BEA WebLogic Integration™

B2B Integration
ワークフローの作成

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

B2B Integration ワークフローの作成

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

3. 協調的ワークフローの開始

4. ビジネス メッセージの使い方

5. ビジネス メッセージの送受信

6. 協調的ワークフローの終了

索引

このマニュアルの内容

このマニュアルでは、コラボレーション アグリーメントで定義された会話の基本単位である協調的ワークフローを作成する方法について説明します。協調的ワークフローを作成するには、B2B Integration プラグイン機能を使用します。このプラグインを使用すると、WebLogic Integration Studio の機能を拡張して、会話定義にバインドされる協調的ワークフローを作成できます。

注意： ebXML ビジネス プロトコルは、このドキュメントで説明しているものよりシンプルな会話モデルを採用しています。このため、使用するのが ebXML ビジネス プロトコルのみであれば、このドキュメントではなく ebXML ドキュメントを直接参照されることをお勧めします。ebXML ビジネス プロトコルのワークフロー作成に関する情報については、『*B2B Integration ebXML の実装*』の「ワークフローで ebXML を使用する」を参照してください。

このマニュアルの内容は以下のとおりです。

- 第 1 章「ワークフローの作成について」では、B2B Integration プラグインについて概説します。また、WebLogic Integration Studio の概念および Studio を使用して会話定義をまとめる方法についても説明します。
- 第 2 章「ワークフロー テンプレートの会話プロパティの定義」では、会話定義のロールを実装する協調的ワークフローのテンプレートを定義する方法について説明します。
- 第 3 章「協調的ワークフローの開始」では、協調的ワークフローの開始プロパティをコンフィグレーションして、実行時に協調的ワークフローを開始する方法について説明します。
- 第 4 章「ビジネス メッセージの使い方」では、トレーディング パートナ間で交換されるビジネス ドキュメントのデータの作成および抽出を開始するアクションのプロパティを、ワークフローのタスクとイベントに割り当てる方法について説明します。またこの章では、Message Manipulator API に付属のメッセージ マニピュレータ クラスを操作して、ビジネス メッセージの内容を作成および処理する方法についても説明します。

- 第 5 章「ビジネス メッセージの送受信」では、ビジネス メッセージの送受信を開始するアクションのプロパティを、ワークフローのタスクとイベントに割り当てる方法について説明します。
- 第 6 章「協調的ワークフローの終了」では、会話終了の処理をする方法および協調的ワークフローを終了する方法について説明します。

対象読者

このマニュアルは主に、次のユーザを対象としています。

- WebLogic Integration Studio を使用して、WebLogic Integration 環境での会話用に定義されたトレーディング パートナのロールを実装する協調的ワークフローを設計するビジネス アナリスト
- WebLogic Integration アプリケーションを設定および管理するシステム管理者

WebLogic Integration のアーキテクチャの概要については、『*WebLogic Integration 入門*』を参照してください。

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、WebLogic Integration マニュアル CD にあります。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。

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

Business Process Management の詳細については、以下のマニュアルを参照してください。

- *WebLogic Integration BPM ユーザーズ ガイド*
- *WebLogic Integration Studio 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 Worklist ユーザーズ 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、WebLogic Integration 7.0 リリースのドキュメントをご使用の旨をお書き添えください。

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
太字	用語集で定義されている用語を示す。
〔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`

太字の等幅テキスト	コード内の重要な箇所を示す。 *例* `void` **`commit`** `( )`
斜体の等幅テキスト	コード内の変数を示す。 *例* `String` *`expr`*

表記法	適用
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。
[ ]	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　ワークフローの作成について

以下の節では、B2B Integration アプリケーションでワークフローを使用するための主要な概念を説明します。

- 続ける前に
- アーキテクチャの概要
- 主要な概念
- 実行時の要件
- ワークフロー統合タスクの概略

続ける前に

このマニュアルの先へ進む前に、次の点に注意してください。

- RosettaNet プロトコルに基づく会話を実装する場合は、『*B2B Integration RosettaNet の実装*』で RosettaNet 1.1 および 2.0 プロトコルに基づくワークフローと会話の実装方法について参照してください。
- ebXML ビジネス プロトコルはこのドキュメントで説明しているものよりシンプルな会話モデルを採用しています。このため、使用するのが ebXML ビジネス プロトコルのみであれば、このドキュメントではなく ebXML ドキュメントを直接参照されることをお勧めします。ebXML ビジネス プロトコルのワークフロー作成に関する情報については、『*B2B Integration ebXML の実装*』の「ワークフローで ebXML を使用する」を参照してください。
- このドキュメントで説明する B2B 機能でサポートしている XOCP ビジネス プロトコルは、WebLogic Integration の本リリースより廃止されています。XOCP の代替となるビジネス プロトコル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。

