タスクにロールを割り当て アクションの追加と定義
- タスクに完了マークを付ける アクションの追加と定義

ワークフローの変数を設定する

ワークフロー変数を設定 アクションの追加と定義の手順は、次のとおりです。

1. [Contact Customer] タスク ノードをダブルクリックして、[タスク プロパティ] ダイアログ ボックスを開きます。

2. [アクティブ時] タブを選択し、[追加] ボタンをクリックして [アクションを追加] ダイアログ ボックスを開きます。
3. [ワークフロー アクション] フォルダを展開し [ワークフロー変数を設定] を選択して [OK] をクリックします。[ワークフロー変数を設定] ダイアログ ボックスが表示されます。

図 4-14 [ワークフロー変数を設定] ダイアログ ボックス

4. [設定する変数] ドロップダウン リストから "OrderStatus" を選択します。
5. [式] オプションを選択し、フィールド内に `"cancelled"` と入力します。
6. [OK] をクリックします。ワークフロー変数を設定 アクションが、[タスク プロパティ] ダイアログ ボックス内の [アクティブ時] タブに表示されます。

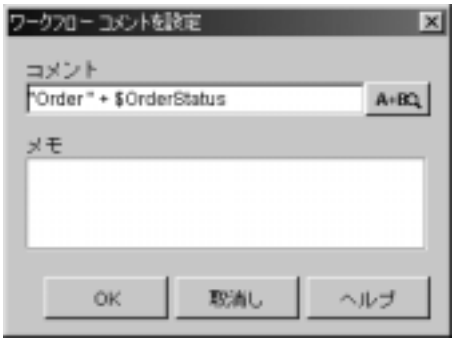
ワークフローのコメントを設定する

OrderStatus 変数を設定しておけば、ビルドする式でこの変数を利用して、ワークフローのコメントを指定できます。コメントは、実行中のアプリケーションに [Order cancelled] と表示されます。

ワークフロー コメントを設定 アクションの追加と定義の手順は、次のとおりです。

1. [Contact Customer Task Properties] ダイアログ ボックスで、[アクティブ時] タブを選択し、[追加] をクリックして [アクションを追加] ダイアログ ボックスを開きます。
2. [ワークフロー アクション] フォルダを展開し、ワークフロー コメントを設定 を選択して [OK] をクリックします。[ワークフロー コメントを設定] ダイアログ ボックスが表示されます。

図 4-15 [ワークフロー コメントを設定] ダイアログ ボックス



3. [コメント] フィールドで直接入力するか、Expression Builder を使用して式 `"Order" + $OrderStatus` を入力します。
4. [OK] をクリックして、ダイアログ ボックスを終了します。ワークフロー コメントを設定 アクションが、[タスク プロパティ] ダイアログ ボックス内の [アクティブ時] タブに表示されます。

タスクのコメントを設定する

Workflow ユーザに対して次のコメントを表示します。

Please contact John Doe at 408 534 9567

実行時にワークフローから変数の値を取得するため CustomerName と CustomerPhone の 2 つの変数を使用します。サンプルの式は次のようになります。

コード リスト 4-3 Contact Customer タスク : タスクのコメント

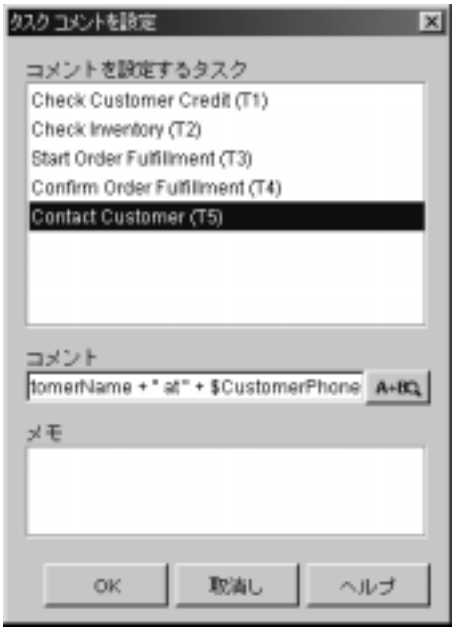
```
"please contact"+ $CustomerName + "at"+ $CustomerPhone
```

プラス記号は文字列リテラルと変数を連結し、スペースは変数の値とリテラル テキストの間の最後のメッセージにスペースが表示されるようにする使い方ができます。

タスク コメントを設定 アクションの追加と定義の手順は、次のとおりです。

1. [Contact Customer Task Properties] ダイアログ ボックスで、[アクティブ時] タブを選択し、[追加] をクリックして [アクションを追加] ダイアログ ボックスを呼び出します。
2. [タスク アクション] フォルダを展開し、タスク コメントを設定 を選択して [OK] をクリックします。[タスク コメントを設定] ダイアログ ボックスが表示されます。

図 4-16 [タスク コメントを設定] ダイアログ ボックス



3. [コメントを設定するタスク] リストから Contact Customer を選択します。
4. [コメント] フィールドに直接入力するか、Expression Builder を起動する [式] ボタンをクリックしてコードリスト 4-3 にあるテキストを入力します。
5. [OK] をクリックし、[タスク コメントを設定] ダイアログ ボックスを終了します。タスク コメントを設定 アクションが [タスク プロパティ] ダイアログ ボックス内の [アクティブ時] タブに表示されます。

タスクをロールに割り当てる

タスクはユーザ名で識別される*ユーザ*に割り当てるほか、関連付けられたすべてのユーザが実行できる*ロール*に割り当てることもできます。

また、タスクを*ロール内のユーザ*に割り当てることもできます。このアクションは、ロール名に関連付けられたユーザのうち、実行時に割り当てられたタスクが最も少ないユーザにタスクを割り当てます。ただし、このサンプルでは限られたユーザとタスクしか使用しないため、タスクをロール名に直接割り当てます。

ここでは、CustomerService ロールにタスクを割り当てます。タスクを割り当てられたユーザ（この場合は *ユーザ* admin のみ）は、タスク コメントを見て、タスクの実行前にそのコメントに対して応答します。

タスクにロールを割り当て アクションの追加と定義の手順は、次のとおりです。

1. [Contact Customer タスク プロパティ] ダイアログ ボックスで、[アクティブ時] タブを選択し、[追加] をクリックして [アクションを追加] ダイアログ ボックスを呼び出します。
2. [タスク アクション] フォルダを展開し、[タスクにロールを割り当て] を選択して [OK] をクリックします。[タスクにロールを割り当て] ダイアログ ボックスが表示されます。

図 4-17 [タスクにロールを割り当て] ダイアログ ボックス

3. [割り当てるタスク] リストから Contact Customer を選択します。
4. [ロール] ドロップダウン リストから CustomerService を選択します。
5. [OK] をクリックして、ダイアログ ボックスを終了します。タスクにロールを割り当てが [タスク プロパティ] ダイアログ ボックス内の [アクティブ時] タブに表示されます。

タスク終了をマークする

このタスクは 1 人のエンド ユーザに割り当てられるタスクなので、タスクに完了マークを付ける アクションを [タスク プロパティ] ダイアログ ボックスの [実行時] タブに追加します。タスクに完了マークを付ける アクションの割り当てがないと、フローはこのノードから先に進めません。

タスクに完了マークを付ける アクションの追加と定義の手順は、次のとおりです。

1. [Contact Customer Task Properties] ダイアログ ボックスで [実行時] タブを選択し、[追加] をクリックして [アクションを追加] ダイアログ ボックスを呼び出します。
2. [タスク アクション] フォルダを展開し [タスクに完了マークを付ける] を選択して [OK] をクリックし、[タスクに完了マークを付ける] ダイアログ ボックスを開きます。
3. [完了マークを付けるタスク] リストから [Contact Customer] を選択し、[OK] をクリックします。タスクに完了マークを付ける が [タスク プロパティ] ダイアログ ボックス内の [実行時] タブに表示されます。
4. [OK] をクリックし、[タスク プロパティ] ダイアログ ボックスを終了します。

ビジネス オペレーションの作成と実行 : Check Inventory タスクの定義

信用チェック分岐で true と評価されると、フローは Inventory Check タスクに進みます。このタスクは、`POBean` EJB 上の `checkInventory()` メソッドを呼び出し、受注品の在庫をチェックし、その結果をワークフローに返します。`checkInventory()` メソッドは、1 個のパラメータ、すなわち商品 ID を受け取り、在庫数を返します。

注意 : `POBean.jar` EJB の詳細は、『*BEA WebLogic Integration Javadoc*』を参照してください。

EJB メソッドを呼び出すためには、次の 2 つを実行する必要があります。

- そのメソッドのビジネス オペレーションの作成
- ビジネス オペレーションを呼び出すノードへの ビジネス オペレーションを実行 アクションの追加と定義

checkInventory() メソッドは値、つまり該当する商品の在庫数を返すので、その結果を格納するワークフロー変数の作成も必要です。

ただし、EJB 上のメソッドを呼び出す前に、まず WebLogic Integration サーバ上に EJB のインスタンスを作成する Bean の create() メソッドを呼び出す必要があります。create() メソッドはまた、EJB インスタンスへの参照も返します。この参照を、*インスタンス*変数と呼ばれるワークフロー変数に格納することによって、EJB 上で複数のメソッドを呼び出せるようになります。

したがって、ここで使用する Check Inventory タスクを定義するには、次の作業が必要です。

- create() メソッドのビジネス オペレーションの作成
- checkInventory() メソッドのビジネス オペレーションの作成
- Perform Business Operation Create OrderBean アクションの追加と定義、および結果を格納する変数の作成
- Perform Business Operation Check Inventory アクションの追加と定義、および結果を格納する変数の作成
- タスクに完了マークを付ける アクションの追加と定義

注意： WebLogic Integration にバンドルされているサンプル プラグインには、プラグイン アクションの Check Inventory が含まれており、このアクションを使用すると、上記の最初の 4 つの手順を 1 つの定義済みのアクションで実行できます。このアクションは、対応する POBean メソッドを呼び出すカスタム コードを作成し、ワークフローとの間でパラメータの受け渡しを行います。一方、入力と出力のパラメータとして機能するワークフロー変数だけは、指定する必要があります。このアクションのダイアログ ボックスを表示するには、[アクション] ダイアログ ボックスで、[Sample Actions] フォルダを展開し [Check inventory for an available item] をダブルクリックします。このアクションとその他のサンプル プラグインの詳細は、『*WebLogic Integration BPM プラグイン プログラミング ガイド*』を参照してください。

ビジネス オペレーションを作成する

ビジネス オペレーションを作成する場合、汎用的なコンフィグレーション、つまり、あらゆるオーガニゼーションのすべてのワークフローで使用可能なコンフィグレーションを行います。ビジネス オペレーションは、基本的に Java クラスや EJB 上のメソッド呼び出しに便利なエリアスです。メソッドが受け取るパラメータに説明的な名前を付けることもできます。そうすれば、ほかのユーザが同じビジネス オペレーションを呼び出そうとする場合に、メソッドに入力が必要なパラメータの内容、およびパラメータの順序を直ちに知ることができます。

Create OrderBean ビジネス オペレーションを作成する必要はありません。ワークフロー オブジェクトのインポート : チュートリアル パッケージ ファイルのインポート」のインポート手順によってインポートしてあります（Order Fulfillment ワークフローもこのビジネス オペレーションを使用しますが、インポート時にそのワークフロー内で未解決の参照が生じないようにしました）。Create OrderBean ビジネス オペレーションは表示だけにとどめます。

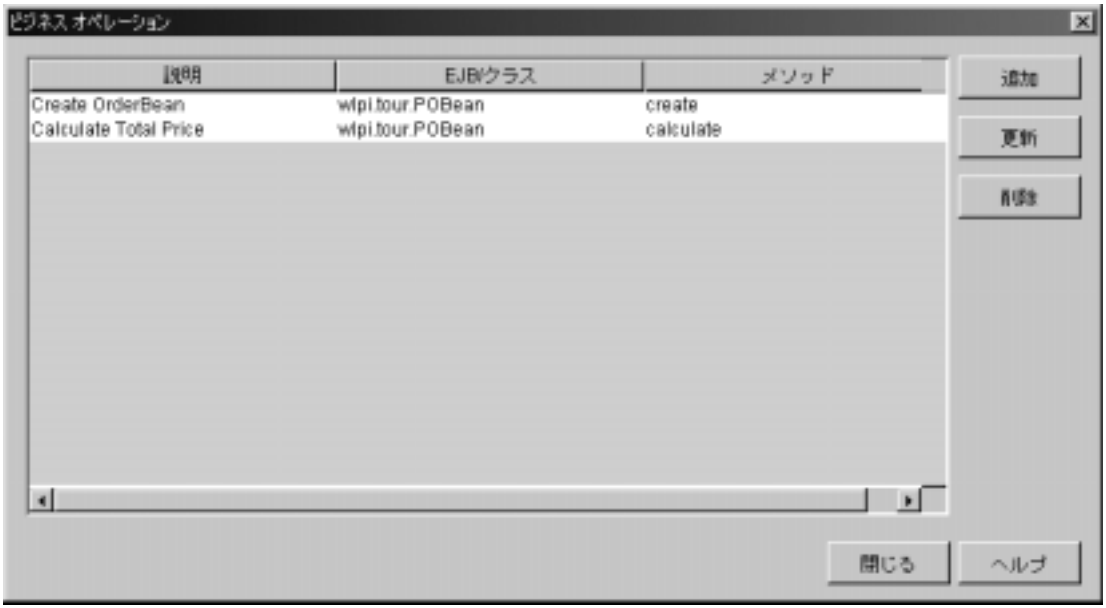
ただし、`POBeans` EJB 上で `checkInventory()` メソッドを呼び出す Check Inventory ビジネス オペレーションを作成し、この Check Inventory タスクで起動できるようにします。

Create OrderBean ビジネス オペレーションを表示する

Create OrderBean ビジネス オペレーションの定義を表示する手順は、次のとおりです。

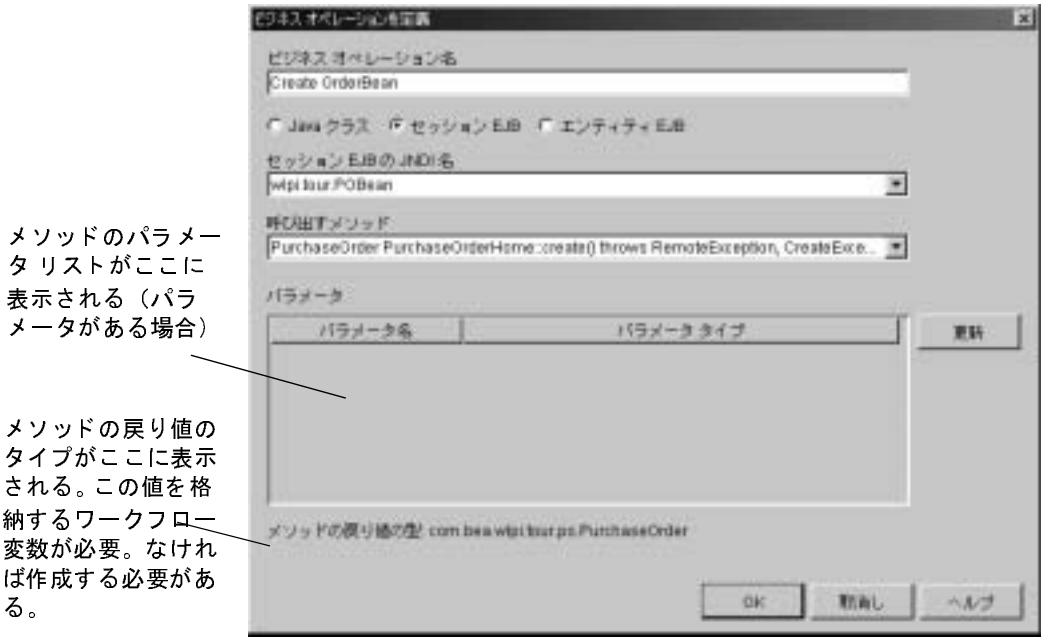
1. [コンフィグレーション | ビジネス オペレーション] を選択します。[ビジネス オペレーション] ダイアログ ボックスが現れ、すでにその中に 2-7 ページの「ワークフロー オブジェクトのインポート : チュートリアル パッケージ ファイルのインポート」でインポートした 2 つのビジネス オペレーション Create OrderBean と Calculate Total Price が表示されています。

図 4-18 [ビジネス オペレーション] ダイアログ ボックス



2. ビジネス オペレーションのリストで、Create OrderBean の説明をダブルクリックします。[ビジネス オペレーションを定義] ダイアログ ボックスが表示されます。

図 4-19 Create OrderBean : [ビジネス オペレーションを定義] ダイアログ ボックス



ダイアログ ボックスは、`create()` メソッドにパラメータがなく、その戻りタイプが POBean EJB のリモート インタフェースへの参照であることを示しています。

3. [取消し] をクリックして、[ビジネス オペレーションを定義] ダイアログ ボックスを終了します。

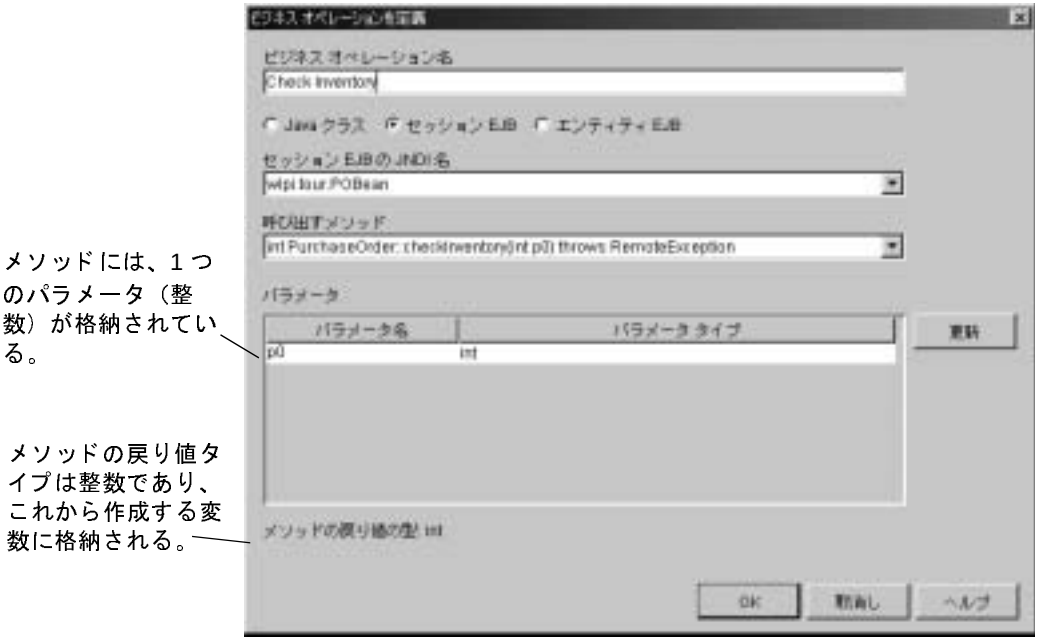
Check Inventory ビジネス オペレーションを作成する

Check Inventory ビジネス オペレーションを作成する手順は、次のとおりです。

1. [ビジネス オペレーション] ダイアログ ボックスで、[追加] ボタンをクリックします。[ビジネス オペレーションを定義] ダイアログ ボックスが表示されます。