アーキテクチャの概要

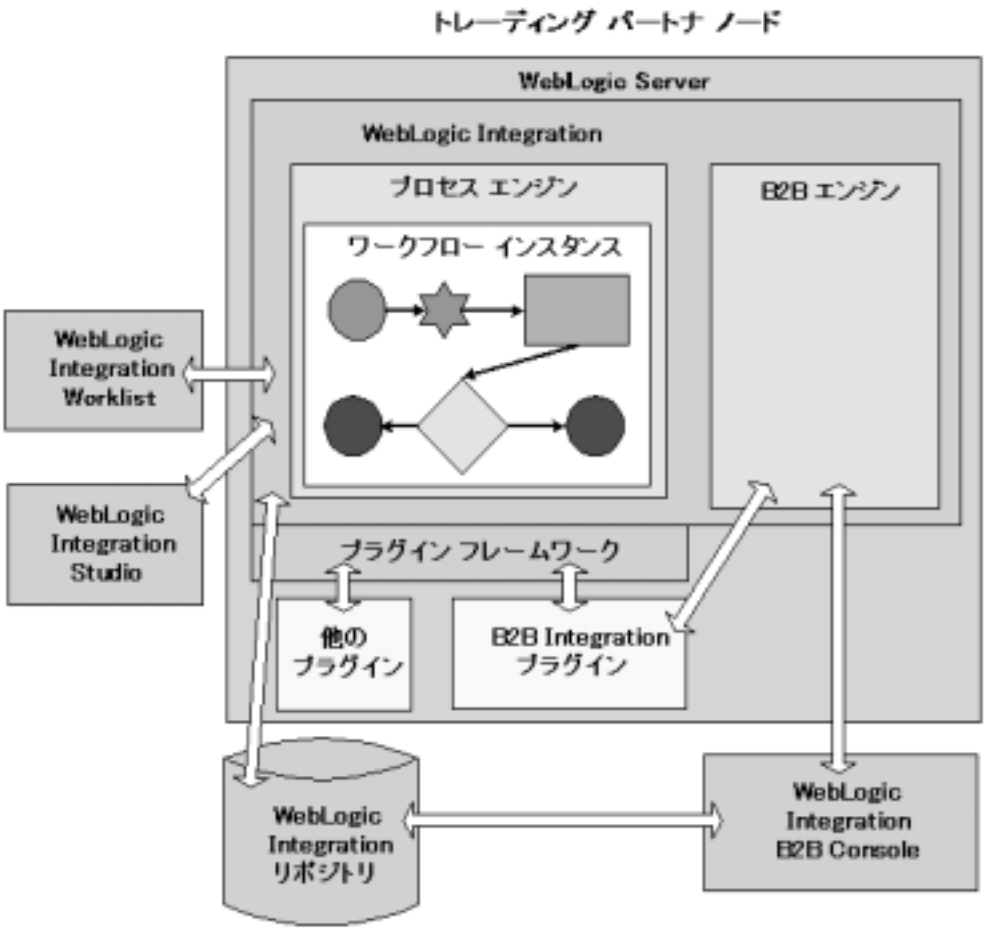
会話を定義する上での基本的なステップは、その会話の各ロールを実行するワークフローの作成です。B2B Integration コンポーネントは、そのような*協調的ワークフロー*（*パブリック ワークフロー*とも呼ぶ）を作成できるプラグインを備えています。WebLogic Integration の B2B Integration コンポーネントと Business Process Management（BPM）コンポーネントを一緒に使用すると、以下のものを利用できます。

- ワークフロー（プロセス モデル）を設計するためのグラフィカルな設計ツール（ビジネス コラボレーションの開発時間を短縮する）
- ワークフローを実行するための実行時プロセス エンジン
- プロセス モニタ機能

B2B Integration プラグインを使用してワークフローを作成する場合は、設計タスクと管理タスクが不可欠であり、プログラミング タスクを行わなければならない場合もあります。

アーキテクチャの構成要素

次の図は、WebLogic Integration のアーキテクチャでどのように B2B Integration プラグインが組み込まれるのかを示しています。



WebLogic Integration コンポーネント

WebLogic Integration のアーキテクチャには、協調的ワークフローの作成と実行をサポートする以下のコンポーネントが含まれています。

表 1-1 WebLogic Integration コンポーネント

コンポーネント	説明
プロセス エンジン	ワークフローを実行および管理し、ワークフロー インスタンスをトラッキングする実行時のコントローラおよびワークフロー エンジン
ワークフロー インスタンス	WebLogic Integration リポジトリでコンフィグレーションされた会話定義のロールを実装する、ワークフロー テンプレート定義のアクティブなインスタンス。ワークフロー インスタンスはワークフロー テンプレートの定義が実行時にインスタンス化されたときに作成される。
WebLogic Integration Studio	設計時にワークフローを定義するため、および実行時にアクティブなワークフローをモニタするために使用するクライアント アプリケーション
WebLogic Integration Worklist	ユーザまたはそのユーザが属するロールに割り当てられているタスクを表示および実行するために使用するクライアント アプリケーション。たとえば、他のユーザへのタスクの再割り当て、タスクに完了のマークを付ける、タスクの完了のマークを外す、ワークフロー ステータスの表示、手動によるワークフローの開始など。
WebLogic Integration リポジトリ	以下のものが格納されるデータベース ■ ワークフローのテンプレートとインスタンス ■ 実行時に WebLogic Integration で使用される会話定義情報（会話、パーティ、会話ロール、プロトコルのバインディングの情報など） リポジトリはトレーディング パートナ ノードにローカルで配置することも、WebLogic Integration ソフトウェアからアクセス可能な別のノードに配置することもできる。リポジトリは、1 つのオーガニゼーション内の 1 つのトレーディング パートナから、または複数のオーガニゼーションの複数のトレーディング パートナからアクセスできるようにデプロイできる。

表 1-1 WebLogic Integration コンポーネント (続き)

コンポーネント	説明
B2B エンジン	実行時にトレーディング パートナ間のメッセージを処理およびルーティングするソフトウェア
WebLogic Integration B2B Console	コラボレーション アグリーメント、会話、配信チャネル、ドキュメント交換、トレーディング パートナなどを作成、コンフィグレーション、管理、およびモニタできるブラウザ ベースのコンポーネント
WebLogic Integration プラグイン フレームワーク	特定の BEA ソフトウェア（BEA Java Application-to-Mainframe（JAM）製品や、WebLogic Integration の B2B Integration、Data Integration、および Application Integration コンポーネント）およびユーザ記述ソフトウェアで WebLogic Integration Studio とプロセス エンジンの機能を拡張できるようにするコンポーネント
B2B Integration プラグイン	以下の機能を持つ B2B 統合プラグイン ■ Studio の機能を拡張し、協調的ワークフロー（次のビジネス プロトコルのいずれかに基づいて会話の個々のロールを実装するワークフロー）を作成するためのワークフローのノード、アクション、イベントといった細目を指定できるようにする。 — XOCP — RosettaNet 2.0 — RosettaNet 1.1 — ebXML ■ プロセス エンジンと会話フローを統合する。
他のプラグイン	WebLogic Integration プラグイン フレームワークでは、追加のプラグインを組み込むことができる（上の WebLogic Integration プラグイン フレームワークの説明を参照）。

Studio、Worklist、プロセス エンジン、プラグイン フレームワークの概要については、『*WebLogic Integration Studio ユーザーズ ガイド*』の「WebLogic Integration Studio の概要」を参照してください。

主要な概念

この節では、B2B Integration アプリケーションで協調的ワークフローを使用する前に理解しておく必要のある主要な概念を説明します。

ワークフロー、協調的ワークフロー、ワークフロー テンプレート、およびワークフロー テンプレート定義

この節では、Business Process Management（BPM）に関する以下の主要な概念を説明します。

- WebLogic Integration は、プロセス エンジンでワークフロー自動化ツールを提供します。*ワークフロー*とは、ビジネス プロセスのことです。ワークフローの自動化は、全体的あるいは部分的なビジネス プロセスの自動化を意味します。ワークフローの自動化では、コンピュータが大部分の作業を実行できるようにし、人間は例外を処理するだけでいいようにするインテリジェントなビジネス ルールに従って、あらゆるタイプの情報が適切なときに適切な参加者に渡されます。
- *協調的ワークフロー*は、会話定義のトレーディング パートナのロールを実装するワークフローです。会話定義では、2 つ以上のロールを定義できます。各ロールは、協調的ワークフローと関連付けられます。

  パブリック プロセスを実装する協調的ワークフローは、通常は*パブリック ワークフロー*と呼ばれます。厳密に言うと、協調的ワークフローはそのワークフローがプライベート プロセスを実行するようにカスタマイズされていない場合のみパブリック ワークフローとなります。ただし、複雑にならないように、このマニュアルでは協調的ワークフローとパブリック ワークフローは互いに交換可能な用語であると見なします。Studio のユーザ インタフェースでは、B2B Integration プラグインについて言及するテキストではパブリック ワークフローという用語のみを使用します。たとえば、協調的ワークフローを開始するように親ワークフローで定義したアクションのことは Start Public Workflow と呼びます。

パブリックとプライベートのビジネス プロセスの詳細については、『*B2B Integration 入門*』の「概要」を参照してください。

- *ワークフロー テンプレート*は、WebLogic Integration Studio のフォルダ（コンテナ）です。このワークフロー テンプレートはワークフローを表し、意味のある名前（Order Processing や Billing など）を持ちます。ワークフロー テンプレートは、その実装のさまざまな定義（バージョン）を 1 つにまとめます。それらの定義はワークフロー テンプレート定義と呼ばれます。さらに、ワークフロー テンプレートではどのオーガニゼーションがそのワークフロー テンプレート定義を使用できるのかも管理します。
- *ワークフロー テンプレート定義*とは、ワークフローの定義（バージョン）のことです。それらの定義は、有効および期限切れになる日付によって区別されます。設計段階では、Studio を使用して、ワークフロー テンプレート定義を会話のロール（*バイヤやセラー*など）にバインドするプロパティを定義できます。実行時には、プロセス エンジンがワークフロー テンプレート定義のインスタンス（セッション）を開始し、最も有効な（または現在のアクティブな）定義を選択します。

これらのコンセプトに関する詳細については、『*WebLogic Integration Studio ユーザーズ ガイド*』の「WebLogic Integration Studio の概要」を参照してください。

会話とビジネス メッセージ

この節では、B2B Integration に関する以下の主要な概念を説明します。

- *会話*とは、会話定義で定義されているトレーディング パートナ間の一連のビジネス メッセージ交換のことです。トレーディング パートナがビジネス メッセージを受信すると、不定数のバックエンド プロセスが発生します。
- *ビジネス メッセージ*は、トレーディング パートナ間の通信の基本単位です。ビジネス メッセージは、1 つまたは複数のビジネス ドキュメントと添付ファイル（オプション）で構成されます。

  *ビジネス ドキュメント*は、ビジネス メッセージ中の XML ベースのペイロード部分です。XML ビジネス ドキュメントは、通常は対応するドキュメント定義と関連付けられます。

ビジネス メッセージには、1 つまたは複数の*添付ファイル*を付けることができます。添付ファイルは XML フォーマットでなくてもかまいません。たとえば、添付ファイルはバイナリ フォーマットで作成できます。

送信ビジネス メッセージを作成するため、または受信ビジネス メッセージを処理するために、ワークフローでは Compose Business Message、Extract Business Message Parts、または Manipulate Business Message といったアクションを適切に使用します。