2. [セッション EJB] オプションを選択します。現在デプロイされている Session EJB はすべて [セッション EJB の JNDI 名] のドロップダウン リストから選択できるようになりました。
3. [セッション EJB の JNDI 名] ドロップダウン リストから、`wlpi.tour.POBean` を選択します。この EJB のメソッドが、[呼び出すメソッド] ドロップダウン リストから選択できるようになりました。
4. [呼び出すメソッド] ドロップダウン リストから、[`int checkInventory(int p0)`] を選択します。パラメータと戻りタイプがダイアログ ボックスに表示されます。

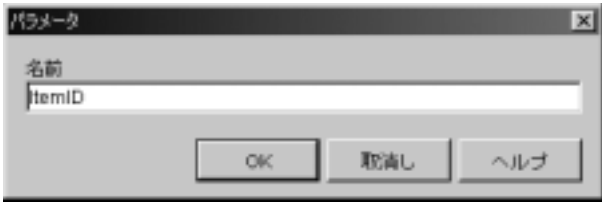
図 4-20 Check Inventory : [ビジネス オペレーションを定義] ダイアログ ボックス



5. [ビジネス オペレーション名] フィールドに、Check Inventory と入力します。

6. （省略可能）[パラメータ] リスト内で、パラメータを 1 つダブルクリックして、そのパラメータの名前を変更します。[パラメータ] ダイアログ ボックスが表示されます。

図 4-21 [パラメータ] ダイアログ ボックス



7. このメソッドはパラメータとして商品 ID を受け取るため、[名前] フィールドに ItemID と入力します。

   注意 : ビジネス オペレーションのパラメータ名にはスペースを入れることはできません。

8. [OK] をクリックします。[ビジネス オペレーションを定義] ダイアログ ボックスの [パラメータ] リストに、新しいパラメータ名が表示されます。

9. [OK] をクリックして [ビジネス オペレーションを定義] ダイアログ ボックスを終了します。新しいビジネス オペレーションが、[ビジネス オペレーション] ダイアログ ボックスに表示されます。

10. [閉じる] をクリックして [ビジネス オペレーション] ダイアログ ボックスを終了します。

ビジネス オペレーションを実行する

必要なメソッドを OrderBean から呼び出すために必要な、ビジネス オペレーションの作成は終わりました。次に、それらをタスク内から開始するアクションを定義する必要があります。

ビジネス オペレーションを開始するアクションを定義するには、以下の変数とパラメータを指定する必要があります。

- 入力パラメータとしてメソッドに渡すワークフロー変数
- メソッドが返した結果を格納するワークフロー変数

- create() メソッドが返す EJB の参照であるインスタンス変数

2 つのビジネス オペレーションに必要なパラメータの関係を次の表にまとめて示します。

表 4-7 Create OrderBean および Check Inventory ビジネス オペレーション : [パラメータ]

ビジネス オペレーション	インスタンス変数	入力パラメータ	結果変数
Create OrderBean	なし	なし	OrderBeanReference
Check Inventory	OrderBeanReference	ItemID	Inventory

このビジネス オペレーションで使用できるようにするには、OrderBeanReference と Inventory 変数の作成が必要です。ビジネス オペレーションで使用する変数は、[ビジネス オペレーションを実行] アクション ダイアログ ボックスで作成できるので、以下の節で、変数の作成手順をアクションの追加と定義の手順と併せて説明します。

Create OrderBean ビジネス オペレーションを実行する

アクションの追加と定義を行う場合は、次の表の変数も作成します。

表 4-8 Create OrderBean ビジネス オペレーション : 結果変数

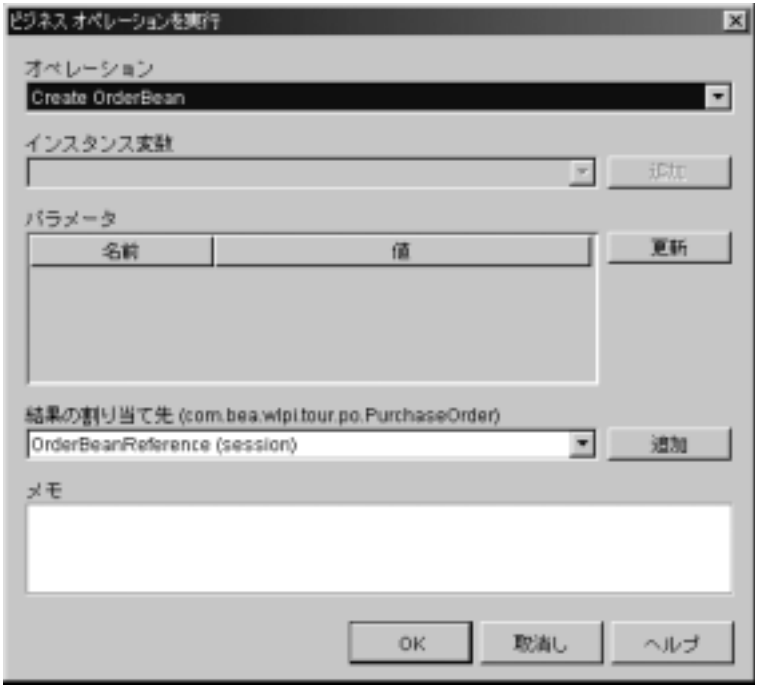
変数	データ型	パラメータ タイプ	ワークフローでの使い方
OrderBeanReference	セッション EJB	なし。	POBean セッション EJB のインスタンスの参照を格納

Create OrderBean の ビジネス オペレーションを実行 アクションの追加と定義の手順は、次のとおりです。

1. [ワークフロー設計] ウィンドウ内で、Check Inventory タスクをダブルクリックして [タスク プロパティ] ダイアログ ボックスを表示します。
2. [アクティブ時] タブを選択し、[追加] ボタンをクリックして [アクションを追加] ダイアログ ボックスを表示します。

3. [統合アクション] フォルダを展開し [ビジネス オペレーションを実行] アクションを選択して [OK] をクリックします。[ビジネス オペレーションを実行] ダイアログ ボックスが表示されます。
4. [オペレーション] ドロップダウン リストから、[Create OrderBean] を選択します。
5. [結果の割り当て先] フィールドの隣の [追加] ボタンをクリックします。[変数プロパティ] ダイアログ ボックスが表示されます。
6. [名前] フィールドに、`OrderBeanReference` と入力します。
7. [タイプ] ドロップダウン リストから、[セッション EJB] を選択します。
8. [入力] チェック ボックスをチェックします (この変数は Order Fulfillment ワークフローでも使用します)。
9. [OK] をクリックします。[結果の割り当て先] フィールドに、変数名とタイプが表示されます。

図 4-22 Create OrderBean : [ビジネス オペレーションを実行] ダイアログ ボックス



10. [OK] をクリックして、ダイアログ ボックスを終了します。Perform Business Operation Create OrderBean アクションが、[タスク プロパティ] ダイアログ ボックス内の [アクティブ時] タブに表示されます。

Check Inventory ビジネス オペレーションを実行する

アクションの追加と定義を行う場合は、次の表の変数も作成します。

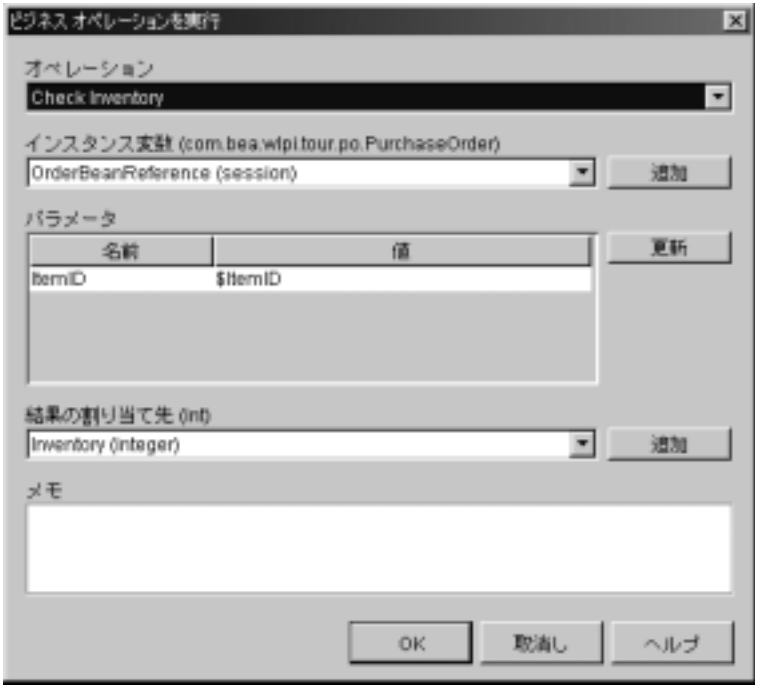
表 4-9 Check Inventory ビジネス オペレーション の 結果変数

変数	データ型	パラメータ タイプ	ワークフローでの使い方
Inventory	Integer	なし。	在庫チェック結果の格納

Check Inventory の ビジネス オペレーションを実行 アクションの追加と定義の手順は、次のとおりです。

1. [Check Inventory タスク プロパティ] ダイアログ ボックスで [アクティブ時] タブを選択し、[追加] をクリックして [アクションを追加] ダイアログ ボックスを表示します。
2. [統合アクション] フォルダを展開し [ビジネス オペレーションを実行] アクションを選択して [OK] をクリックします。[ビジネス オペレーションを実行] ダイアログ ボックスが表示されます。
3. [オペレーション] ドロップダウン リストから [Check Inventory] を選択します。[インスタンス変数] フィールドに、自動的に OrderBeanReference 変数が組み込まれます。
4. [パラメータ] リストで [ItemID] パラ メータをダブルクリックします。[Expression Builder] が表示されます。
5. [変数] オプションを選択し、リスト内の [ItemID] をダブルクリックして [式] フィールドに配置します。
6. [OK] をクリックします。ItemID 変数が [パラ メータ] リストの [値] セクションに表示されます。
7. [結果の割り当て先] フィールドの隣の [追加] ボタンをクリックします。[変数プロパティ] ダイアログ ボックスが表示されます。
8. [名前] フィールドに、`Inventory` と入力します。
9. [タイプ] ドロップダウン リストから [Integer] を選択します。
10. [OK] をクリックします。[結果の割り当て先] フィールドに、変数名とタイプが表示されます。

図 4-23 Check Inventory : [ビジネス オペレーションを実行] ダイアログ ボックス



11. [OK] をクリックして、ダイアログ ボックスを終了します。Perform Business Operation Check Inventory アクションが、[タスク プロパティ] ダイアログ ボックス内の [アクティブ時] タブに表示されます。

タスク終了をマークする

このタスクには、タスクを*実行*するアクション（たとえば、実行を担当する Worklist ユーザへのタスクの割り当て）が含まれていないため、タスクに完了マークを付ける アクションを [タスク プロパティ] ダイアログ ボックスの [実行時] タブではなく [アクティブ時] タブに追加する必要があります。

タスクに完了マークを付ける アクションの追加と定義の手順は、次のとおりです。

1. [タスク プロパティ] ダイアログ ボックスで、[アクティブ時] タブを選択し、[追加] をクリックして [アクションを追加] ダイアログ ボックスを開きます。
2. [タスク アクション] フォルダを展開し [タスクに完了マークを付ける] を選択して [OK] をクリックし、[タスクに完了マークを付ける] ダイアログ ボックスを開きます。
3. [完了マークを付けるタスク] リストから [Check Inventory] を選択し、[OK] をクリックします。タスクに完了マークを付ける アクションが [タスク プロパティ] ダイアログ ボックス内の [アクティブ時] タブに表示されます。
4. [OK] をクリックし、[タスク プロパティ] ダイアログ ボックスを終了します。

不等価のテスト：Check Inventory 分岐の定義

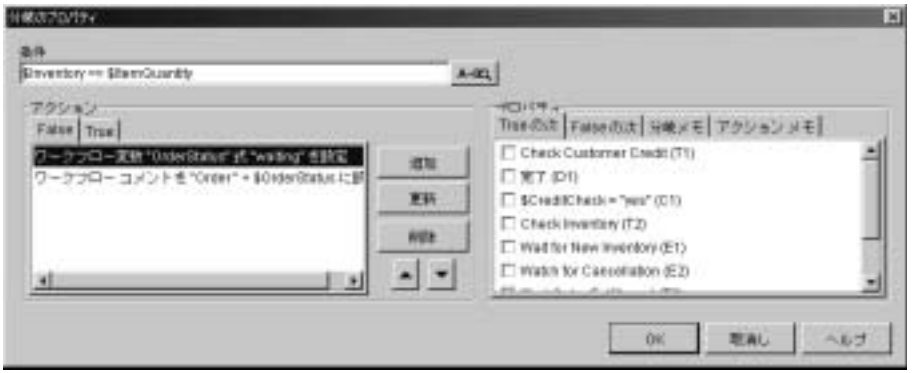
このワークフローの 2 番目の分岐では、Check Inventory ビジネス オペレーションから返された在庫値と顧客の受注品の数量（ワークフロー開始時に `Neworder` XML ドキュメントで指定され、ItemQuantity 変数に格納された数量）とを比較して、在庫が十分かどうかをチェックします。

在庫が十分かどうかチェックするために、2 つの変数を対象とする不等価テストを作成します。変数 Inventory の値が、変数 ItemQuantity の値以上であるかをテストするわけです。条件によって true と分岐された場合、フローは Start Order Fulfillment タスクに進み、false と分岐された場合、受注は "waiting" に設定され、フローは Wait for Inventory タスクに進みます。

C2 分岐ノードを定義する手順は、次のとおりです。

1. [ワークフロー設計] ウィンドウ内で、[C2] 分岐シェイプをダブルクリックして [分岐のプロパティ] ダイアログ ボックスを表示します。

図 4-24 Check Inventory 分岐



2. [条件] フィールドに、`$Inventory>=$ItemQuantity` と入力します。
3. [False] タブを選択し、[追加] ボタンをクリックして [アクションを追加] ダイアログ ボックスを開きます。
4. [ワークフロー アクション] フォルダを展開し [ワークフロー変数を設定] を選択して [OK] をクリックし、[ワークフロー変数を設定] ダイアログ ボックスを表示します。
5. [変数] ドロップダウン リストから [OrderStatus] を選択します。
6. [式] オプションを選択し、フィールド内に `"waiting"` と入力します。
7. [OK] をクリックして [ワークフロー変数を設定] を終了します。ワークフロー変数を設定 アクションが [False] タブに表示されます。
8. [追加] ボタンをクリックして [アクションを追加] ダイアログ ボックスを表示します。
9. [ワークフロー アクション] フォルダを展開し、ワークフロー コメントを設定 を選択して [OK] をクリックし、[ワークフロー コメントを設定] ダイアログ ボックスを表示します。
10. [コメント] フィールドに、式 `"Order"+ $OrderStatus` を入力します。
11. [OK] をクリックし、[ワークフロー コメントを設定] ダイアログ ボックスを終了します。ワークフロー コメントを設定 アクションが [False] タブに表示されます。

12. [OK] をクリックし、[分岐のプロパティ] ダイアログ ボックスを終了します。

イベントの作成 : Wait for New Inventory イベントの定義

在庫チェック分岐が、false となった場合、フローは [Wait for New Inventory] イベントに進みます。このイベントは、たとえば、該当する XML メッセージが在庫報告アプリケーションから JMS キューに到着することによってトリガされます。ノードはメイン フローに埋め込まれているので、フローは先に進まず、ノードがトリガされるまで待機状態に置かれます。ノードがトリガされると、フローは Inventory Check タスクにループバックし、現在の在庫が受注の処理に十分かどうかをチェックします。

実際の XML ドキュメントを使用していないので、このイベントがサンプル ワークフロー内で実際にトリガされることはありません。しかし、次のドキュメントのような仮の XML ドキュメントをイメージすることができます。

コード リスト 4-4 Newinventory XML ドキュメント

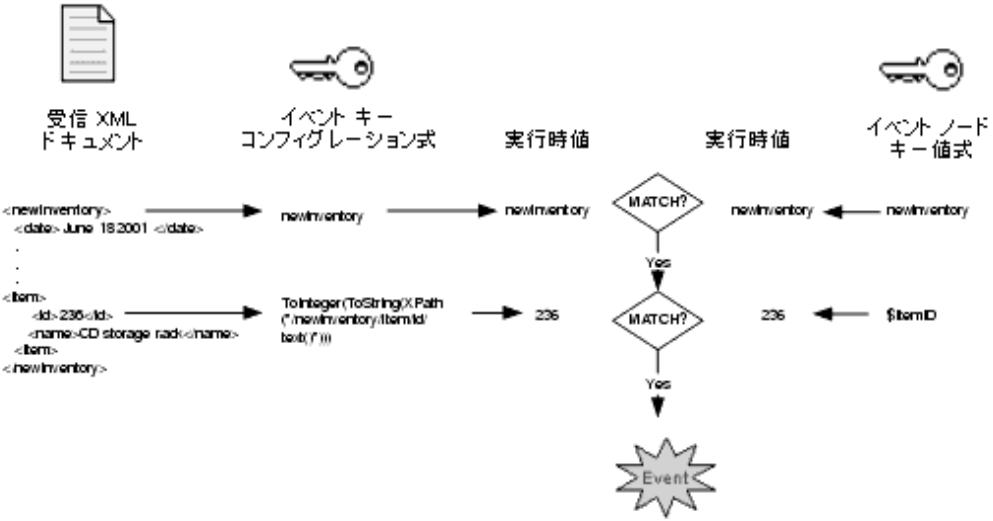
```
<newinventory>
   <date> June 18 2001 </date>
   .
   .
   .
   <item>
      <id>236</id>
      <name>CD storage rack</name>
   <item>
</newinventory>
```

実行時に、該当する商品の情報を含む <newinventory> ドキュメント（このサンプルの最初の XML ドキュメントでは CD 収納ラック）のインスタンスによってのみイベントがトリガされるようにするため、イベント プロセッサの監視対

象の要素（この場合は商品 ID）をメッセージドキュメントで指定するイベント キーを定義します。イベント キーの式を、XML ドキュメント内の対象要素から値を抽出する XPath 文として指定します。

次に、イベント プロセッサが <newiventory> ドキュメントを参照するイベントを含むすべてのワークフロー インスタンスを検索しなくても済むように、イベント ノードそのものが、イベント キーで抽出された値と一致するようにキー値の式を定義します。2 つの値が一致すると、イベントがトリガされます。そのメカニズムを次の図に示します。

図 4-25 受信 XML ドキュメント、イベント キーの式、キー値の式の関係



コードリスト 4-4 の例では、当初の商品 ID が 236 だったのでイベントがトリガされます。

受信 XML ドキュメントの要素値、イベント キーのコンフィグレーション式、イベント ノードのキー値式が、[Wait for Inventory] ノード用にどのように互いに対応するかを次の表に示します。

表 4-10 Wait for Inventory イベント：イベント キーとイベント ノード式の関係

要素	XML ドキュメント	イベント キー コンフィグレーション	イベント ノード
ルート要素 / 文書型	<newinventory>	newinventory	newinventory
イベント キー式 / キー値式	<item> <id>	ToInteger(ToString(XPath("/newinventory/item/id/text()")))	$ItemID

イベント キーのコンフィグレーション式では、開始ノードまたはイベント ノード内のキー値式と同じデータ型で評価する必要があります。XPath 式は*ノード リスト*のタイプを返すため、通常、正しいデータ型を返すには、ワークフロー式の言語で記述される型キャストを実行する関数を使用する必要があります。この例では、変数が整数で保存されているため、最初にノード リストを文字列に変換し、次に文字列を整数に変換する必要があります。型キャストその他のワークフロー関数の詳細は、『*WebLogic Integration Studio ユーザーズ ガイド*』の「ワークフロー式の使用」の「関数を使用する」を参照してください。

ここで使用するイベントを作成するには、次の作業を行う必要があります。

- イベント キーのコンフィグレーション
- イベントの定義

イベント キーをコンフィグレーションする

イベント キーをコンフィグレーションする場合は、XML ドキュメント、および抽出対象の特定の要素の値を探すためにドキュメントを解析する式の文書型またはルート要素を指定します。イベント キーをコンフィグレーションすると、すべてのワークフローのテンプレートで使用できます。

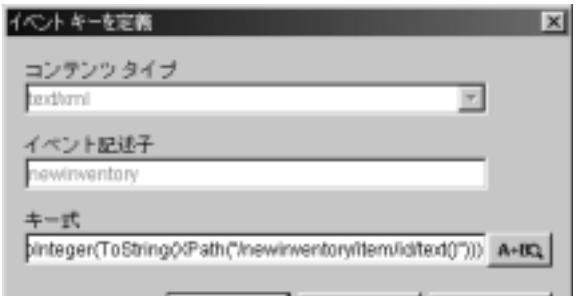
[newinventory] イベント キーを定義する手順は、次のとおりです。

1. [コンフィグレーションを選択 | イベント] を選択します。第 2 章「WebLogic Integration Studio 入門」でインポートした [cancelledorder] イベント キーが表示されている、[イベント キー] ダイアログ ボックスが表示されます。

図 4-26 [イベント キー] ダイアログ ボックス

2. [追加] ボタンをクリックします。[イベント キーを定義] ダイアログ ボックスが表示されます。

図 4-27 [イベント キーを定義] ダイアログ ボックス



3. [イベント記述子] フィールドに、`newinventory` と入力します。
4. [式] フィールドに、直接入力するか、Expression Builder を起動する [式] ボタンをクリックして、表 4-10 の XPath 式を入力します。

5. [OK] をクリックします。[イベント キー] ダイアログ ボックスに、新しいイベント キーが表示されます。

6. [閉じる] ボタンをクリックして、[イベント キー] ダイアログ ボックスを終了します。

イベントを定義する

イベント ノードに、受信 XML ドキュメントのルート要素と、XML ドキュメントにより提示される値と一致させたいキー値の式を指定します。各イベント ノードで使用するキー値の式は、それぞれユニークである必要があります。条件を追加して受信ドキュメントをフィルタ処理できますが、このサンプルでは使用しません。

注意 : イベント キー機能はパフォーマンスがよいため、常にイベント条件よりも優先してイベント キーを使用する必要があります。イベントをトリガする XML ドキュメントのインスタンスをより厳密に制限したい場合、かならずイベント キーに*追加*するかたちで条件を使用します。

Wait for Inventory イベントを定義する手順は、次のとおりです。

1. [ワークフロー設計] ウィンドウで、[Wait for Inventory] イベント ノードをダブルクリックします。[イベントのプロパティ] ダイアログ ボックスが表示されます。

図 4-28 Wait for Inventory イベントの [イベントのプロパティ] ダイアログ ボックス

2. [タイプ] ドロップダウン リストから [XML イベント] を選択します。
3. [ドキュメント タイプ / ルート要素] フィールドに `newinventory` と入力します。
4. [キー値の式] フィールドに、直接入力するか、Expression Builder を起動する [式] ボタンをクリックして、`$ItemID` と入力します。
5. [OK] をクリックし、[イベントのプロパティ] ダイアログ ボックスを終了します。

サブワークフローの呼び出し：Start Order Fulfillment タスクの定義

在庫チェックの分岐が true となった場合、ワークフローは Start Order Fulfillment タスクに進み、Order Fulfillment ワークフローを呼び出します。サブワークフローが実行を終了すると、コントロールは Start order Fulfillment タスクに戻り、そのタスクに完了マークが付くと、フローはメイン ワークフローの次のノード、すなわち、Confirm Order Fulfillment タスクに進みます。

サブフローを呼び出す場合は、メイン ワークフローの実行中に収集または設定された値を、サブフローの *input* 変数に渡す必要があります。たとえば、出荷タスクに必要な顧客の住所、請求書生成タスクによって呼び出される `POBean calculate()` メソッドに必要なパラメータなどの値を渡します（インポートされた Order Fulfillment ワークフローに定義済みの変数リストは、テンプレート定義を開いてフォルダ ツリーの [変数] フォルダを展開すると表示できます。変数が input、output のどちらに定義されているかを知るには、変数を右クリックし、表示されるポップアップ メニューから [プロパティ] を選択して、変数の [プロパティ] ダイアログ ボックスを表示します）。

また、`calculate()` メソッドが提出する注文合計金額など、サブワークフローから返される値を格納する変数を作成し、指定する必要もあります。

Start Order Fulfillment タスクを定義するには、次の作業が必要です。

- OrderTotalPrice 変数の作成
- ワークフローを設定 アクションの追加と定義
- 入力パラメータの定義
- 結果変数の定義
- タスクに完了マークを付ける コールバック アクションの追加と定義

OrderTotalPrice 変数を作成する

calculate() メソッド（Order Fulfillment ワークフローによって呼び出される）から返される結果を格納するために、次の表で説明する OrderTotalPrice 変数を作成する必要があります。必要に応じて、次の表から値を代入し、4-6 ページの「変数を作成する」で説明する手順を参照してください。

表 4-11 Start Order Fulfillment タスクの変数

変数	データ型	パラメータ タイプ	ワークフローでの使い方
OrderTotalPrice	Double	Input	Order Fulfillment ワークフローで呼び出された calculate() メソッドの結果の格納する。

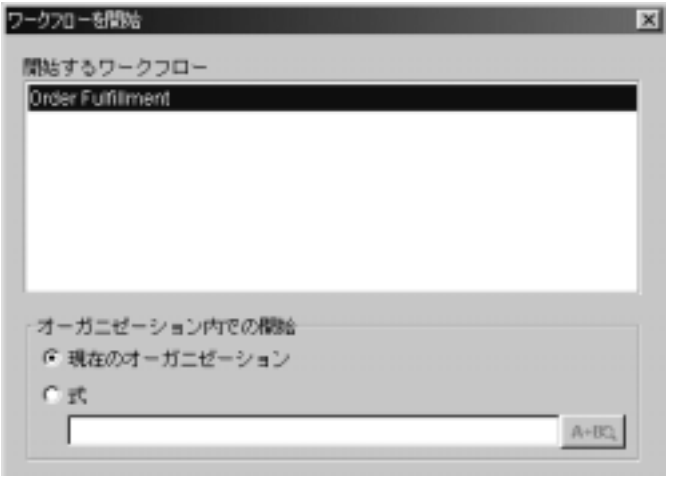
呼び出しワークフローを開始する

サブワークフローを開始するには、サブワークフローの開始ノードが [呼び出し] と定義されており、さらにワークフローがアクティブである必要があります。この例では、2-7 ページの「ワークフロー オブジェクトのインポート：チュートリアル パッケージ ファイルのインポート」で Order Fulfillment ワークフローをインポートしたときに、このワークフローをアクティブにしています。

ワークフローを開始 アクションの追加と定義の手順は、次のとおりです。

1. [ワークフロー設計] ウィンドウ内で、[Start Order Fulfillment] タスク ノードをダブルクリックして [タスク プロパティ] ダイアログ ボックスを表示します。
2. [アクティブ時] タブを選択し、[追加] ボタンをクリックして [アクションを追加] ダイアログ ボックスを表示します。
3. [ワークフロー アクション] フォルダを展開し [ワークフローを開始] を選択して [OK] をクリックします。[ワークフローを開始] ダイアログ ボックスが表示されます。
4. [開始するワークフロー] ウィンドウで [Order Fulfillment] を選択します。

図 4-29 [ワークフローを開始] ダイアログ ボックス



5. [オーガニゼーション内での開始] 領域で、[現在のオーガニゼーション] を選択します。

入力パラメータを定義する

[ワークフローを開始] ダイアログ ボックスで呼び出すサブワークフローを選択する場合、サブワークフロー内で input と定義されている変数がダイアログ ボックス下部の [パラメータ] タブに表示されます。

図 4-30 Start Order Fulfillment ワークフロー：入力パラメータ

ここで、パラメータの値を設定する必要があります。値を定義する式を使用できますが、このサンプルでは現在の変数の値を単純に渡すだけなので、変数名をパラメータ式として指定できます。

Start Order Fulfillment ワークフローの入力パラメータを定義する手順は、次のとおりです。

1. [ワークフローを開始] ダイアログ ボックス下部の [パラメータ] タブを選択します。
2. リスト内のパラメータをダブルクリックします。[パラメータを設定] ダイアログ ボックスが表示されます。

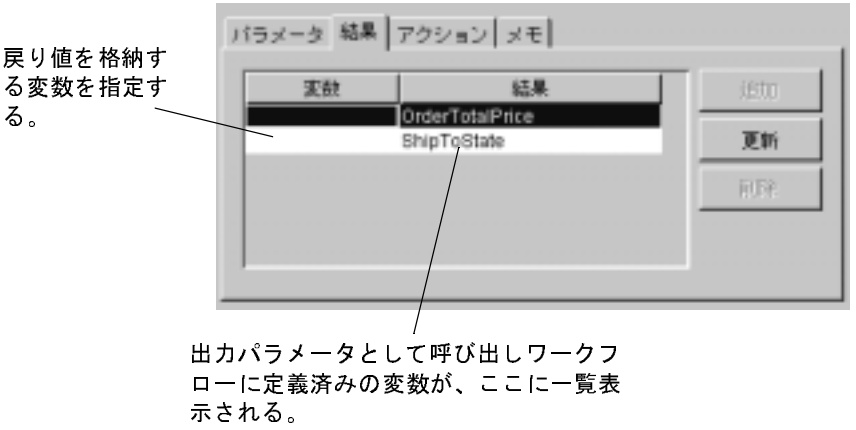
図 4-31 [パラメータを設定] ダイアログ ボックス

3. [式] フィールドで、パラメータに対応する変数名を先頭にドル記号を付けて入力し、[OK] をクリックします。
4. 表の変数全部に対して、手順 1 から 3 までを繰り返します。
5. 終了すると、次のようにリストが完成します。

結果変数を定義する

[ワークフローを開始] ダイアログ ボックスで、呼び出すワークフローを選択する場合、サブワークフロー内で output と定義されている変数が、ダイアログ ボックス下部の [結果] タブに表示されます。

図 4-32 Start Order Fulfillment ワークフロー : [結果]



ここで、サブワークフローから返される結果が対応する変数に確実に格納されるようにします。このサンプルでは、すべての変数名が同じなので、それぞれの結果パラメータに対応する変数名だけを指定します。

Start Order Fulfillment ワークフローの結果パラメータを定義する手順は、次のとおりです。

1. [ワークフローを開始] ダイアログ ボックス下部の [結果] タブを選択します。
2. 結果リスト内の項目をダブルクリックします。[結果から変数を設定] ダイアログ ボックスが表示されます。

図 4-33 [結果から変数を設定] ダイアログ ボックス

3. [変数] ドロップダウン リストから、結果の割り当て先とする変数を選択し、[OK] をクリックします。
4. 両方の結果に対して、手順 1 から 3 を繰り返し行います。
5. 終了すると、次のようにリストが完成します。

タスク終了をマークする

メイン ワークフローが、サブワークフローの終了を待ってから次に進むようにするため、ワークフローを開始 アクション内にサブアクションとして タスクに完了マークを付ける を追加します。

タスクに完了マークを付ける アクションの追加と定義の手順は、次のとおりです。

1. [ワークフローを開始] ダイアログ ボックスの下部で、[アクション] タブを選択し、[追加] をクリックして [アクションを追加] ダイアログ ボックスを開きます。
2. [タスク アクション] フォルダを展開し [タスクに完了マークを付ける] を選択して [OK] をクリックし、[タスクに完了マークを付ける] ダイアログ ボックスを開きます。
3. [完了マークを付けるタスク] リストから、Start Order Fulfillment を選択し、[OK] をクリックします。タスクに完了マークを付ける アクションが、[ワークフローを開始] ダイアログ ボックスの [アクション] タブに表示されます。

電子メール メッセージの送信：Confirm Order Fulfillment タスクの定義

Order Fulfillment が実行された後は、Watch for Cancellation イベントがトリガされず、ワークフローを終了できないように設定します。Confirm Order Fulfillment タスクは Watch for Cancellation イベントをキャンセルし、受注ステータスを完了に設定し、その旨を記したワークフロー コメントを追加し、受注の完了を確認する 電子メールを顧客への送信（省略可能）を行います。

このタスクをセットアップする作業は、次のとおりです。

- Cancell Event アクションの追加と定義
- ワークフロー変数を設定 アクションの追加と定義
- ワークフロー コメントを設定 アクションの追加と定義
- 電子メール送信 アクションの追加と定義
- タスクに完了マークを付ける アクションの追加と定義

イベントをキャンセルする

Order Fulfillment ワークフローが実行されて受注品が出荷された後は、受注は取り消せなくなります。Watch for Cancellation イベントの受信メッセージに対するリスンを停止するには、イベントの取消し アクションを Confirm Order Fulfillment タスクに追加します。

イベントの取消し アクションの追加の手順は、次のとおりです。

1. [ワークフロー設計] ウィンドウ内で、Confirm Order Fulfillment タスクをダブルクリックして [タスク プロパティ] ダイアログ ボックスを表示します。
2. [アクティブ時] タブを選択し、[追加] ボタンをクリックして [アクションを追加] ダイアログ ボックスを開きます。続いて、[その他のアクション] フォルダを展開し、ワークフロー イベントを取消し を選択して [OK] をクリックします。[ワークフロー イベントを取消し] ダイアログ ボックスが表示されます。

図 4-34 [ワークフロー イベントを取消し] ダイアログ ボックス

3. [取り消すイベント] リストから [Watch for Cancellation] を選択して [OK] をクリックします。ワークフロー イベントを取消し アクションが、[タスク プロパティ] ダイアログ ボックス内の [アクティブ時] タブに表示されます。

ワークフロー変数を設定する

次の手順では、ワークフロー変数 OrderStatus に complete という値を設定する方法を説明します。

ワークフロー変数を設定 アクションを追加する手順は、次のとおりです。

1. [アクティブ時] タブを選択し、[追加] ボタンをクリックして [アクションを追加] ダイアログ ボックスを開きます。次に、[ワークフロー アクション] フォルダを展開し、ワークフロー変数を設定 を選択し、[OK] をクリックして [ワークフロー変数を設定] ダイアログ ボックスを開きます。
2. [設定する変数] ドロップダウン リストから [OrderStatus] を選択します。
3. [式] オプションを選択し、フィールド内に `"complete"` と入力します。

4. [OK] をクリックします。ワークフロー変数を設定 アクションが、[タスク プロパティ] ダイアログ ボックス内の [アクティブ時] タブに表示されます。

ワークフローのコメントを設定する

OrderStatus 変数を complete に設定すれば、この変数をワークフローのコメント式で使用できます。Order complete に対するワークフロー コメントの設定方法を、以下の手順で説明します。

ワークフロー コメントを設定 アクションの追加と定義の手順は、次のとおりです。

1. [タスク プロパティ] ダイアログ ボックスで、[アクティブ時] タブを選択し、[追加] をクリックして [アクションを追加] ダイアログ ボックスを呼び出します。
2. [ワークフロー アクション] フォルダを展開し、ワークフロー コメントを設定 を選択して [OK] をクリックし、[ワークフロー コメントを設定] ダイアログ ボックスを表示します。
3. [コメント] フィールドで、直接入力するか Expression Builder を使用して、次の式を入力します。

   ```
   "Order "+ $OrderStatus
   ```

4. [OK] をクリックして、ダイアログ ボックスを終了します。ワークフロー コメントを設定 アクションが、[タスク プロパティ] ダイアログ ボックス内の [アクティブ時] タブに表示されます。

電子メール メッセージを送信する

受注を完了するために、受注の完了を確認する 電子メール メッセージを発注元の顧客に対して送信するワークフローを設定できます。

3-15 ページの「内部 XML イベントのポスティング : Neworder XML ドキュメントの編集」で、ユーザの 電子メール アドレスとその他のデータを入力済みの場合は、第 7 章「サンプル ワークフローの実行とモニタ」でワークフローを実行すると 電子メールを受信します。

注意 : `tutorial.jar` ファイルにある Order Processing テンプレート定義には、このアクションは含まれません。以下の文で説明するように、正しいサーバ コンフィグレーションが必要となるためです。

実行時に電子メールを送信しようとする場合、以下の条件が前提となります。

- ユーザの電子メール サーバが、WebLogic Integration のインストール時またはインストール後に正しくコンフィグレーションされていること。インストール後の電子メール サーバのプロパティのセットアップの詳細は、『*WebLogic Integration の起動、停止およびカスタマイズ*』の「WebLogic Integration のカスタマイズ」の中の「メール セッション プロパティのカスタマイズ」を参照してください。
- 有効な電子メール アドレスを設定すること。Order Processing Trigger ワークフローに設定されているデフォルトの電子メール アドレスを編集する必要があります。3-15 ページの「内部 XML イベントのポスティング : Neworder XML ドキュメントの編集」で説明する手順に従って編集してください。