以上の概念の詳細については、『*B2B Integration 入門*』を参照してください。

開始者と参加者

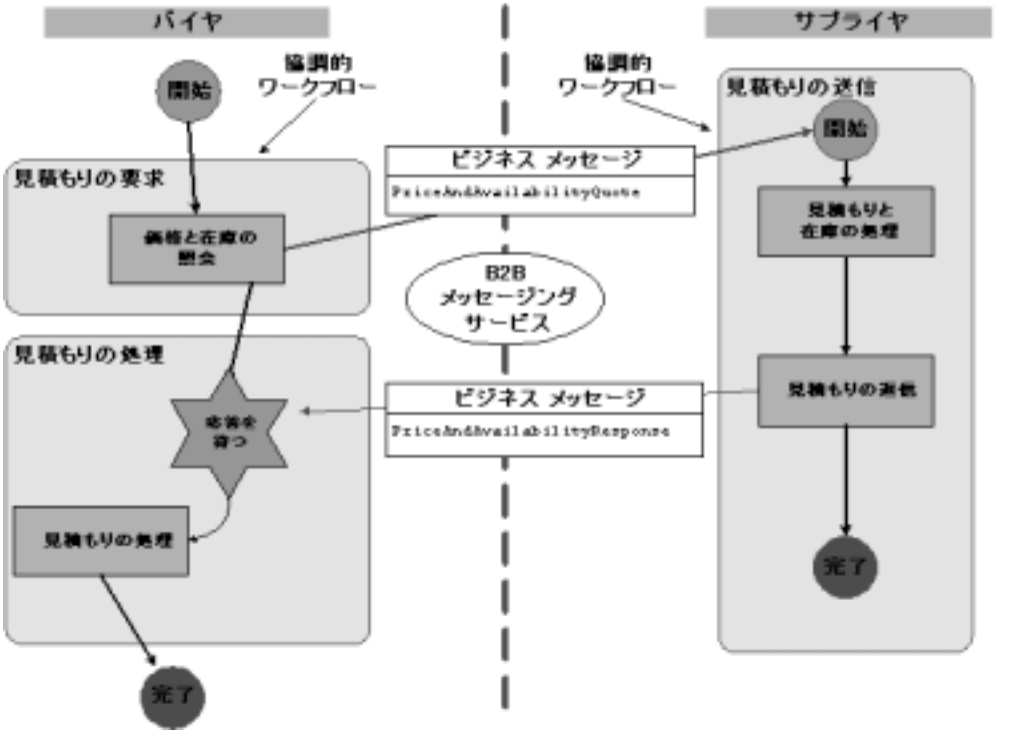
会話には、会話を開始する*開始者*と、開始された会話に参加する*参加者*が関与します。それぞれには、異なる種類の協調的ワークフローが必要です。

表 1-2 ワークフローの種類

ワークフローの種類	説明
会話開始者ワークフロー	会話プロパティおよび非ビジネス メッセージ開始プロパティを持つように定義される。このタイプのワークフローによって会話が開始および終了される。
会話参加者ワークフロー	会話プロパティおよびビジネス メッセージ開始プロパティを持つように定義される。このタイプのワークフローは会話に参加し、会話から抜けることができるが、会話を開始または終了することはできない。

会話のコンテキストにおいて、これらの 2 種類のワークフローは連動します。たとえば、バイヤが多数のサプライヤから入札を受け付けたいと考えているとします。その会話は、次の図のように進行すると考えられます。

図 1-1　ワークフローが 2 つのビジネス コラボレーション例



1. Studio で、バイヤ（会話を開始するトレーディング パートナ）が協調的ワークフローのインスタンス（ビジネス コラボレーション開始者ワークフロー）を開始します。このワークフローは、`PriceAndAvailabilityQuote` ビジネス メッセージ（XML ドキュメント形式の入札要求が含まれる）を作成して、B2B エンジン経由で 1 つまたは複数の有資格セラー（ビジネス プロトコルによって異なる）に送信し、応答を待ちます。

   注意： ワークフローは、さまざまな方法で開始できます。たとえば、プログラム的に開始することや、別個のローカルのワークフローから開始することも可能です。

2. 各有資格サプライヤ（参加トレーディング パートナ）がビジネス メッセージを受信します。そのビジネス メッセージによってサプライヤのノードでワークフロー（会話参加者ワークフロー）のインスタンスが開始されます。

サプライヤ ロールを実装する協調的ワークフローが、受信した入札要求を処理し、入札に参加するかどうかを判断します。参加する場合は、`PriceAndAvailabilityResponse` ビジネス メッセージ（XML ドキュメント形式の入札応答が含まれる）を作成して送信し、ワークフローを終了します。このワークフロー インスタンスにはビジネス メッセージ開始プロパティがあります。従って、このワークフローはバイヤの `PriceAndAvailabilityQuote` ビジネス メッセージがサプライヤのノードで受信された時に開始されます。

注意： サプライヤ ロールを実装する協調的ワークフローは、会話プロパティとビジネス メッセージ開始プロパティで定義されます。

3. バイヤ側で、ワークフローがすべての有資格サプライヤからの入札応答を受信し、入札の結果を処理して、会話を終了します。

ビジネス メッセージの送受信

トレーディング パートナ間でビジネス メッセージを交換するときには、通常は開始者と参加者の両方のワークフローがビジネス メッセージを送受信します。

ワークフローのどの部分でビジネス メッセージが送信され、どの部分でビジネス メッセージが受信されるのかを覚えておくことが重要です。たとえば、バイヤからセラーに入札要求（ビジネス メッセージ）が送信される場合があります。その場合は、バイヤ ワークフローがビジネス メッセージを送信し、セラー ワークフローがそのメッセージを受信します。セラーが入札（別のビジネス メッセージ）で要求に応答するときには、それらのロールが逆転します。つまり、セラー ワークフローが送信側になり、バイヤ ワークフローが受信側になります。

設計タスクは、ワークフローでビジネス メッセージが送信されるのかそれとも受信されるのかによって異なります。ただし、両方の場合において、ワークフロー テンプレート定義で特定のプロパティを定義する必要があります。B2B Integration プラグインでは、ビジネス メッセージを扱うための以下の 2 つの手段が提供されます。

- ユーザ記述アプリケーション コードがなくても操作可能なビジネス メッセージを作成するか、そのようなメッセージから情報を抽出できるワークフロー アクション
- `com.bea.b2b.wlpi.MessageManipulator` インタフェースを実装してビジネス メッセージを作成するか、ビジネス メッセージから情報を抽出するアプ