前記の 2 つの条件に当てはまらない場合は、電子メール メッセージを送信 アクションを追加するこの手順を行わないでください。

電子メール送信 アクションの追加を選択した場合は、メッセージのコンテンツとして次に示すメッセージを使用できます。

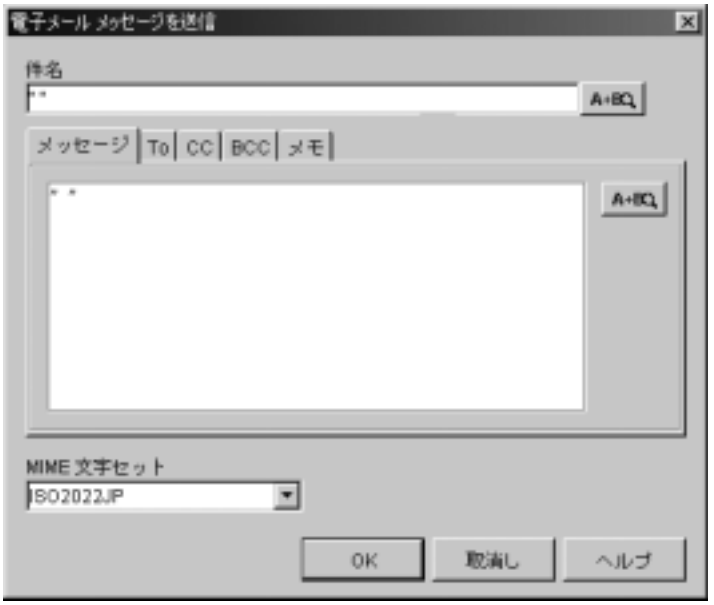
```
Subject: Your order #order number

Your order for item (quantity) has been shipped. The total price
for your order is amount. Thank you for your business.
```

電子メール メッセージを送信 アクションの追加と定義の手順は、次のとおりです。

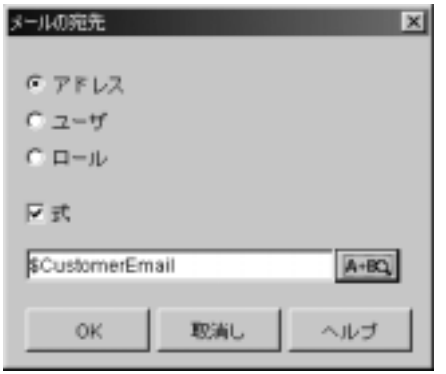
1. [タスク プロパティ] ダイアログ ボックスで、[アクティブ時] タブを選択し、[追加] をクリックして [アクションを追加] ダイアログ ボックスを呼び出します。
2. [その他のアクション] フォルダを展開し [電子メール メッセージを送信] を選択して [OK] をクリックします。[電子メール メッセージを送信] ダイアログ ボックスが表示されます。

図 4-35 [電子メール メッセージを送信] ダイアログ ボックス



3. [件名] フィールドに、`"Your order #" + $OrderID` または任意のメッセージを入力します。
4. [To] タブを選択し、[追加] ボタンをクリックします。[メールの宛先] ダイアログ ボックスが表示されます。

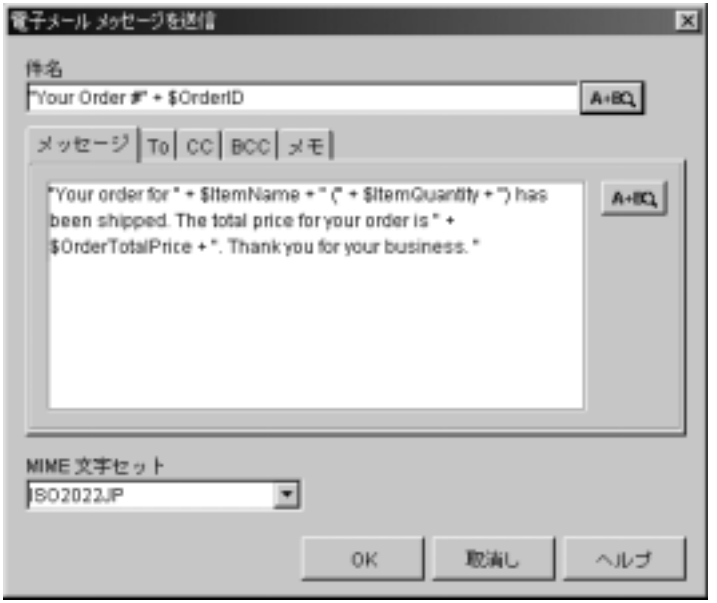
図 4-36 [メールの宛先] ダイアログ ボックス



5. [アドレス] オプションを選択します。
6. [式] チェック ボックスをチェックし、フィールド内に `$CustomerEmail` と入力します。
7. [OK] をクリックします。CustomerEmail 変数が [宛先] リストに表示されます。

8. [メッセージ] タブを選択し、任意の文を入力します。次の図のように入力すると、前に示したメッセージと同じような文になります。

図 4-37 [電子メール メッセージを送信] ダイアログ ボックス : 受注確認メッセージ



9. 完了したら [OK] をクリックします。
10. [電子メール送信] アクションが、[タスク プロパティ] ダイアログ ボックス内の [アクティブ時] タブに表示されます。

タスク終了をマークする

このタスクは、タスクを*実行*するアクションを持たないため、タスクに完了マークを付ける アクションを [タスク プロパティ] ダイアログ ボックスの [アクティブ時] タブに追加する必要があります。

タスクに完了マークを付ける アクションの追加と定義の手順は、次のとおりです。

1. [タスク プロパティ] ダイアログ ボックスで、[アクティブ時] タブを選択し、[追加] をクリックして [アクションを追加] ダイアログ ボックスを開きます。

2. [タスク アクション] フォルダを展開し [タスクに完了マークを付ける] を選択して [OK] をクリックし、[タスクに完了マークを付ける] ダイアログ ボックスを開きます。
3. [完了マークを付けるタスク] リストから [Confirm Order Fulfillment] を選択し、[OK] をクリックします。タスクに完了マークを付ける アクションが [タスク プロパティ] ダイアログ ボックス内の [アクティブ時] タブに表示されます。
4. [OK] をクリックし、[タスク プロパティ] ダイアログ ボックスを終了します。

イベントの作成 : Watch for Cancellation イベントの定義

Order Fulfillment サブワークフローの [Ship Order] タスクが実行されるまでのプロセス全体の任意の時点で、たとえば、該当する XML メッセージが Web 層のアプリケーションから JMS キューに到着することによって、Watch for Cancellation イベントをトリガできます。ノードがフローの外に存在し、開始ノードと完了ノードに直接アタッチされているため、現在実行中のノードとは無関係にフローを中断できます。Order Fulfillment ワークフローの出荷タスクの実行前にノードがトリガされた場合、Order Processing ワークフローは完了ノードに進み終了します。

ここでは実際の XML ドキュメントを使用していないため、サンプル ワークフロー内でこのイベントは実際に開始されません。ただし、次のドキュメントのような仮の XML ドキュメントをイメージできます。

コード リスト 4-5 Cancellorder XML ドキュメント

```
<cancelledorder>
   <date> June 18 2001 </date>
   <order>
      <id>9654</id>
   </order>
   <customer>
      <id>232</id>
      <name>Harold Jones</name>
```

```
    </customer>
.
.
.
</cancelledorder>
```

受信 XML ドキュメント、イベント キーのコンフィグレーション式、イベント ノードのキー値式が、このノードに関して相互に対応する方法を次の表で示します。

表 4-12 Watch for Cancellation イベント：イベント キーとイベント ノード式の関係

要素	XML ドキュメント	イベント キー コンフィグレーション	イベント ノード
ルート要素 / 文書型	`<cancelledorder>`	`cancelledorder`	`cancelledorder`
イベント キー式 / キー値式	`<order>` `  <id>`	`ToInteger(ToString(XPath ("cancelledorder/ order/id/text()")))`	`$OrderID`

Order Fufillment ワークフロー（詳細は、6-1 ページの「Order Fulfillment ワークフローの概要」を参照）にも使用される [cancelledorder] イベント キーは、すでにインポート済みのため、イベント キーを作成する必要はありません。イベントの定義だけが必要です。

イベントを定義する

Watch for Cancellation イベントを定義する手順は、次のとおりです。

1. [ワークフロー設計] ウィンドウで、[Wait for Cancellation] イベント ノードをダブルクリックします。[イベントのプロパティ] ダイアログ ボックスが表示されます。

図 4-38 Watch for Cancellation イベント : [イベントのプロパティ] ダイアログ ボックス

2. [タイプ] ドロップダウン リストから [XML イベント] を選択します。
3. [ドキュメント タイプ / ルート要素] フィールドに `cancelledorder` と入力します。
4. [キー値の式] フィールドに、直接入力するか、Expression Builder を起動する [式] ボタンをクリックして、`$OrderID` を入力します。
5. （省略可能）OrderStatus 変数を次の手順に従って 取消し に設定します。
   a. [変数] タブを選択し、[追加] ボタンをクリックして [ワークフロー変数の割り当て] ダイアログ ボックスを開きます。
   b. [変数] ドロップダウン リストから [OrderStatus] を選択します。

c. [式] フィールドに、`"cancelled"` を入力します。

d. [OK] をクリックし、[ワークフロー変数の割り当て] ダイアログ ボックスを終了します。変数と式が [変数] タブに表示されます。

6. （省略可能）ワークフローのコメントを次の手順に従って Order cancelled に設定します。

a. [アクション] タブを選択し、[追加] ボタンをクリックして [アクションを追加] ダイアログ ボックスを開きます。

b. [ワークフロー アクション] フォルダを展開し、ワークフロー コメントを設定 を選択して [OK] をクリックし、[ワークフロー コメントを設定] ダイアログ ボックスを表示します。

c. [コメント] フィールドに、式 `"Order"+ $OrderStatus` を入力します。

d. [OK] をクリックし、[ワークフロー コメントを設定] ダイアログ ボックスを終了します。ワークフロー コメントを設定 アクションが [アクション] タブに表示されます。

7. [OK] をクリックし、[イベントのプロパティ] ダイアログ ボックスを終了します。

テンプレート定義を保存すると、実行準備は完了です。次の節では、Order Fulfillment ワークフローについて説明します。定義済みのワークフローがインポート済みです。

5　ワークフローの作成

注意： Worklist クライアント アプリケーションは WebLogic Integration リリース 7.0 より非推奨となりました。代替機能に関する詳細については、『*WebLogic Integration リリース ノート*』を参照してください。

チュートリアルのこの節では、Order Processing ワークフローの作成方法を説明します。ワークフローの概要を説明した後で、以下の操作について説明します。

- テンプレートの作成
- テンプレート定義の作成
- ノードのシェイプの配置、名称変更、接続によるワークフローの作成
- ワークフロー ラベルの追加
- Expression Builder による式の設定
- テンプレート定義のアクティブ化
- テンプレート定義の保存

Order Processing ワークフローの設計概要

Order Processing ワークフローは、受注から顧客信用情報のチェックや在庫チェックまでのプロセスを処理します。次に、コントロールを、出荷と請求の処理を行う Order Fulfillment ワークフローに渡します。Order Fulfillment プロセスが終了すると、コントロールは Order Processing ワークフローに戻され、そこで受注処理完了の確認が行われます。

インタフェース ビューのワークフローを高レベルで描写した図を、以下に表示しています。図中で番号が振られている手順については、図の下にある表で説明します。表では、実際のプロセスをワークフロー モデルにマップしています。

図 5-1 Order Processing ワークフロー：インタフェース ビュー

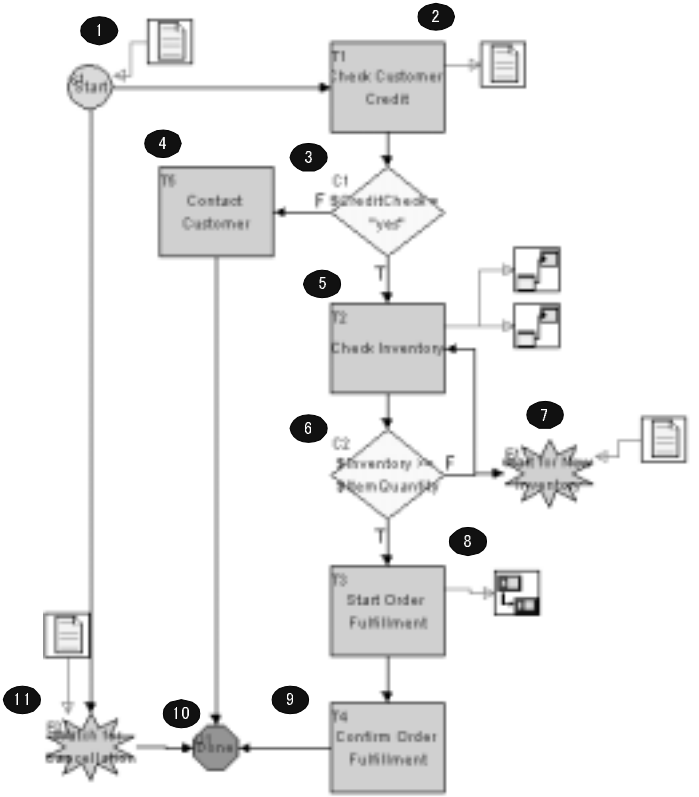


表 5-1 Order Processing ワークフローの概要

プロセス	実装
処理システムによってデータが読み取られ、ID 番号が付けられて受理されます。	1. ワークフローは XML ドキュメントの受信によって開始される。
その受注はカスタマサービスに転送され、顧客の信用情報がチェックされます。	2. XML メッセージが Worklist ユーザに送信され、信用チェックの確認が行われる。
	3. 分岐ノードは、Worklist ユーザの応答がはいかいいえかを評価する。

表 5-1 Order Processing ワークフローの概要 （続き）

プロセス	実装
信用チェックができなかった場合、顧客サービス担当者は顧客に通知して、正しい信用情報を取得する。これ以降のプロセスは手動で行う。	4. 顧客への問い合わせタスクを Worklist ロールに割り当て、ワークフローは終了する。
信用チェックが通ると、商品 ID を基に受注商品の現在の目録がデータベースでチェックされ、出荷可能な商品の数量と受注数量が比較されます。	5. EJB が呼び出され、在庫チェックが行われる。
	6. 分岐ノードによって、在庫が十分かどうかが評価される。
在庫が十分ではない場合は、次の新規入庫の到着までその受注は保留されます。新規入庫の在庫表が受領されると、その在庫が受注に対応可能であると確認できるまで、手順 5 が繰り返されます。	7. イベント ノードは XML ドキュメントを受信するまで待機する。イベントが開始すると、フローは在庫チェック タスク ノードにループ バックする。
在庫が十分な場合は、受注が 2 人の担当者に同時に転送される。担当者の 1 人は出荷を担当し、もう 1 人は受注に対する請求書生成をシステムに指示する。	8. Order Fulfillment サブワークフローが呼び出される。
次に、受注品が出荷されたことがシステムで確認され、顧客に電子メールで通知されます。	9. 電子メールが顧客に送信される。
	10. ワークフローが終了する。
出荷前のトランザクションにおけるどの時点でも、顧客からの通知により受注をキャンセルできます。	11. イベント ノードは XML ドキュメントを受信するまで待機する。

テンプレートの作成

ワークフローのテンプレートを作成するときに、それに関連付けするオーガニゼーションを指定します。

注意： 1 つのテンプレートを複数のオーガニゼーションに関連付ける場合、そのテンプレートに加えたすべての変更は、他のオーガニゼーションがそのテンプレートを表示すると自動的に反映されます。

Order Processing のテンプレートを作成する手順は次のとおりです。

1. Studio のフォルダ ツリーで [テンプレート] フォルダを右クリックします。表示されるポップアップ メニューから [テンプレートの作成] を選択します。[テンプレートのプロパティ] ダイアログ ボックスが表示されます。

図 5-2 [テンプレートのプロパティ] ダイアログ ボックス

2. [名前] フィールドに、`Order Processing` と入力します。[オーガニゼーション] リスト ボックスで [CDExpress] のチェック ボックスをチェックします。
3. [OK] をクリックします。Order Processing のテンプレートが、フォルダ ツリーの [テンプレート] フォルダの下に追加されます。

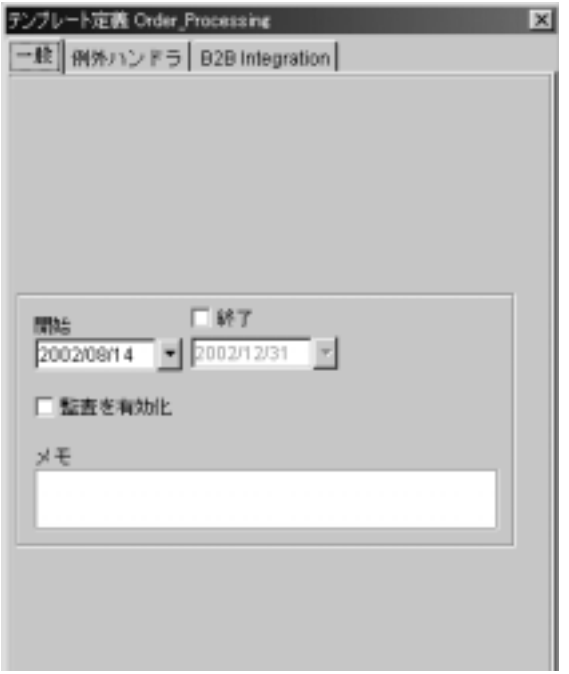
テンプレート定義の作成

テンプレート定義を作成するときに、有効期限の開始と終了を指定します。デフォルトでは、開始日は現在の日付であり、終了日は指定されていません。

新しいワークフロー テンプレートのテンプレート定義を作成する手順は次のとおりです。

1. [テンプレート] フォルダの下の Order Processing のテンプレートを右クリックします。表示されるポップアップ メニューから [テンプレート定義を作成] を選択します。[テンプレート定義 Order_Processing] ダイアログ ボックスが表示されます。

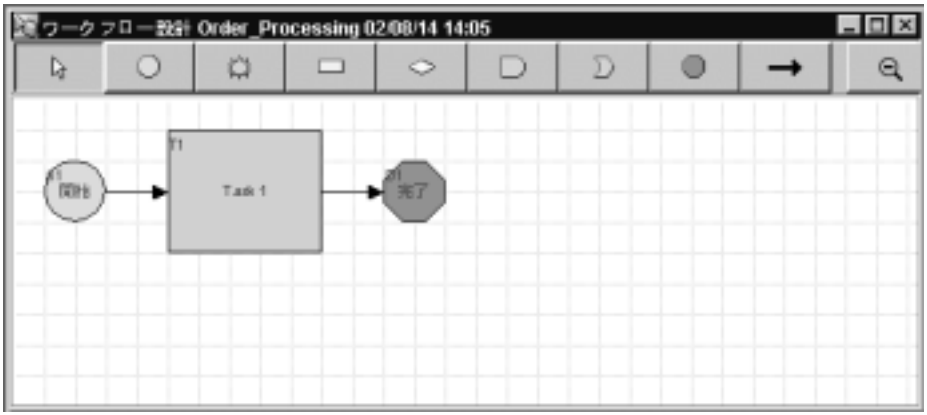
図 5-3 [テンプレート定義 Order_Processing] ダイアログ ボックス