リケーション コードを起動するワークフロー アクション（複雑なメッセージ処理が要求される状況で使用）

詳細については、第 4 章「ビジネス メッセージの使い方」を参照してください。

実行時の要件

実行時にメッセージを交換するためには、まず以下の要件を確実に満たしておかなければなりません。

- WebLogic Integration がインストールおよびコンフィグレーションされている（1-12 ページの「管理タスク」を参照）
- ワークフローが定義されて、会話定義にリンクされている（1-13 ページの「設計タスク」を参照）
- メッセージを操作したり、会話開始者ワークフローを開始したりするためのアプリケーション コードが記述済みで、テストも済んでいる（1-15 ページの「プログラミング タスク」を参照）
- すべてのトレーディング パートナについて、WebLogic Integration がコンフィグレーションされて実行されている
- 各トレーディング パートナについて、関連するすべてのワークフローがアクティブであり、WebLogic Integration リポジトリに格納されている
  - 会話を開始するトレーディング パートナについては、会話開始者ワークフローがアクティブであり、非ビジネス メッセージ開始プロパティで定義されている必要があります。会話開始者ワークフローは、そのワークフローを開始するアプリケーションの起動を待ちます。
  - すべての参加トレーディング パートナについて、会話参加者ワークフローがアクティブであり、ビジネス メッセージ開始プロパティで定義されている必要があります。会話参加者ワークフローは、会話の最初のビジネス メッセージの受信を待ちます。

ワークフロー統合タスクの概略

協調的ワークフローを使用して WebLogic Integration でビジネス メッセージを交換するには、管理、設計、およびプログラミングのタスクを行う必要があります。

管理タスク

協調的ワークフローを作成するには、以下の管理タスクを行う必要があります。

1. 『*BEA WebLogic Platform インストール ガイド*』の説明に従って WebLogic Integration をインストールします。

   注意： WebLogic Integration の Samples レルムは、次の手順で説明されているとおりに事前にコンフィグレーションされています。

2. 『*WebLogic Integration の起動、停止およびカスタマイズ*』の説明に従って WebLogic Integration をコンフィグレーションします。

   注意： WebLogic Integration のサンプルを実行する前に、必ず WebLogic Integration をインストールしてください。サンプルを実行することによって、インストールが成功しているかどうかを確認できます。サンプル アプリケーションの実行の詳細については、『*WebLogic Integration チュートリアル*』および『*B2B サンプルの使い方*』を参照してください。

3. B2B Console を使用して、コラボレーション アグリーメント、配信チャネル、トレーディング パートナ、ドキュメント交換など、WebLogic Integration リポジトリで必要なエンティティを作成およびコンフィグレーションします。詳細については、『*B2B Integration 管理ガイド*』を参照してください。

   注意： すべての協調的ワークフロー テンプレート定義では、会話定義が必要です。

4. 『*WebLogic Integration Studio ユーザーズ ガイド*』の「データの管理」の説明に従って、Studio を使用して WebLogic Integration リポジトリでオーガニゼーション、ユーザ、およびロールを指定します。

設計タスク

協調的ワークフローを作成するには、Studio で以下の設計タスクを行う必要があります。

1. 以下のいずれかの方法で、会話定義の各ロールを実装する協調的ワークフローを作成および設計します。
   - ゼロからワークフローを作成できます。詳細については、『*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」を参照してください。
   - WebLogic Integration の前のバージョンで作成したワークフローをインポートすることもできます。詳細については、『*WebLogic Integration 移行ガイド*』を参照してください。
2. 標準のワークフロー プロパティを定義するだけでなく、B2B Integration プラグインを使用してワークフローを会話にリンクするプロパティを定義することも必要です。この手順の以降のタスクはワークフローの会話への統合に関するものです。
3. 各ワークフロー テンプレートについて、次のように会話プロパティを指定します。
   - ワークフロー テンプレートを WebLogic Integration リポジトリの会話定義のロールに明示的にリンクします。詳細については、2-6 ページの「テンプレートと会話のリンク」を参照してください。
   - 必要に応じて、他の会話プロパティを指定します。詳細については、2-9 ページの「ワークフロー テンプレートでの XOCP メッセージ配信のサービス品質の定義（非推奨）」を参照してください。
4. 各ワークフロー テンプレート定義について、ワークフローのタイプに従って開始アクションを定義します。
   - 会話開始者ワークフローの場合は、3-3 ページの「会話開始者ワークフローの開始」の説明に従って非ビジネス メッセージ開始プロパティを定義します。
   - 会話参加者ワークフローの場合は、3-6 ページの「会話参加者ワークフローの開始ノードの定義」の説明に従ってビジネス メッセージ開始プロパティを定義します。

5. 各ワークフロー テンプレート定義について、ワークフローがどのように終わるのかを定義します。そのためには、完了シェイプを追加し、そのプロパティを定義します。6-1 ページの「会話開始者ワークフローの終了の定義」で説明されているように、通常、会話開始者ワークフローは完了ノードで会話を終了します。

   会話参加者ワークフローには、カスタム完了ノードの定義はありません。

6. ワークフロー、またはそのワークフローと会話するアプリケーションで使用する入力変数または出力変数をワークフロー テンプレート定義で定義します。詳細については、2-16 ページの「ワークフロー変数の使用について」を参照してください。