2. （省略可能）[開始] と [終了]（ワークフローを確実に実行可能な期間）を入力し、[OK] をクリックします。

開始ノード、タスク ノード、完了ノード、そしてタスク ノード内に 2 種類のアクションを表示するシンプルなデフォルトのワークフロー設計ウィンドウが開きます。タスクをユーザ Workflow Attribute (“initiator”) に割り当てと タスクに完了マークを付ける を割り当てます。この 2 つは、実行可能なワークフローの定義に最低限必要なワークフロー オブジェクトです。

図 5-4 [ワークフロー設計] ウィンドウ



ワークフローの作図

[ワークフロー設計] ウィンドウのツールバー アイコンを使用して、独自のワークフローを作図できます。この節では、ノード シェイプをワークフロー設計領域に配置し、これらを接続して名前を変更することにより、Order Processing ワークフローを作図します。

シェイプを配置する

このワークフローの作図では次のシェイプを使用します。

- 4 種類のタスク（デフォルトのタスクに追加）
- 2 種類のイベント

- 2 種類の分岐ノード

シェイプを追加する手順は次のとおりです。

1. ツールバー上のシェイプのアイコンをクリックします。
2. ワークフロー設計領域の任意の場所でクリックします。
3. マウスのボタンを放してシェイプを領域内にドロップします。

注意： シェイプを配置すると、最初に番号が割り当てられます。ただし、ワークフロー内のシェイプの最終的な順番は重要ではありません。

- シェイプを移動するには、ワークフロー設計領域内の任意の位置にドラッグアンド ドロップします。
- シェイプを削除するには、シェイプを右クリックし、表示されるポップアップ メニューから [削除] を選択します。削除確認のダイアログ ボックスが表示されたら、[はい] をクリックします。

ノード名を変更する

ノードを接続する前に、分かりやすくするためにノード名を変更します。

注意： 分岐ノードを、通常は変数を含む式の名前に変更します。最初に条件式に使用する変数を設定する必要があるため、第 4 章「ワークフローのノードの定義」で分岐ノードに名前を付けます。

タスク及びイベント ノードの名称変更 :

1. ワークフロー設計領域で、ノードをダブルクリックします。[プロパティ] ダイアログ ボックスが表示されます。
2. タスク ノードの [名前] フィールドまたはイベント ノードの [説明] フィールドに、ノード名を入力します。次の表を参照してください。

表 5-2 Order Processing ワークフロー：タスクおよびイベント ノード

ノード	名前
T1	Check Customer Credit
T2	Confirm Order Fulfillment

表 5-2 Order Processing ワークフロー：タスクおよびイベント ノード

ノード	名前
T3	Check Inventory
T4	Start Order Fulfillment
T5	Contact Customer
E1	Wait for New Inventory
E2	Watch for Cancellation

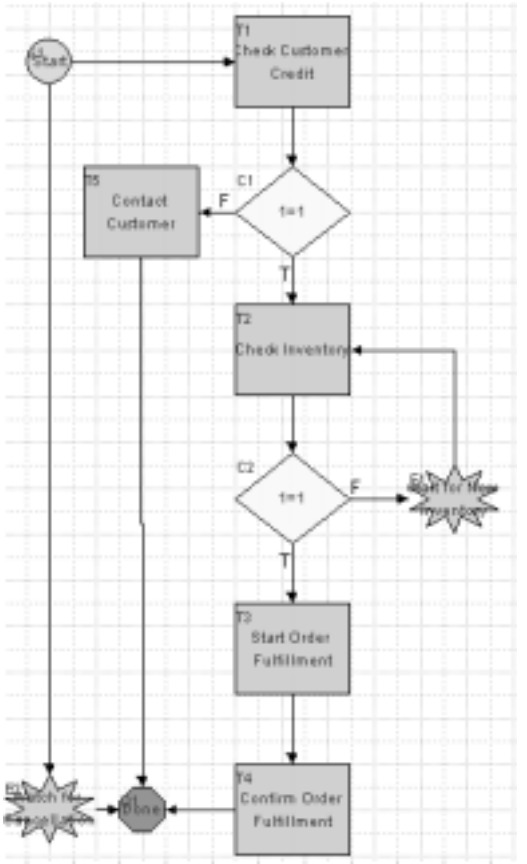
3. [OK] をクリックして、ダイアログ ボックスを終了します。これで、ワークフロー作図領域のノード上にノード名が表示されます。
4. 表にあるノードすべてに対して、1 から 3 までの手順を繰り返します。

注意： 入力するノード名が表内のノード番号と正確に対応していなくても気にしないでください。フロー内のシェイプの最終的な順番は重要ではありません。

シェイプを配置してノードを接続する

次に、シェイプを自由に配置して接続し、最終的に次の図のようにフローを作成します。

図 5-5　Order Processing ワークフロー：完成図



シェイプを移動するには、ワークフロー設計領域内の任意の位置にドラッグ アンド ドロップします。

接続するには、次の 2 つの方法があります。1 つはツールバーから行う方法、もう 1 つは、ノードの [プロパティ] ダイアログ ボックスから行う方法です。

ツールバーから接続する手順は次のとおりです。

1. ツールバー上の [コネクタ] ボタンをクリックします。

2. 接続元のノードをクリックし、接続先のノードにドラッグします。次に、マウスのボタンを放して接続を確定します。分岐ノードの場合は、True または False 接続のどちらかを選択するよう求められます。

ノードの [プロパティ] ダイアログ ボックスから接続するには、次の手順で行います。

1. 接続の起点とする接続元のノードをダブルクリックします。そのノードの [プロパティ] ダイアログ ボックスが表示されます。
2. [次へ] タブで、接続先のノードを選択する。分岐ノードの場合は、[次の True] タブと [次の False] タブからノードを選択します。

コネクタを削除するには、次の 2 つの方法があります。1 つは作図領域から削除する方法、もう 1 つはノードの [プロパティ] ダイアログ ボックスから削除する方法です。

作図領域内からコネクタを削除する手順は次のとおりです。

1. 削除したいコネクタ上で右クリックします。表示されるポップアップ メニューから [コネクタを削除] を選択します。
2. 確認ダイアログ ボックスが表示されたら [はい] を選択し、削除を確認します。

ノードの [プロパティ] ダイアログ ボックスからコネクタを削除する手順は、次のとおりです。

1. 接続元のノードをダブルクリックする。ノードの [プロパティ] ダイアログ ボックスを起動します。
2. [次へ] タブで、接続元のノードに接続されている接続先のノードの選択を解除します。分岐ノードの場合は、[次の True] と [次の False] タブからノードの選択を解除します。

ワークフロー ラベルの追加

3-13 ページの「ワークフローの式 : ワークフロー変数を設定 と ユーザにタスクを割り当て アクションの表示」で、Order Processing Trigger のワークフローでは、変数 OrderID が 1 づつ加算されるように設定しました。この節では、実行時アプリケーションにその情報を表示するラベルを作成し、ワークフローの各インスタンスごとにラベルがユニークになるようにします。

ワークフロー ラベルを作成するには、ワークフロー式を指定します。Order *番号* のラベルを表示するには、次の式を使用します。

コード リスト 5-1　ワークフロー ラベル

```
“Order ” + $OrderID
```

ラベルを追加するには、Order Processing Trigger ワークフロー内にすでに存在する変数に対応する、Order ID 変数も設定する必要があります。Expression Builder 内から変数を設定できるので、変数の設定と式の定義を結合する手順を次に示します。

Expression Builder で式を設定する

OrderID 変数を設定し、ワークフロー ラベルを定義する手順は次のとおりです。

1. Order Processing のテンプレート定義を開き、フォルダ ツリー内の定義フォルダ、またはワークフロー設計ウィンドウ内の任意の場所で右クリックして、表示されるポップアップ メニューから [プロパティ] を選択します。[テンプレート定義 Order_Processing] ダイアログ ボックスが表示され、そこに [ワークフロー ラベル] フィールドができています。

   注意 : このフィールドは、初めてテンプレート定義を作成したときは使用できません。その後、このダイアログ ボックスを開いて初めて有効になります。

図 5-6 [テンプレート定義 Order_Processing] ダイアログ ボックスの [ワークフロー ラベル] フィールド [ワークフロー ラベル] フィールド

2. [ワークフロー ラベル] フィールドの隣にある ボタンをクリックします。[Expression Builder] が表示されます。

図 5-7 [Expression Builder] による変数の設定

3. [変数] ラジオ ボタンをオンにし、右側のペインにある変数リストの中で、New をダブルクリックします。[変数プロパティ] ダイアログ ボックスが表示されます。

図 5-8 [変数プロパティ] ダイアログ ボックス

4. 下記の表にある OrderID 変数の関連情報を入力します。

表 5-3 テンプレート定義の ID 変数

変数名	データ型	パラメータタイプ	ワークフローでの使い方
OrderID	Integer	Input	Worklist 内にワークフロー ラベルとして表示し、ワークフローの実行のたびに加算する。 Order Fulfillment ワークフローに渡される。

これで、新しい変数がフォルダ ツリーの [変数] フォルダに表示されます。

5. [OK] をクリックして、[変数プロパティ] ダイアログ ボックスを終了します。これで、[Expression Builder] ダイアログ ボックスの [式] フィールドにこの変数が表示されます。

図 5-9 Expression Builder の使い方

6. コードリスト 5-1 で表示された式の設定を続けます。設定するには、以下のいずれかの方法を行います。
   - [式] ウィンドウに直接タイプします。
   - クオーテーション マークを入力するには、[リテラル] ラジオ ボタンをオンにして double quote を選択します。
   - プラス記号を入力するには、[演算子] ラジオ ボタンをオンにして plus を選択します。
   - 変数を入力するには、[変数] ラジオ ボタンをオンにして、使用する変数名をリストから選択します。

終了すると、[式] ウィンドウは次のようになります。

7. [OK] をクリックして、[Expression Builder] ダイアログ ボックスを終了します。これで、[テンプレート定義] のプロパティ ダイアログ ボックスの [ワークフロー ラベル] フィールドに、設定した式が表示されます。

8. [OK] をクリックして、[テンプレート定義 Order_Processing] ダイアログ ボックスを終了します。

ワークフローのアクティブ化

ワークフローを実行可能な状態にするには、ワークフローをアクティブにする必要があります。

Order Processing ワークフローをアクティブにする手順は、次のとおりです。

1. Order Processing のテンプレート定義を開き、フォルダ ツリー内の定義フォルダ、またはワークフロー設計ウィンドウ内の任意の場所で右クリックして、表示されるポップアップ メニューから [プロパティ] を選択します。[アクティブ] チェック ボックスがある [テンプレート定義 Order_Processing] ダイアログ ボックスが表示されます。

   注意 : このチェック ボックスは、初めてテンプレート定義を作成したときは使用できません。その後、このチェック ボックスを開いて初めて有効になります。

図 5-10 [テンプレート定義 Order_Processing] ダイアログ ボックスの [ワークフロー ラベル] フィールド [アクティブ] チェック ボックス

2. [アクティブ] チェック ボックスをチェックします。
3. [OK] をクリックして、ダイアログ ボックスを終了します。

ワークフローの保存

ワークフローを少なくとも一度保存しないと、ワークフローは実行できません。

Order Processing ワークフローを保存するには、テンプレート定義を開き、フォルダ ツリーでこのフォルダを右クリックして、表示されるポップアップ メニューから [保存] を選択します。

次の節で、ワークフローに必要なすべての変数とアクションを追加して定義します。この後、チュートリアルを続ける場合は、一定の間隔でワークフローを保存してください。

6　カスタム例外ハンドラの使い方

注意： Worklist クライアント アプリケーションは WebLogic Integration リリース 7.0 より非推奨となりました。代替機能に関する詳細については、『*WebLogic Integration リリース ノート*』を参照してください。

この節では、Order Fulfillment ワークフローで使用されるユーザ定義の例外ハンドラの例を示し、Studio の例外処理機能を中心に説明します。この節ではワークフロー内のすべてのノードを概説した後で、以下の方法について説明します。

- カスタム例外ハンドラとそのアクションを定義する。
- カスタム例外ハンドラを起動する。
- 特定の例外を検出するための条件を評価する。
- 実行時エラーを修正するため、クライアントへ XML メッセージを送信する。

チュートリアルのこの節では、Order Fulfillment ワークフロー、Create OrderBean と Calculate Total Price ビジネス オペレーション、cancelledorder イベント キーが、2-7 ページの「ワークフロー オブジェクトのインポート：チュートリアル パッケージ ファイルのインポート」の説明に従ってインポート済みであることを前提とします。

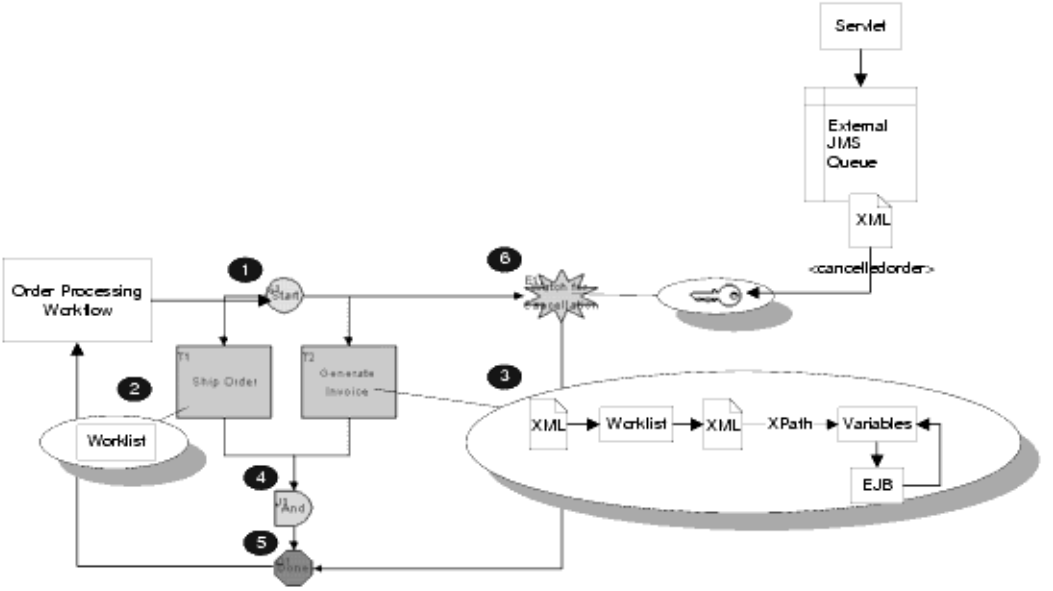
Order Fulfillment ワークフローの概要

Order Processing ワークフローから呼び出される Order Fulfillment ワークフローには、受注品の出荷と請求を処理するタスクが含まれています。ワークフロー内の 2 つのタスクは平行して実行する必要があるので、両者は開始ノードに直接結び付けられます。両方のタスクがワークフローの終了前に必ず実行されるようにするため、AND ノードで 2 つのタスクを結合します。

また、ワークフローは、Order Processing Trigger ワークフロー内の元の XML ドキュメントで指定されていた無効な国名の省略形の修正を Worklist ユーザに求めるカスタム例外ハンドラを定義し、呼び出します。

Order Fulfillment ワークフローの詳細を次の図に示します。図中に表示された番号が振られている手順については、図の下の表で説明しています。この表では、実際のプロセスをサンプル ワークフロー内の実際の実装にマップします。

図 6-1 Start Order Fulfillment ワークフロー：詳細表示



例外ハンドラ

すべての WebLogic Integration ワークフローのテンプレート定義には、システム例外ハンドラというデフォルトの例外ハンドラが設定されています。システム例外ハンドラは、アクティブなトランザクションをロールバックのみにマーク付けしたり、例外をクライアントに送出したりすることにより、実行時に発生する例外に応答します。

カスタム例外ハンドラを定義し、ワークフローのアクションを例外発生時に実行させることもできます。カスタム例外ハンドラをコンフィグレーションすると、実行時に発生するすべての例外の検出や、呼び出し EJB に定義済みの特定の例外処理が可能になります。カスタム例外ハンドラを定義するには、アクティブなトランザクションに対して実行される*コミット*および*ロールバック*アクションを指定します（詳細は、「例外ハンドラのアクション」を参照してください）。

ワークフロー レベルの例外ハンドラは、任意のノード内から呼び出すことができます。例外ハンドラを定義した後で、以下の 2 つのアクションのいずれかを使用して、ノード内から例外ハンドラを起動します。

- ワークフロー例外ハンドラを設定 ー システム例外ハンドラを非アクティブ化し、指定のワークフローの有効期間に例外ハンドラを設定します。例外ハンドラ アクションは、例外が実際に送出されたときにのみ起動されます。必要に応じてこの例外アクションを使用して、例外ハンドラをシステム例外ハンドラにリセットできます。
- 例外ハンドラの呼び出し ー このアクションは、ワークフローの任意の時点で、例外が実際に発生するかどうかとは無関係に、指定の例外ハンドラで定義されたアクションを開始します。システム例外ハンドラは、コントロールがメイン ワークフローに戻るときにリセットされます。

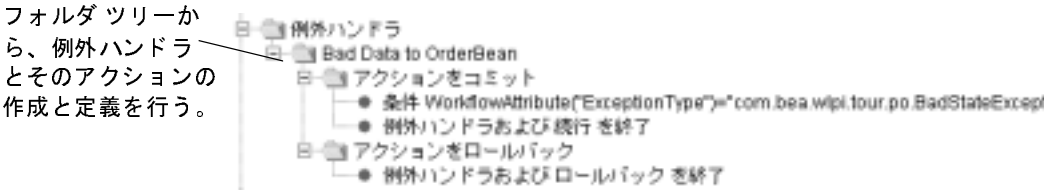
フォルダ ツリーと、テンプレート定義の [プロパティ] ダイアログ ボックスにある [例外ハンドラ] タブの両方から、例外ハンドラを新規に作成したり既存の例外ハンドラを表示したりできます。

フォルダ ツリーで Order Fulfillment ワークフロー用に定義された例外ハンドラを表示する手順は、次のとおりです。

1. フォルダ ツリーで、[Order Fulfillment] テンプレートを展開し、テンプレート定義を右クリックします。表示されるポップアップ メニューから [開く] を選択してワークフローを開きます。

2. フォルダ ツリーで [例外ハンドラ] フォルダを展開します。展開すると次の図のようになります。

図 6-2 Start Order Fulfillment ワークフロー : 例外ハンドラ



このワークフローには、Bad Data to OrderBean というユーザ定義例外ハンドラが 1 つ含まれます。

3. Bad Data to OrderBean 例外ハンドラが呼び出される位置を確認します。フォルダ ツリーでそのフォルダを右クリックし、表示されるポップアップ メニューから [使用] を選択します。[例外ハンドラの使用場所] ダイアログ ボックスが表示されます。

図 6-3 [例外ハンドラの使用場所] ダイアログ ボックス

このウィンドウで示すように、カスタム例外ハンドラは [Generate Invoice] ノードから呼び出されます。次に、このノードについてもっと詳しく検討したうえで、Bad Data to OrderBean 例外ハンドラのプロパティを詳細に見ることにします。

4. [OK] をクリックします。[例外ハンドラの使用場所] ダイアログ ボックスを終了します。

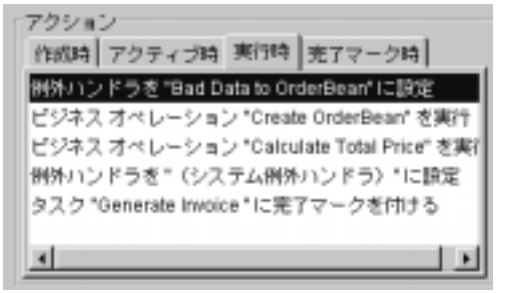
Generate Invoice タスクの表示

最初に、ビジネス オペレーションや Bad Data to OrderBean カスタム例外ハンドラへの呼び出しを含め、Generate Invoice タスクを構成するアクションを見てみましょう。

Generate Invoice タスクのプロパティを表示する手順は、次のとおりです。

1. [ワークフロー設計] ウィンドウ内で、Generate Invoice タスクをダブルクリックします。[タスクのプロパティ] ダイアログ ボックスを表示します。[アクティブ時] タブには、Accounting ロールにタスクを割り当てるアクションが 1 つ含まれています。
2. [実行時] タブを選択します。

図 6-4 [Generate Invoice タスク プロパティ] ダイアログ ボックス : [実行時] タブ



[実行時] タブには以下のアクションが表示されます。

- 例外ハンドラを “Bad Data to OrderBean” に設定 ー これは、この時点以降ワークフローの有効期間中に、ワークフローが別の例外ハンドラを設定するまで、ユーザ定義の例外ハンドラを設定する *例外処理* アクションです。例外ハンドラの定義については、「カスタム例外ハンドラの定義 Bad Data to OrderBean 例外ハンドラ」で詳細に説明しています。
- ビジネス オペレーション “Create OrderBean” を実行 ー これは、`POBean` 上の `checkInventory()` メソッドを呼び出す前に、4-44 ページの「Create OrderBean ビジネス オペレーションを実行する」で使用した統合アクションと同じものです。このアクションでは、実行時にサーバ上に EJB のインスタンスが作成されます。
- ビジネス オペレーション “Calculate Total Price” を実行 ー これは、`POBean` 上の `calculate()` メソッドを呼び出す統合アクションです。「Calculate Total Price ビジネス オペレーションの表示」で詳細に説明しています。
- 例外ハンドラを “(system exception handler)” に設定 ー これは、ワークフローの例外ハンドラをリセットして、実行時にすべてのワークフローに有効なデフォルトのシステム例外ハンドラに戻す例外処理アクションです。
- タスク Generate Invoice に完了マークを付ける ー これは、Generate Invoice タスクを完了するタスク アクションです。Shipping タスクも完了にマークされたときに、ワークフローは完了ノードに進み、ワークフローは終了します。

Calculate Total Price ビジネス オペレーションの表示

ビジネス オペレーション Calculate Total Price を実行 アクションをダブルクリックして、プロパティを表示します。

図 6-5 [ビジネス オペレーションを実行] ダイアログ ボックス : Calculate Total Price アクション

`calculate()` メソッドは、国または州の売上税を含む合計金額を OrderTotalPrice 変数に返します。

`calculate()` メソッドに関する詳細は、ビジネス オペレーションのコンフィグレーションを参照してください。

Calculate Total Price ビジネス オペレーションのコンフィグレーションを表示する手順は次のとおりです。

1. [取消し] をクリックして、[ビジネス オペレーションを実行] ダイアログ ボックスを終了します。
2. [取消し] をクリックして、[タスクのプロパティ] ダイアログ ボックスを終了します。
3. [コンフィグレーション | ビジネス オペレーション] を選択します。[ビジネス オペレーション] ダイアログ ボックスが表示されます。
4. ビジネス オペレーションのリストで、Calculate Total Price ビジネス オペレーションをダブルクリックします。[ビジネス オペレーションを定義] ダイアログ ボックスが表示されます。

図 6-6 [ビジネス オペレーションを定義] ダイアログ ボックス : Calculate Total Price アクション

[呼び出すメソッド] フィールドで、`calculate()` メソッドによる例外 `BadStateException` の送出を確認できます。EJB のクラス ファイルでは、この例外は、無効な州データがメソッドに渡された場合に発生するように定義されています。ソース コードにおいて無効な州データとは、アメリカが州名として採用する有効な 2 文字の略称と整合がとれないテキストのことです。

この例外の送出に備えて、Bad Data to OrderBean 例外ハンドラがワークフローに設定され、実行時にエラーを修正します。この例外ハンドラを、以下の節で更に詳しく説明します。

注意 : `POBean.jar` EJB の詳細は、『*BEA WebLogic Integration Javadoc*』を参照してください。

5. [取消し] をクリックして、[ビジネス オペレーションを定義] ダイアログ ボックスを終了します。
6. [閉じる] をクリックして、[ビジネス オペレーション] ダイアログ ボックスを終了します。

カスタム例外ハンドラの定義 Bad Data to OrderBean 例外ハンドラ

第 3 章「ワークフロー オブジェクトとプロパティ」で、Order Processing ワークフローをトリガした Neworder XML ドキュメントに含まれる <shiptostate> 値内の POBean に無効なデータのオカレンスが渡される場合をシミュレートしました。そのワークフローでは、ShipToState 変数を不正な値 KK に設定し、それが [Start Order Fulfillment] タスク ノードによって Order Fulfillment ワークフローに渡されていました。

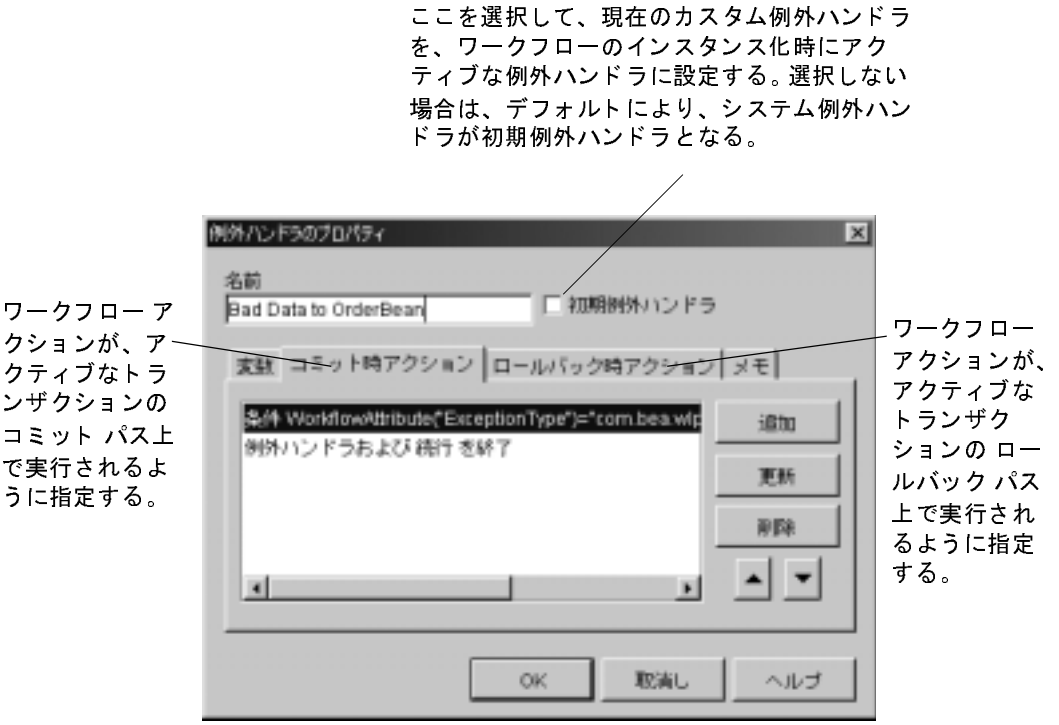
Generate Invoice タスクが実行されると、ビジネス オペレーションを実行 アクションは間違った ShipToState 変数の値を入力に使用して、calculate() メソッドを呼び出そうと試みます。POBean の calculate() メソッドは、無効な州データを受け取った場合に BadStateException を送出するため、サーバはカスタム定義の例外ハンドラ Bad Data to OrderBean を呼び出します。

ここで、Bad Data to OrderBean 例外ハンドラと、その中に定義されているアクションを見てみましょう。

Bad Data to OrderBean 例外ハンドラを表示する手順は次のとおりです。

1. フォルダ ツリーで、[例外ハンドラ] フォルダを展開し、Bad Data to OrderBean 例外ハンドラを右クリックします。表示されるポップアップ メニューから [プロパティ] を選択します。[例外ハンドラのプロパティ] ダイアログ ボックスが表示されます。

図 6-7 [例外ハンドラのプロパティ] ダイアログ ボックス



例外ハンドラのアクション

[コミット時アクション] と [ロールバック時アクション] タブを使用すると、トランザクションのコミット パスやロールバック パスで実行するワークフロー アクションを指定できます。トランザクションがロールバック専用にマークされている場合を除き、ほとんどの場合はトランザクションがコミットされます。この場合、例外ハンドラはロールバック タブで指定されたアクションを実行します。呼び出された EJB メソッドでトランザクションをロールバックのみにマークすることは可能ですが、両方のタイプのアクションを指定することをお勧めします。

- コミット時アクション ー このタブで、例外ハンドラを終了アクションを含め、以下のオプションを付けた任意のワークフロー アクションを指定する。

- 終了してロールバック
- 終了して停止 (残っている処理を実行しない)
- 終了して再試行 (失敗した処理を再試行する)
- 終了して続行 (次の処理で再開する)

■ ロールバック時アクション ー 使用可能なアクションは、ワークフロー変数を設定、統合 アクション、例外ハンドラおよびロールバックを終了、電子メール送信、処理なし、条件を評価、監査エントリを作成 です。

[例外ハンドラのプロパティ] ダイアログ ボックスにあるように、Bad Data to OrderBean 例外ハンドラは以下のアクションを定義しています。

■ コミット

- 条件を評価 ー例外処理アクションの適用対象となる特定の条件をテストしたり、テスト結果に基づいて true または false アクションを指定したりするその他のアクションです。このアクションについては、「条件を評価アクションの表示」で詳細を説明します。
- 例外ハンドラを終了して続行 ー 例外ハンドラを停止し、次の操作でワークフローの実行を継続しようとする例外処理アクションです（この場合、例外の原因となったビジネス オペレーションのあとに続くアクション）。

■ ロールバック

例外ハンドラを終了してロールバックー例外ハンドラを停止して、アクティブなトランザクションを例外が送出される前の状態にロールバックし、例外をクライアントに再送出する例外処理アクションです。

条件を評価 アクションの表示

条件を評価 アクションを表示するには、[例外ハンドラのプロパティ] ダイアログ ボックスで、条件を評価 アクションをダブルクリックします。[条件を評価] ダイアログ ボックスが表示されます。

図 6-8 [条件を評価] ダイアログ ボックス

条件は次の式で定義されます。

```
WorkflowAttribute(“ExceptionType”) =
“com.bea.wlpi.tour.po.BadStateException”
```

この場合、この式はワークフロー属性 例外タイプ を使用して、送出された例外が `POBean` で指定された `BadStateException` と等しいかどうかをテストします。このような条件を使用することは、特定の例外を検出するように例外ハンドラを設定する場合の一般的な方法です。WorkflowAttribute 関数を使用して、以下に挙げるさまざまな例外の属性によって目的の例外を識別することができます。

- "ExceptionType"(String) − 例外の完全修飾 Java クラス名
- "Exception Text"(String) − 例外メッセージのテキスト
- "ExceptionObject"(Exception) − 例外オブジェクト自体

ワークフローの関数と属性の全リストについては、『*WebLogic Integration Studio ユーザーズ ガイド*』の「ワークフローの式を使用する」を参照してください。

条件を評価 アクションは、テスト結果、つまり、式による true または false の評価に従って実行するアクションを指定します。この例では、このアクションは以下のようなアクションを指定します。

- True

  XML をクライアントに送信 − `calculate()` メソッドに渡された無効なデータの修正を Worklist ユーザに求める XML メッセージを Worklist ユーザに送信し、Generate Invoice タスクを再実行する統合アクションです。このアクションについては、「XML をクライアントに送信 アクションの表示」で詳しく説明します。

- False

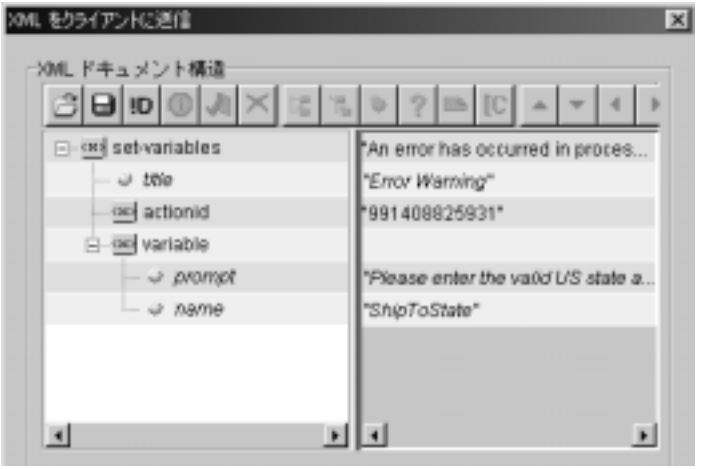
  ここではアクションの指定はありません。条件によって false となった場合、例外ハンドラが、[例外ハンドラのプロパティ] ダイアログ ボックスで指定された ロールバック アクションを実行します。

XML をクライアントに送信 アクションの表示

XML をクライアントに送信 アクションは、`POBean` に渡され、BadStateException の発生原因となった誤った不正なデータの修正を求めるメッセージを Worklist ユーザに送信します。次に、[Generate Invoice タスク プロパティ] ダイアログ ボックスの [実行時] タブで指定されたアクションを再実行し、タスクが正常に終了するようにします。

XML をクライアントに送信 アクションを表示するには、[条件を評価] ダイアログ ボックスで、[True] タブを選択し、続いて XML をクライアントに送信 アクションをダブルクリックします。[XML をクライアントに送信] ダイアログ ボックスが表示されます。

図 6-9 [Bad Data to OrderBean] 例外ハンドラ : [XML をクライアントに送信] ダイアログ ボックス



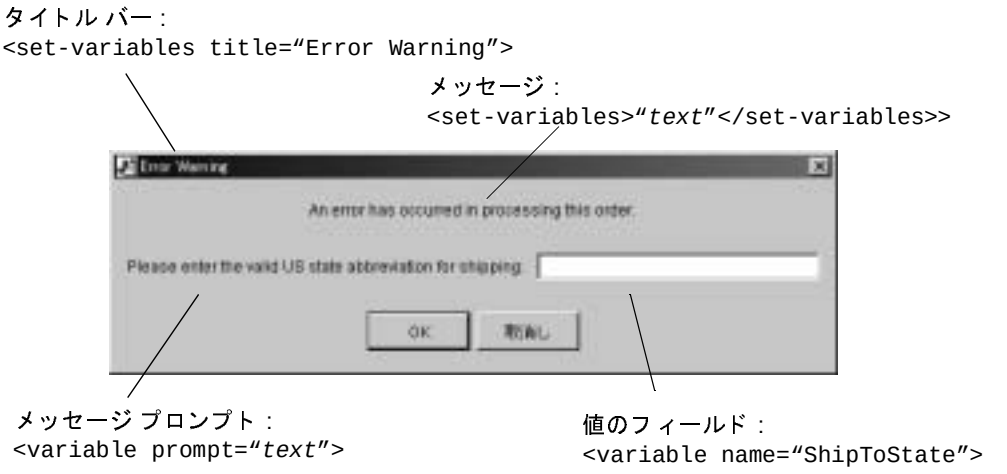
この XML をクライアントに送信 アクションでは、`ClientSetVarsReq` DTD に基づき、XML ドキュメントが Worklist クライアントに送信され、Worklist ユーザが情報を入力するためのフィールドを含むメッセージ ボックスが Worklist によって表示されます。XML ドキュメントは、次のような構造になっています。

コード リスト 6-1 ClientSetVarsReq XML ドキュメントの構造

```
<set-variables title="text">
text
   <actionid>provided by default</actionid>
   <variable name="variable name" prompt="text" />
   [<variable name="variable name" prompt="text" />]
</set-variables>
```

Worklist クライアントによって表示されるメッセージ ボックスに対応する XML ドキュメント内のタグと値を、次の図に示します。

図 6-10 [Error Warning] メッセージ ボックス



Worklist ユーザからの応答を取り込むために、データは `ClientSetVarsResp.dtd` に基づいて Worklist クライアントが返した XML ドキュメント、特にコード リスト 6-2 に示す `<variable name>` 属性から抽出されます。

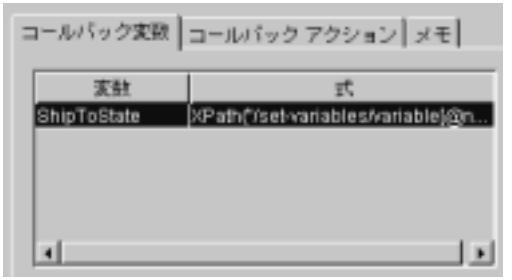
コード リスト 6-2 ClientMsgBoxReq XML ドキュメントの構造

```
<set-variables>
   <variable name="variable name" />
   [<variable name="variable name" />]
</set-variables>
```

必要な情報を抽出するために、次の XPath 式が定義されます。

```
XPath("/set-variables/variable[@name=\"ShipToState\"]/text()")
```

次に、[XML をクライアントに送信] ダイアログ ボックスの [コールバック変数] タブで、結果が ShipToState 変数に割り当てられます。



最後にコールバック アクションが指定されます。

このアクションは、Worklist ユーザが有効なデータを入力した後で、[Generate Invoice タスク プロパティ] ダイアログ ボックスの [実行時] タブにリストされるすべてのアクションを再実行します（6-5 ページの「Generate Invoice タスクの表示」を参照）。ユーザが無効なデータを入力した場合は、許容できるデータが入力されるまで、メッセージ ボックスが繰り返し表示され、正しく入力するよう求められます。