7. 各ワークフロー テンプレート定義について、ビジネス メッセージの処理方法を定義します。

   - すべての協調的ワークフローについて、2-16 ページの「ワークフロー変数の使用について」の説明に従ってビジネス メッセージを格納するワークフロー変数を定義します。
   - すべての協調的ワークフローについて、送信するビジネス メッセージを作成するか、受信したビジネス メッセージを処理するかたちでビジネス メッセージを操作するタスク ノードを定義します。詳細については、第 4 章「ビジネス メッセージの使い方」を参照してください。
   - ビジネス メッセージを送信するワークフローでは、5-1 ページの「ビジネス メッセージを送信するワークフローの定義」の説明に従って Send Business Message アクションを定義します。
   - 会話参加者ワークフローでは、開始ノードをビジネス メッセージの開始として定義します。そうすることで、会話開始者ワークフローから最初のビジネス メッセージを受信した時点で協調的ワークフローが開始されます。詳細については、3-6 ページの「会話参加者ワークフローの開始ノードの定義」を参照してください。さらに、受信ビジネス メッセージを処理するための関連するビジネス メッセージ操作アクションを追加します。
   - 会話開始者ワークフローまたは会話参加者ワークフローで受信される最初のビジネス メッセージ以外のビジネス メッセージについては、5-29 ページの「Business Message Receive イベントの定義」の説明に従って受信ビジネス メッセージ会話イベントを定義します。さらに、受信ビジネス メッセージを処理するための関連するビジネス メッセージ操作アクションを追加する必要があります。

8. 会話の終了方法を定義します。終了の方法はプロトコルによって異なります。
   - XOCP ベースの会話の会話開始者ワークフローでは、通常は会話を終了するカスタム完了ノードを定義します。XOCP ベースの会話開始者ワークフローからコラボレーション終了イベントが仲介機能（ルーティング プロキシ）に送信されると、仲介機能からすべての会話参加者に会話終了イベントが送信されます。
   - RosettaNet ベースのワークフローでは、会話を終了する独自の手段を定義します。詳細については、『*B2B Integration RosettaNet の実装*』を参照してください。
   - XOCP ベースの会話の会話参加者ワークフローでは、必要に応じて会話終了イベントをリスンするイベント ノードを定義できます。そのようなイベント ノードを持つことで、ワークフローでは制御を完了ノードに渡す前にハウスキーピング処理を実行できます。会話参加者ワークフローで会話終了リスナ イベント ノードを定義する方法については、6-5 ページの「会話参加者ワークフローの終了の定義」を参照してください。

注意： ワークフローは、B2B の機能と統合されていない場合でも WebLogic Integration 環境で実行できます。たとえば、B2B 機能と統合するための特別な処理をせずに Studio で作成したワークフローを実行できます。

プログラミング タスク

プログラミング タスクは、ワークフローを利用する各アプリケーションの特定のニーズによって異なります。以下のタスクを行う必要があります。

- 親ワークフローで定義された Start Public Workflow アクションを通じてインスタンス化されるのではない会話開始者ワークフローでは、ワークフローを開始するアプリケーションを記述します。このアプリケーションでは、`com.bea.b2b.wlpi.WorkflowInstance` クラスを使用して、ワークフローがバインドされている会話に関する以下のような情報を渡します。
  - 会話定義の名前とバージョン
  - 開始者の会話ロール
  - 会話に参加する 1 つまたは複数のパーティのアイデンティティ

  この情報を利用することで、ワークフローは会話で必要とされるユニークなコラボレーション アグリーメントとバインドできます。アプリケーションで

は他のタスク（ワークフロー変数への値の割り当てなど）も実行できますが、最低限、ワークフローがバインドされる会話の情報を渡さなければなりません。詳細については、3-21 ページの「会話開始者ワークフローを開始するアプリケーションの開発」を参照してください。

注意： ワークフローをサブワークフローとして開始する方法については、3-10 ページの「Start Public Workflow アクションの定義」を参照してください。

- 複雑なメッセージ処理を必要とする会話開始者ワークフローと会話参加者ワークフローの両方について、`com.bea.b2b.wlpi.MessageManipulator` インタフェースを実装するアプリケーション コードを記述します。このクラスを使用して、ワークフローから起動されたアプリケーションではワークフローが送受信するビジネス メッセージで複雑な操作、抽出、受信、および送信プロセスを実行できます。詳細については、4-16 ページの「複雑なメッセージ操作（非推奨）」を参照してください。

2　ワークフロー テンプレートの会話プロパティの定義

以下の節では、会話定義のロールを実装するワークフローのテンプレートおよびテンプレート定義の定義方法を説明します。

- WebLogic Integration がサポートしているビジネス プロトコル
- B2B Integration プラグインを使用したテンプレートの定義について
- テンプレートとテンプレート定義について
- ワークフロー テンプレートの作成
- テンプレートと会話のリンク
- テンプレートとコラボレーション プロトコルのリンク
- メッセージのロギング
- ワークフロー変数の使用について

WebLogic Integration がサポートしているビジネス プロトコル

WebLogic Integration がサポートしているビジネス プロトコルに関して以下の注意事項があります。

- RosettaNet プロトコルに基づく会話とワークフローを実装する場合、手順については『*B2B Integration RosettaNet の実装*』を参照してください。これらの手順は RosettaNet 1.1 および 2.0 プロトコルに基づいています。
- ebXML ビジネス プロトコルはこのドキュメントで説明しているものよりシンプルな会話モデルを採用しています。このため、使用するのが ebXML ビ

ジネス プロトコルのみであれば、このドキュメントではなく ebXML ドキュメントを直接参照されることをお勧めします。詳細については、『*B2B Integration ebXML の実装*』の「ワークフローで ebXML を使用する」を参照してください。

- このドキュメントで説明する B2B 機能でサポートしている XOCP ビジネス プロトコルは、WebLogic Integration の本リリースより廃止されています。XOCP の代替となるビジネス プロトコル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。

B2B Integration プラグインを使用したテンプレートの定義について

Application Integration で会話ロールのワークフローを作成するには、B2B Integration プラグインを使用します。ワークフローの作成では、まず初めに、ワークフロー テンプレートとテンプレート定義を定義します。『*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」で説明されているテンプレートとテンプレート定義の標準プロパティの他に、テンプレート定義に基づくワークフローを会話で使用できるようにする追加のプロパティも定義する必要があります。

たとえば、ワークフロー テンプレート定義を、WebLogic Integration リポジトリのコラボレーション アグリーメントにバインドされる会話定義およびバージョンの特定のロールにリンクするとします。その場合は、そのワークフローで実装される会話ロールがバインドされるプロトコルなどの追加属性や、そのプロトコルで必要とされる可能性のある追加のコンフィグレーション データも定義する必要があります。

ワークフロー テンプレートの定義の概略については、『*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」を参照してください。

テンプレートとテンプレート定義について

『*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」では、テンプレートとテンプレート定義について包括的に説明しています。特に、テンプレートとテンプレート定義で供給する情報の種類の違いについて説明しています。

テンプレート定義で必須のデータで、テンプレートで定義することのできないのは、期限の日付とそのテンプレート定義がアクティブかどうかだけです。テンプレートでは、以下のような会話プロパティをすべて設定することをお勧めします。

- 会話の名前とバージョン
- 会話ロール
- 会話プロトコル

テンプレート定義を作成するときには、テンプレートで定義された会話プロパティを必要に応じて柔軟にオーバーライドできます。しかし、テンプレート定義の代わりにテンプレートでそれらのプロパティを割り当てる第一の利点は、特定のテンプレートに含まれるすべてのテンプレート定義のプロパティを指定できるということです。

注意： WebLogic Integration の制限により、テンプレート定義をエクスポートするときにテンプレートのプロパティはエクスポートされません。つまり、テンプレート定義がテンプレートから継承したプロパティはエクスポートされません。テンプレートあるいはテンプレート定義で定義するプロパティを決めるときには、エクスポートの要件を考慮する必要があります。

テンプレート定義のプロパティはエクスポート可能ですが、1 つのテンプレート定義のみに限定されます。

以降の節では、テンプレートで会話プロパティを定義する方法を説明しますが、それらすべてのプロパティは代わりにテンプレート定義で定義することも可能です。

ワークフロー テンプレートの作成

B2B Integration プラグインを使用して Studio でワークフロー テンプレートを作成するには、次の手順を行います。

1. 以下のいずれか 1 つを実行します。
   - 新しいテンプレートを作成するには、フォルダ ツリーで新しいテンプレートが格納されるテンプレートを右クリックして、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。
   - 既存のテンプレートを開くには、フォルダ ツリーでテンプレートを右クリックし、ポップアップ メニューから [開く] を選択します。
2. テンプレート名を右クリックし、ポップアップ メニューから [プロパティ] を選択して [テンプレートのプロパティ] ダイアログ ボックス（下の図）を表示します。

図 2-1 [テンプレートのプロパティ] ダイアログ ボックス

3. 『*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」の説明に従って [一般] ページのフィールドを設定します。
4. [OK] をクリックして変更を保存します。

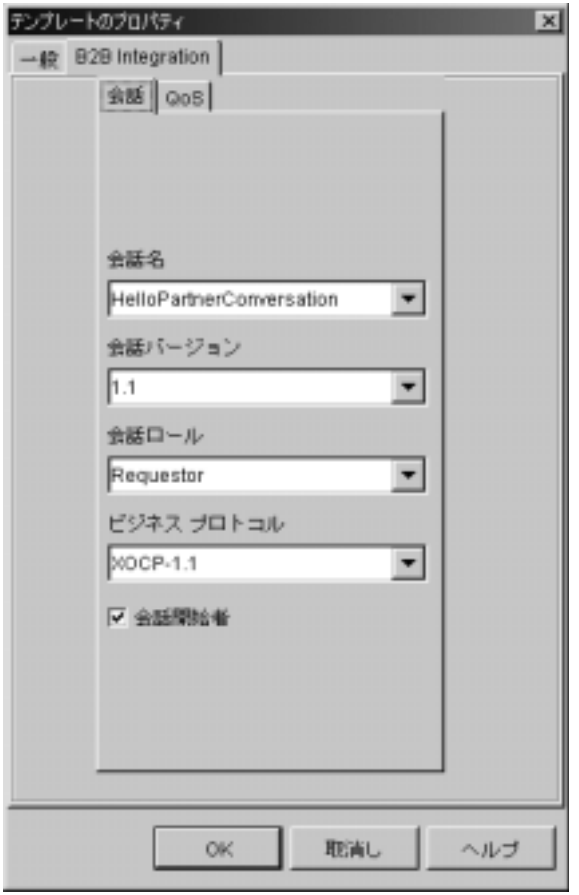
テンプレートと会話のリンク

協調的ワークフローを使用して Application Integration でビジネス メッセージを交換する前に、まずワークフロー テンプレートをリポジトリの特定の会話定義（会話名、バージョン、ロール、およびプロトコル）にリンクする必要があります。

ワークフロー テンプレートを会話定義にリンクするには、次の手順を行います。

1. 2-4 ページの「ワークフロー テンプレートの作成」の説明に従って [テンプレートのプロパティ] ダイアログ ボックスを開きます。
2. [テンプレートのプロパティ] ダイアログ ボックスで、[B2B Integration] タブ（次の図）を選択します。

図 2-2 [B2B Integration] ページ



3. [会話] ページの以下のフィールドを設定します。

表 2-1 [会話] ページのフィールド

フィールド	説明
[会話名]	このワークフロー テンプレート定義とリンクするリポジトリ内の会話定義の名前
[会話バージョン]	このワークフロー テンプレート定義とリンクするリポジトリ内の会話定義のバージョン番号

表 2-1 [会話] ページのフィールド (続き)