Generate Invoice タスクの最後のアクションによってタスクの完了がマークされると、ワークフローは AND ノードに進みます。Ship Order タスクが完了とマークされると、ワークフローは終了します。この時点で、ワークフローの実行中に収集された変数値が、Order Processing ワークフローに返されます。

7 サンプル ワークフローの実行とモニタ

注意： Worklist クライアント アプリケーションは WebLogic Integration リリース 7.0 より非推奨となりました。代替機能に関する詳細については、『*WebLogic Integration リリース ノート*』を参照してください。

この節では、WebLogic Integration の Worklist からサンプル ワークフローを実行します。次に、Studio でワークフローの実行中のインスタンスをモニタします。

この節で説明する手順を実行する以前に、以下の作業が完了していることを前提にしています。

- 2-2 ページの「WebLogic Integration Studio の起動」の説明に従って、Studio にログオンしていること。
- Order Processing Trigger と Order Fulfillment のテンプレート定義とその依存関係を、2-7 ページの「ワークフロー オブジェクトのインポート：チュートリアル パッケージ ファイルのインポート」の説明に従ってインポートし、アクティブ化が完了していること。
- 2-7 ページの「ワークフロー オブジェクトのインポート：チュートリアル パッケージ ファイルのインポート」の説明に従って、Order Processing のテンプレート定義とその依存関係がインポートされている、または、第 5 章「ワークフローの作成」および第 4 章「ワークフローのノードの定義」の説明に従って、テンプレートを最初から作成し定義していること。

注意： Worklist および Studio のモニタ機能がこの節で説明されますが、この節はこれらのチュートリアルではありません。これらのトピックの詳細は、『*WebLogic Integration Worklist ユーザーズ ガイド*』と、『*WebLogic Integration Studio ユーザーズ ガイド*』の「ワークフローをモニタする」を参照してください。

Worklist アプリケーション内でのワークフローの実行

Order processing Trigger ワークフローには手動による開始が含まれているので、Worklist アプリケーションからこのワークフローを実行できます。このワークフローは次に、Order Processing と Order Fulfillment ワークフローのすべてのタスクを開始します。この節では、以下のことを行います。

- Worklist へのログ オン
- Order Processing Trigger ワークフローの開始
- Order Processing ワークフロー内のタスクの実行
- Order Fulfillment ワークフロー内のタスクの実行

Worklist アプリケーションにログ オンする

ユーザ admin を、サンプルワークフロー内で使用されるすべてのロールのメンバーとして定義しているので、admin としてログ オンします。

WebLogic Integration Worklist を起動する手順は、次のとおりです。

1. 以下のいずれか 1 つを実行します。
   - Windows システムでは、[スタート ｜プログラム｜ BEA WebLogic Platform 7.0 ｜ WebLogic Integration 7.0 ｜ Worklist] の順に選択します。
   - UNIX システムでは、`WLI_HOME/bin` ディレクトリに移動し、コマンド プロンプトに次のように入力して Worklist 起動スクリプトを実行します。

     ```
     sh worklist.sh
     ```

   [WebLogic Integration へのログオン] ダイアログ ボックスが [WebLogic Integration Worklist] アプリケーション ウィンドウの前面に表示されます。

図 7-1 [WebLogic Integration へのログオン] ダイアログ ボックス

2. [ユーザ名] フィールドに `admin` と入力します。
3. [パスワード] フィールドに `security` と入力します。

   注意： ユーザ名とパスワードは大文字と小文字の区別が必要です。パスワードとユーザ名は、必ず小文字で入力してください。

4. [Server [:port]] フィールドに、WebLogic Integration サーバ アプリケーションを実行中のシステムを次のように指定します。

   `t3://`*`host`*`:7001`

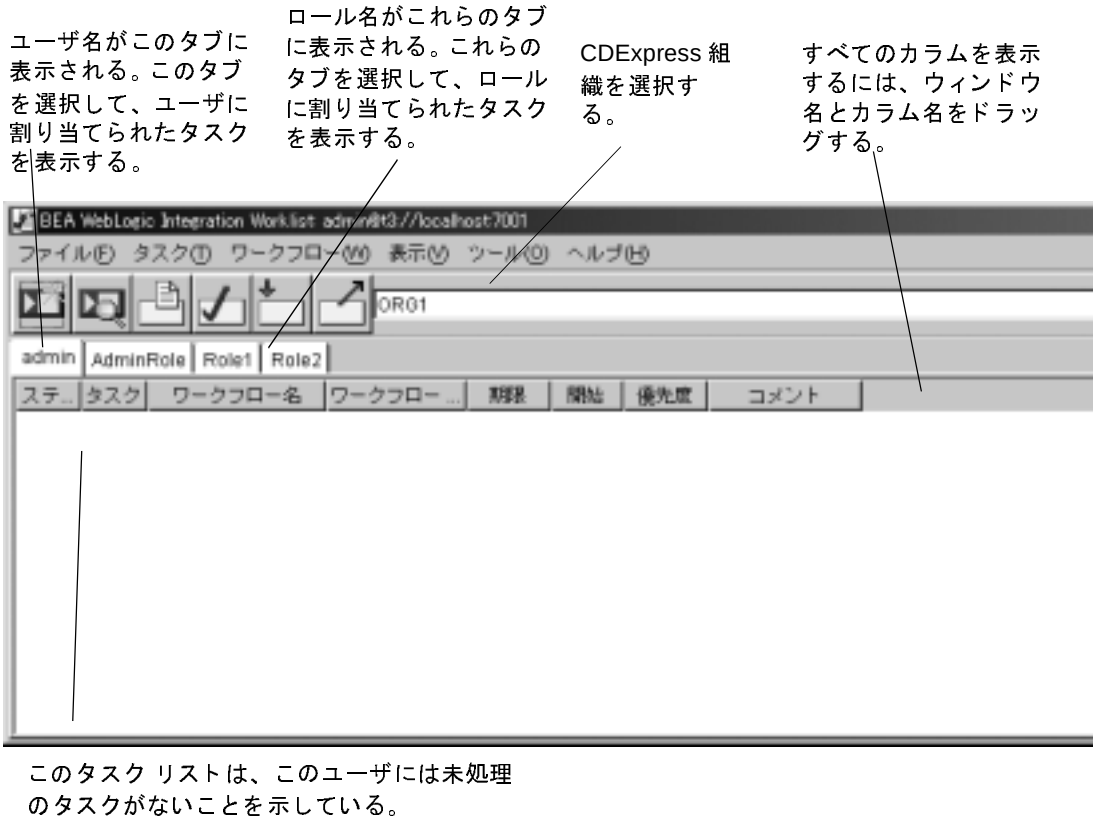
   この *`host`* とは、コンピュータ名または WebLogic Integration サーバ実行中のシステムの IP アドレスです。サーバが Worklist アプリケーションと同じコンピュータ上で実行中の場合は、`localhost` と指定します。

5. [OK] をクリックします。[WebLogic Integration Worklist] ダイアログ ボックスが表示されます。

図 7-2　[WebLogic Integration Worklist] ダイアログ ボックス

6. [OK] をクリックします。[Worklist] のメイン ウィンドウが表示されます。

図 7-3　Worklist アプリケーションのメイン ウィンドウ



7. ウィンドウの最上部にあるドロップダウン リストから [CDExpress] を選択します。その組織のユーザに関連付けられたロールを示すタブが、ユーザ名タブの隣に表示されます。この場合、ロールは、[Accounting]、[CustomerService]、[Shipping] です。
8. [表示] メニューから、[完了] チェック ボックスをチェックします。これを選択すると、[ワークリスト] ウィンドウに、保留中 と 完了 タスクの両方を表示できます。

サンプル ワークフローを開始する

手動で開始するように作られている Order Processing Trigger ワークフローを開始して、サンプル ワークフローを開始します。このワークフローは、サンプルのメイン ワークフローである Order Processing をオフに設定します。

Worklist ユーザは全員がこのワークフローを開始できます。

[Order Processing Trigger] ウィンドウを開く手順は、次のとおりです。

1. [ワークフロー（W）｜ Start a Workflow ワークフローを開始（S）] を選択します。[ワークフローを開始] ダイアログ ボックスが表示されます。

図 7-4 [ワークフローを開始] ダイアログ ボックス

2. [オーガニゼーションを選択] ドロップダウン リストで [CDExpress] を選択します。
3. [開始するワークフローを選択] ウィンドウから [Order Processing Trigger] を選択し、[OK] をクリックします。ワークフローのインスタンスが作成され、ワークフローの開始が成功したことを確認するメッセージが表示されます。

ワークフローの最初のタスクが、ワークフローを開始ユーザに割り当てられるので（詳細は、3-9 ページの「タスク ノードのプロパティ : Start Order Processing タスク ノードの表示」を参照）、次のメッセージも表示されます。

4. [OK] をクリックします。メッセージ ボックスが両方とも終了します。Start Order Processing タスクが Worklist で同じユーザ名の下に表示されます。

図 7-5 Worklist アプリケーションのタスク リスト

5. タスクを実行するには、タスク リスト内のタスクをダブルクリックします。タスクが実行されると、Order Processing ワークフローを開始する XML ドキュメントが作成され、最初のタスクが admin に割り当てられます。

[Start Order Processing] タスクのステータスが完了には変わらないことに注意してください。これは、ワークフローの設計で、ループ メカニズムで指定され、完了ノードが欠落しているので、タスクが無限に割り当てられるためです（3-9 ページの「タスク ノードのプロパティ : Start Order Processing タスク ノードの表示」を参照してください）。

このタスクを実行するたびに、このタスクの別のインスタンスが admin のタスク リストに表示されます。したがって、Order Processing ワークフローを開始しようとするたびに、[ワークフロー] メニューから Order Processing Trigger ワークフローを開始する必要がなく、[ワークリスト] で Start Order Processing タスクをダブルクリックするだけで開始できます。

Order Processing ワークフロー タスクを実行する

Order Processing ワークフローの最初のタスクは信用チェックであり、ユーザ admin に割り当てられました。この節では、Check Customer Credit タスクに対する [はい] の応答以降のワークフロー パスで指定されているすべてのタスクを実行します。

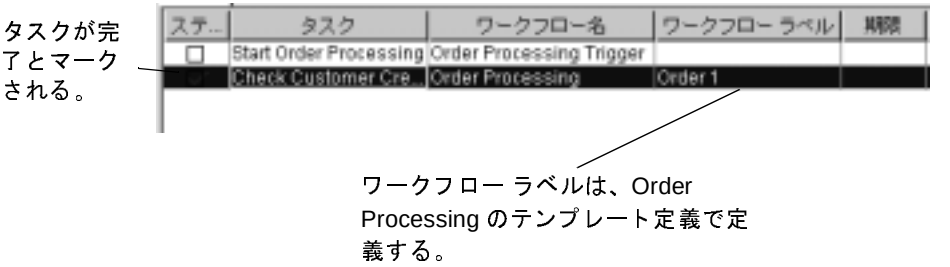
Order Processing ワークフロー タスクを実行する手順は、次のとおりです。

1. 新しいタスクを通知するメッセージ ボックスで、[OK] をクリックします。
2. [admin] ユーザ タブで、Check Customer Credit タスクをダブルクリックして、そのタスクを実行します。[Credit Check] メッセージ ボックスが表示されます （このメッセージ ボックスは、4-26 ページの「XML ドキュメント構造体を定義する」でセットアップ済みです）。

図 7-6 [Credit Check] メッセージ ボックス

3. このメッセージ ボックス内で [はい] をクリックします。これで、[ステータス] カラムのチェック マークからわかるように、このタスクに完了のマークが付けられます。

図 7-7　完了マークが付いたタスク



ここで、Check Customer Credit タスクが含まれる Order Processing ワークフローのステータスを表示します。[Check Customer Credit] タスクを選択し、[ワークフロー | ワークフローのステータス] を選択します。[ワークフローのステータス] ダイアログ ボックスが表示されます。

図 7-8　[ワークフローのステータス] ダイアログ ボックス

自動的に行われるタスクである Check Inventory タスクには、完了マークがすでに付いています。在庫チェックを通過すると（このサンプルでは常時行われる）、次のタスクは、Order Fulfillment サブワークフローを自動的に開始する Start Order Fulfillment タスクです。

Order Fulfillment ワークフロー タスクを実行する

ワークフローは Generate Invoice タスクを Accounting ロールに、Ship Order タスクを Shipping ロールに同時に割り当てます。[Accounting] ロール タブと [Shipping] ロール タブを切り替えると、タスクが同時に表示されたことを確認できます。これらのタスクを実行する順序は重要ではありませんが、ここでは出荷タスクから始めます。

Accounting ロールのメンバーとして定義された joe で Worklist にログ オンした場合、Generate Invoice タスクが [Accounting] ロール タブの joe のワークリストにも含まれています。Shipping ロールのメンバーとして定義された mary で Worklist にログ オンした場合、Ship Order タスクも [Shipping] ロール タブの mary のワークリストに含まれます。これらのワークリストからタスクを実行しようとする場合、この 2 人のユーザ名でログ オンし、Worklist の別のインスタンスを開きます。2 人のユーザのパスワードを、`password` と入力します。

Order Fulfillment ワークフロー タスクを実行する手順は、次のとおりです。

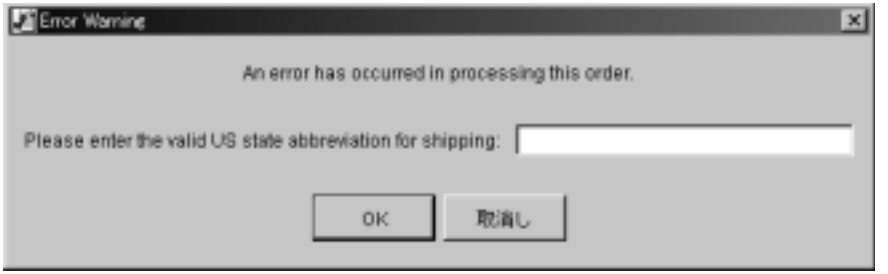
1. [Shipping] タブを選択します。Ship Order タスクに出荷指示コメントが含まれます。このコメントは、ワークフローのテンプレート定義で指定されたものです。
2. [Ship Order] タスクをダブルクリックして、このタスクを実行します。
3. [Accounting] タブを選択します。
4. [Accounting] タスクをダブルクリックして、このタスクを実行します。

   6-9 ページの「カスタム例外ハンドラの定義 Bad Data to OrderBean 例外ハンドラ」で説明したように、このタスクを実行すると、Calculate Total Price ビジネス オペレーションが実行されます。その結果、この操作によって、Order Processing Trigger ワークフローの XML イベントに含まれた不正な `KK` という州名データが、`POBean` EJB 上の `calculate()` メソッドに渡されることになります。次に `calculate()` メソッドによって、サーバが `BadStateException` 例外を送出します。

   Order Fulfillment ワークフローは例外を検出するための例外ハンドラを定義するので、Worklist クライアント上では例外に関する表示されません。代わりに、Bad Data to OrderBean 例外ハンドラで定義された XML をクライアントに送信 アクションが指定するメッセージ ボックスが表示されるだけです

（詳細については、6-13 ページの「XML をクライアントに送信 アクションの表示」を参照してください）。

図 7-9 [Error Warning] ダイアログ ボックス



再び無効な州データを入力した場合、どうなるかを確認します。

5. メッセージ ボックス フィールドに、XYZ のような無効データを入力します。同じメッセージが表示されて、再び修正を求められます。
6. ここで、CA のような有効な 2 文字の州名の略称を入力します。

今度は、Generate Invoice タスクが実行され、完了マークが付けられます。Calculate Total Price ビジネス オペレーションは、注文の合計金額を計算し、値を Order Processing ワークフローに返します。Confirm Order Fulfillment タスクは自動的に実行され、プロセス全体が終了します。

Order Processing ワークフローを定義するときに、電子メール メッセージを送信アクションを入れると（4-64 ページの「電子メール メッセージの送信 : Confirm Order Fulfillment タスクの定義」を参照してください）、自分が書いたテキストを記した 電子メールを受け取ることになります。

電子メール メッセージを送信 アクションを入れない場合は、以下の節の手順に従って全ワークフロー操作の結果を表示できます。

Studio による実行中のワークフローのモニタ

この節では、Order Processing ワークフローに指定された次の 2 つのパスに従って、ワークフローを 2 回実行します。1 回目は、Check Customer Credit タスクに いいえ を返し、Order Processing ワークフローを終了する Contact Customer タスクを実行します。2 回目は はい を返して、Order Processing ワークフローと Order Fulfillment ワークフロー両方の残りのタスクを実行します。

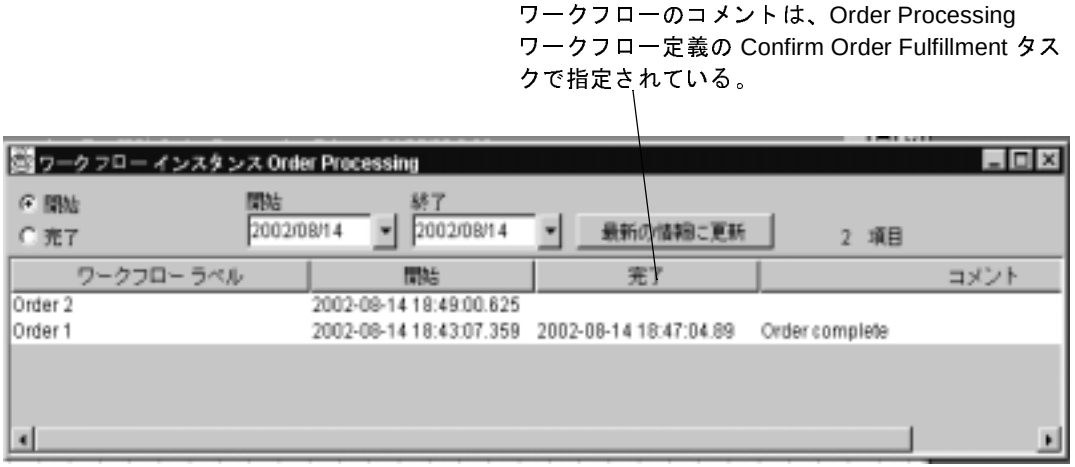
Worklist 内でワークフローを実行するのと同時に、Studio から実行中のワークフローのインスタンスをモニタします。CDExpress 組織の Worklist と Studio のインスタンスを両方とも開く必要があります。

Order Processing ワークフローのインスタンスの実行とモニタの手順は、次のとおりです。

1. Worklist で、[admin] タブを選択します。Start Order Processing タスクをダブルクリックします。Order Processing ワークフローが開始します。
2. Studio で、フォルダ ツリー内の Order Processing のテンプレートを右クリックします。表示されるポップアップ メニューから [インスタンス] を選択します。[ワークフロー インスタンス] ダイアログ ボックスが表示されます。

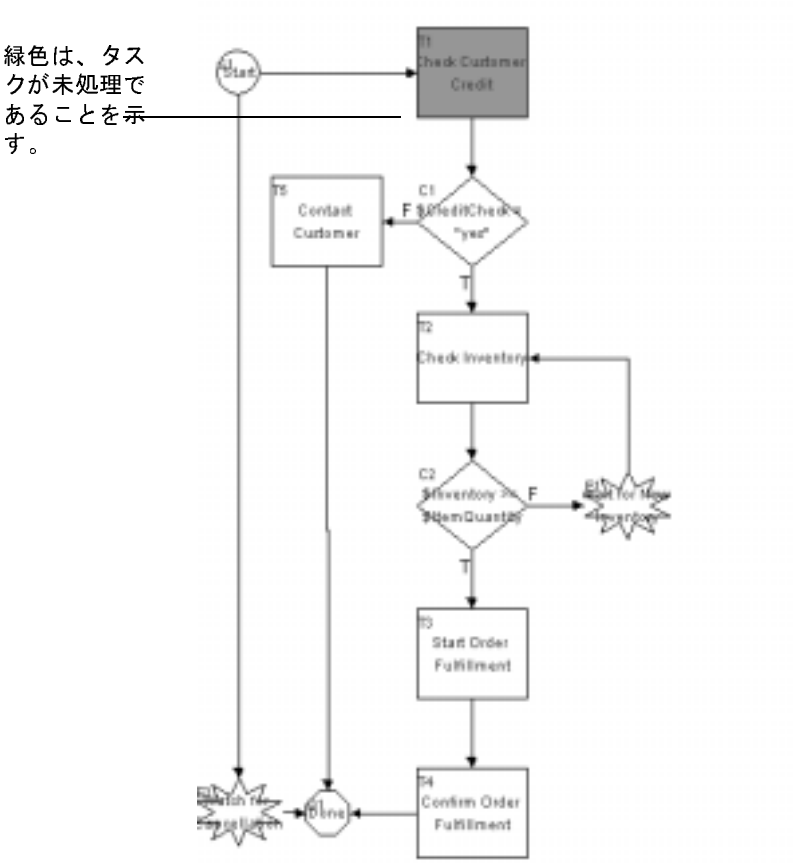
   2 つのインスタンスが表示されます。1 つ目は、前の章で実行したワークフローで、2 つ目は、今開始したばかりのインスタンスです。Order ID の番号が不得手、現在は 2 になっていることに注目してください。

図 7-10 [ワークフロー インスタンス Order Processing] ダイアログ ボックス



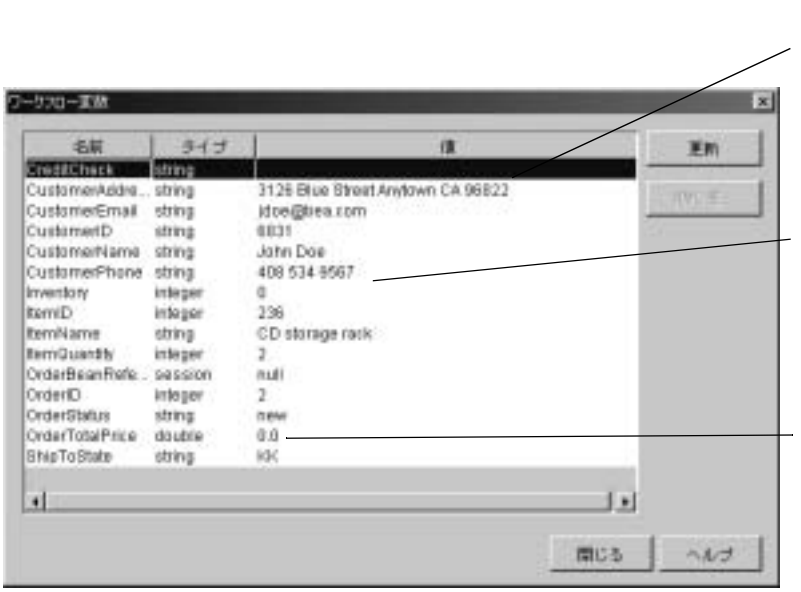
3. インスタンス リストで、[Order 2] をダブルクリックします。[ワークフロー ステータス] ウィンドウが表示されます。

図 7-11 [ワークフロー ステータス Order Processing] ウィンドウ



4. [変数] ボタンをクリックします。ワークフロー変数の現在の設定を表示します。[ワークフロー変数] ウィンドウが表示されます。

図 7-12 [ワークフロー変数] ウィンドウ

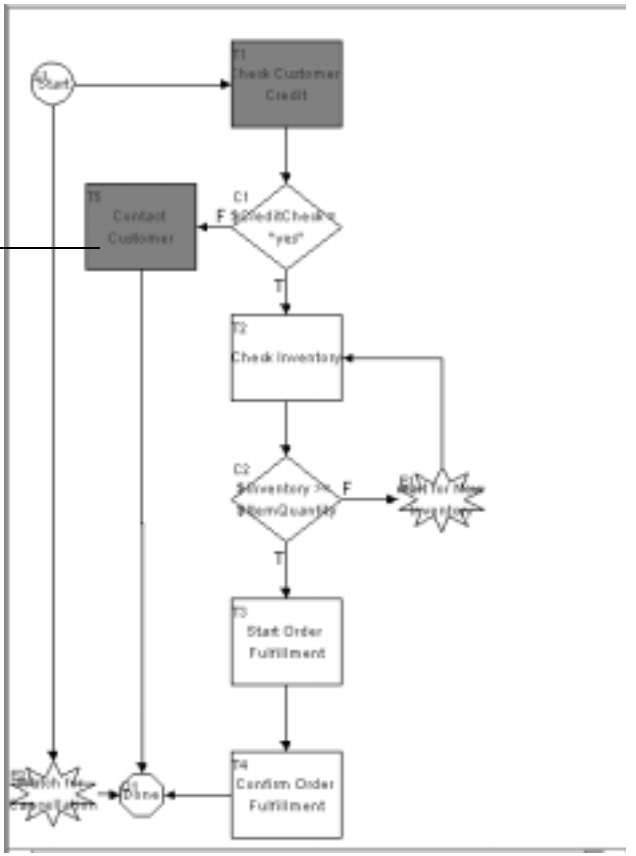


5. [閉じる] をクリックして、[ワークフロー変数] ウィンドウを終了します。
6. [ワークフロー ステータス] ウィンドウを終了します。
7. [ワークリスト] に切り替えます。
8. [admin] ユーザ タブで [Check Customer Credit] タスクをダブルクリックして、そのタスクを実行します。
9. メッセージ ボックスが表示されたら [いいえ] をクリックします。これでフローは、CustomerService ロール（admin はそのメンバー）に割り当てられた Contract Customer タスクに進みます。
10. [CustomerService] タブをクリックします。[コメント] カラムには、4-31 ページの「タスクとワークフローのコメントの追加 : Contact Customer タスクの定義」で定義した式が表示されます。コメントには、そのタスクを実行する前にユーザが従うべき指示が表示されます。

図 7-13　タスクのコメント

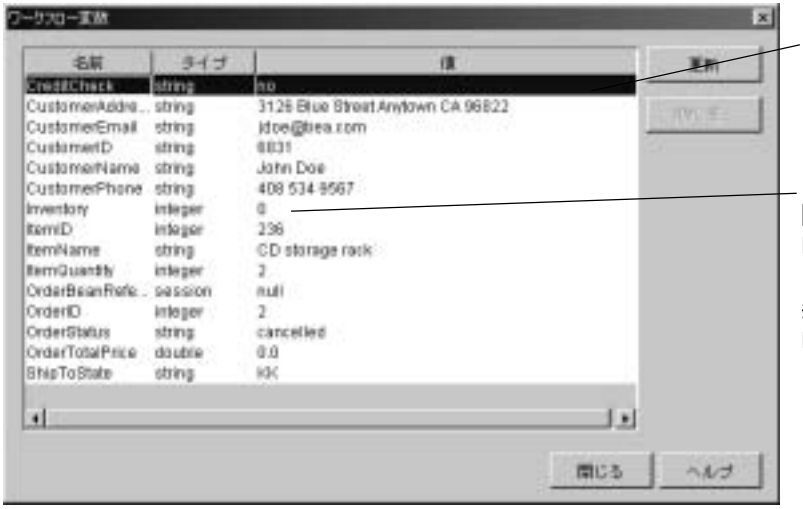
11. タスクをダブルクリックして、このタスクを実行します。これでフローは終了します。Studio に戻り、ワークフロー インスタンスに表示された結果を確認します。
12. Studio の [ワークフロー インスタンス] ダイアログ ボックスで、[最新の情報に更新] をクリックします。ワークフローが終了したことが表示されています。ここには 4-66 ページの「ワークフローのコメントを設定する」で定義したワークフローのコメント Order cancelled が含まれています。

13. [Order 2] インスタンスをダブルクリックします。[ワークフロー ステータス] ウィンドウが表示されます。Contact Customer タスクはグレー表示になっています。



グレーは、タスクが完了とマークされたことを示めす。

14. [変数] をクリックして、[ワークフロー変数] ダイアログ ボックスを表示します。



15. [閉じる] をクリックして、[ワークフロー変数] ウィンドウを終了します。

16. [ワークフロー ステータス] ウィンドウを終了します。

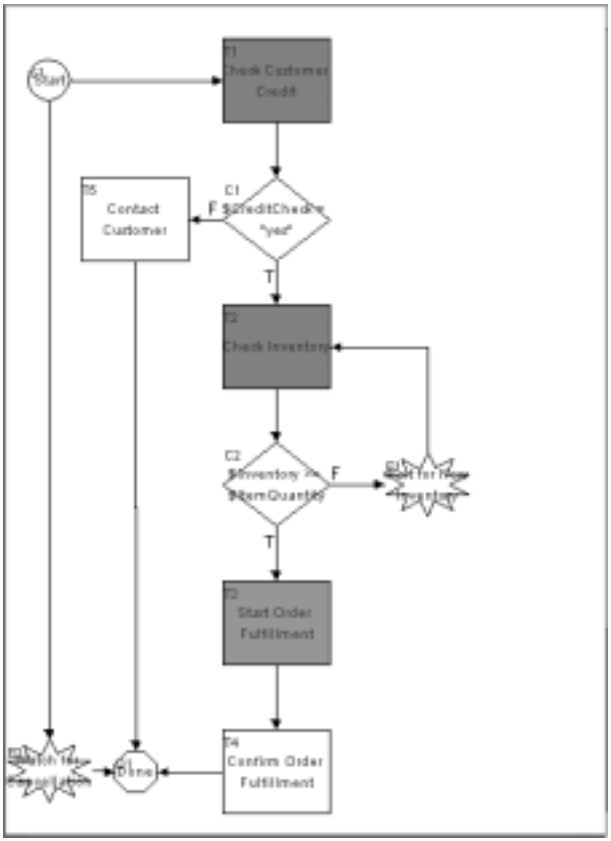
ここで、3 回目のワークフローの繰り返しを行い、Credit Check タスクに対して はい を応答後に、終了したワークフローのインスタンスをモニタします。

Order Processing および Order Fulfillment ワークフローの再実行とモニタの手順は、次のとおりです。

1. [ワークリスト] で、[admin] ユーザ タブを選択します。次に Start Order Processing タスクをダブルクリックします。Order Processing ワークフローが開始します。ここでは Order 3 というラベルが表示されています。
2. [admin] タブを選択します。次に [Check Customer Credit] タスクをダブルクリックして、そのタスクを実行します。[Credit Check] メッセージ ボックスが再び表示されます
3. 今回は、[はい] をクリックします。ワークフローは Check Inventory タスクに進みます。
4. ここで、Studio に切り替えます。

5. [ワークフロー インスタンス] ダイアログ ボックスで、[最新の情報に更新] をクリックします。新しいインスタンスが表示され、今回は Order 3 となっています。

6. [Order 3] インスタンスをダブルクリックして、[ワークフロー ステータス] ウィンドウを呼び出します。今回のウィンドウでは、Check Customer Credit タスクと Check Inventory タスクはグレー表示されています。



7. ここで、[Vars] をクリックして、[ワークフロー変数] ウィンドウを表示します。

今度は CreditCheck と Inventory の変数が設定されている。

8. [閉じる] をクリックして、[ワークフロー変数] ウィンドウを終了します。

   [Inventory] 変数（536）が、[ItemQuantity] 変数（2）より大きいので、今度は Order Fulfillment ワークフローが開始され、そのインスタンスをモニタできます。

9. フォルダ ツリーで、Order Fulfillment のテンプレートを右クリックします。表示されるポップアップ メニューから [インスタンス] を選択します。ワークフローの 2 つのインスタンスが表示された [ワークフロー インスタンス Order Fulfillment] ダイアログ ボックスが表示されます。

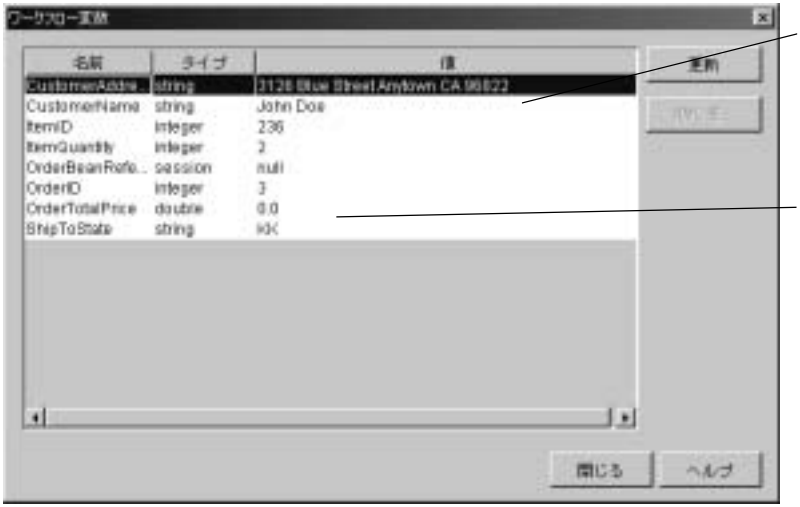
図 7-14　[ワークフロー インスタンス Order Fulfillment] ダイアログ ボックス

10. [Order 3] をダブルクリックします。[ワークフロー ステータス] ウィンドウが表示されます。

11. 今回は、[List Instances] ボタン  をクリックします。リスト ビューでステータスを確認します。

図 7-15　[ワークフロー ステータス Order Fulfillment] ウィンドウ

12. [変数] をクリックして、[ワークフロー変数] ウィンドウを表示します。



13. [閉じる] をクリックして、[ワークフロー変数] ウィンドウを終了します。
14. [ワークリスト] ウィンドウに戻ります。
15. [Shipping] ロール タブを選択します。次に [Shipping] タスクをダブルクリックして、そのタスクを実行します。
16. [Accounting] タブを選択し、次に [Generate Invoice] タスクをダブルクリックします。[Error Warning] メッセージ ボックスに、CA などの有効な州の略称を入力します。
17. Studio に戻ります。
18. [ワークフロー ステータス Order Fulfillment] ウィンドウで [Instance Refresh] ボタン  をクリックします。

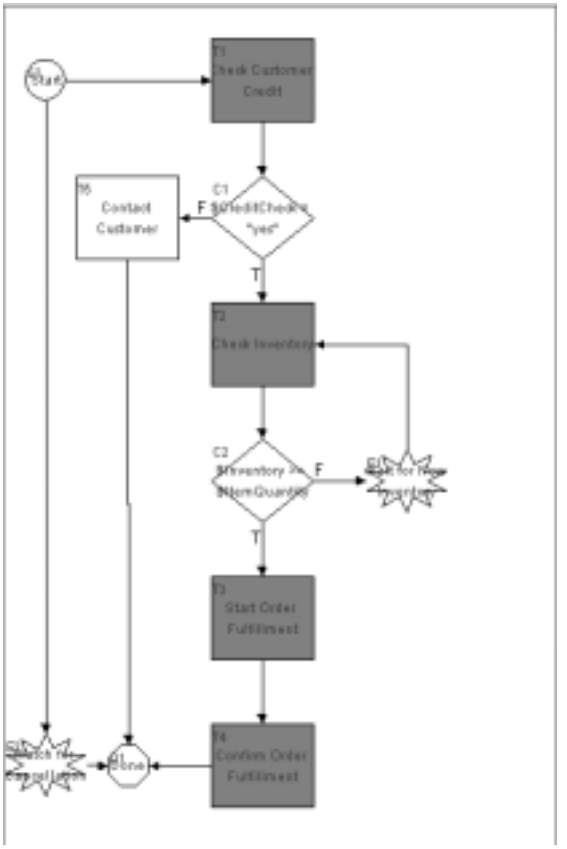
19. ここで、[変数] をクリックして、[ワークフロー変数] ウィンドウを表示します。

20. [閉じる] をクリックして、ウィンドウを終了します。

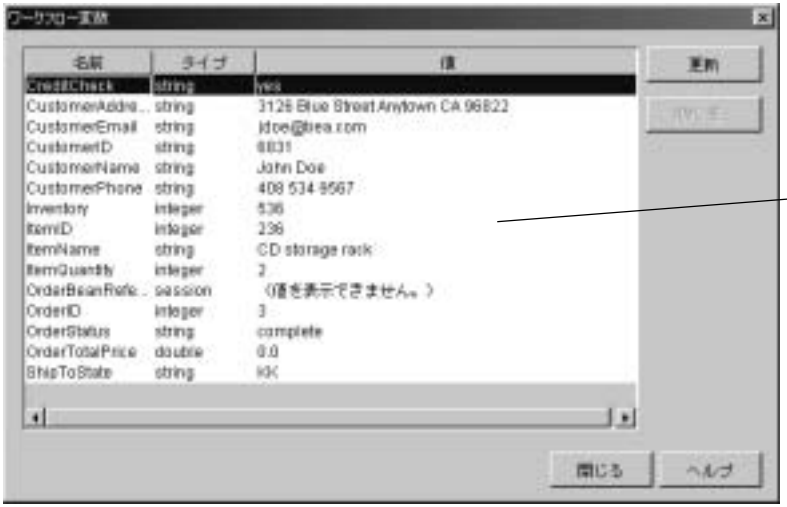
21. [ワークフロー ステータス Order Fulfillment] ウィンドウを終了します。

22. [ワークフロー インスタンス Order Processing] ダイアログ ボックスで、[最新の情報に更新] をクリックします。ウィンドウは、ワークフローが終了したことを示しています。

23. [Order 3] をダブルクリックして、[ワークフロー ステータス] ウィンドウを確認します。ウィンドウは、ワークフローが終了したことを示しています。



24. [変数] をクリックします。終了したワークフローの変数が表示されます。



25. [ワークフロー変数] ウィンドウを終了して、次に [ワークフロー ステータス] ウィンドウも終了します。

26. 他に開いているウィンドウがあれば、それも終了します。

これで、Business Process Management のチュートリアルは終了です。