フィールド	説明
[会話ロール]	このワークフロー テンプレート定義とリンクする会話定義のロール。トレーディング パートナがこの会話でメッセージを受信するためには、実行時に会話のこのロールで登録する必要がある。
[ビジネス プロトコル]	このワークフロー テンプレート定義とリンクされた会話定義のビジネス プロトコル 利用可能なビジネス プロトコルは以下のとおり。 ♦ XOCP -1.1 ♦ RosettaNet -2.0 ♦ RosettaNet -1.1 XOCP を選択した場合は、追加のタブが利用可能になる。そのタブでは、XOCP プロトコルで交換されるビジネス メッセージのサービス品質（QoS）設定を選択できる。 RosettaNet プロトコルを選択する場合、RosettaNet ベースの会話で使用するワークフローの会話プロパティを定義する方法については、『*B2B Integration RosettaNet の実装*』を参照。
[会話開始者]	ワークフローから会話が開始される場合はこのオプションを選択する。

4. [OK] をクリックして変更を保存します。

テンプレートとコラボレーション プロトコルのリンク

通常、テンプレートを特定のプロトコルにリンクすると、Studio 全体で B2B Integration プラグインのダイアログ ボックスやタブの内容が変わります。このマニュアルで、B2B Integration プラグインのさまざまなコンポーネントを使用してさまざまなタスク（ワークフロー開始プロパティの定義、ビジネス メッセージの送受信や、会話の終了など）を実行する方法を説明していくにしたがって、ワークフロー テンプレートで選択したプロトコルによって各プラグイン コンポーネントの表示がどのように変わるのかを確認できます。

たとえば表 2-1 では、ワークフロー テンプレートのコンフィグレーションで XOCP プロトコルを選択した場合に、そのテンプレートに基づくワークフローで交換されるビジネス メッセージのサービス品質（QoS）設定を指定できるタブが追加表示されることが説明されています。

テンプレートとプロトコルのリンクに応じて、次のことを行います。

- ワークフロー テンプレートを RosettaNet プロトコルにリンクする場合は、[テンプレートのプロパティ] ダイアログ ボックスで [OK] をクリックします。テンプレートの定義タスクはこれで完了です。
- テンプレートを XOCP プロトコルにリンクする場合は、必要に応じて、次節の説明に従って [QoS] タブで QoS 設定を指定できます。

ワークフロー テンプレートでの XOCP メッセージ配信のサービス品質の定義（非推奨）

サービス品質（QoS）は、ビジネス メッセージの信頼性を高めるために定義する属性のセットです。QoS 属性は、XOCP プロトコルで送信するビジネス メッセージでのみ設定できます。

QoS は、Studio を使用して以下の要素で設定できます。

- ワークフロー テンプレート。アクションの定義によってオーバーライドされない限り、設定はすべての Send Business Message アクションに適用されます。
- ワークフロー テンプレート定義。同じように、アクションの定義でオーバーライドされない限り、設定はすべての Send Business Message アクションに適用されます。
- Send Business Message アクション。設定は特定のアクションのみに適用されますが、テンプレートまたはテンプレート定義で指定された設定をオーバーライドします。詳細については、5-12 ページの「Send Business Message アクションでのメッセージ配信のサービス品質の定義（非推奨）」を参照してください。

ワークフロー テンプレートでサービス品質を指定するには、次の手順を行います。

1. 次の図で示されている [QoS] タブを選択します。

図 2-3 [QoS] プロパティのページ

2. [QoS] ページの以下のフィールドを設定します。

表 2-2 [Qos] プロパティ

フィールド	説明
[配信確認]	メッセージ配信確認のレベル（[XOCP ハブまで]（デフォルト）、[XOCP ハブ ルータまで]、または [すべての送り先]）。どれを選択するかによって、[メッセージ トークンの割り当て] ダイアログ ボックスで選択できるオプションが決まる（5-7 ページの「メッセージ トークン情報のワークフロー変数への割り当て」を参照）。 以下のいずれかの値を選択できる。 ♦ [XOCP ハブまで] — メッセージが仲介機能に達したときに配信確認を必要とする（デフォルト）。このオプションを選択すると、最高の実行時パフォーマンスを維持して基本的な配信確認が提供される。 ♦ [XOCP ハブ ルータまで] — メッセージが仲介機能のルータに達したときに配信確認を必要とする。このオプションでは、メッセージを受信するためにルータによって選択されたトレーディング パートナのリストが提供される。 ♦ [すべての送り先] — すべての送り先から配信確認を必要とする。このオプションを選択すると、最高レベルの配信確認が提供される。実行時のパフォーマンスが低下する恐れがある。
[持続性]	[メッセージの永続性] ボックスを選択すると、メッセージが永続的に保存される。このオプションを選択すると、システムの障害から回復する可能性が高まるが、処理が増えて実行時のパフォーマンスが低下する場合がある。 [メッセージの永続性] が選択されていない場合、メッセージは永続的には保存されない。このオプションでは、実行時のパフォーマンスが向上するが、システムの障害から回復する可能性が低下する。

表 2-2 [Qos] プロパティ (続き)

フィールド	説明
[再試行回数]	メッセージ送信の最大の再試行回数（デフォルトは 0）。WebLogic Integration プロセス エンジンでは、送信が成功するか、最大再試行回数に達するまでメッセージの送信が繰り返し試行される。最大の再試行回数を超えると例外が送出される。
[相関 ID]	メッセージをアプリケーションの他のビジネス メッセージと関連付けるために使用するメッセージ識別文字列（デフォルトは none）。たとえば、トレーディング パートナでは応答が元の要求と一致するように要求で相関 ID を指定する場合がある。B2B メッセージング サービスでは、メッセージにこのプロパティが含まれる。
[送信タイムアウト]	メッセージを送信する際のタイムアウト値（日、時、分、秒単位）。デフォルトは 0 で、その場合はタイムアウトは適用されない。WebLogic Integration プロセス エンジンは、配信確認が届くか、タイムアウトに達するまで待機する。
[メモ]	説明テキスト（省略可能）

3. [OK] をクリックして設定を保存します。

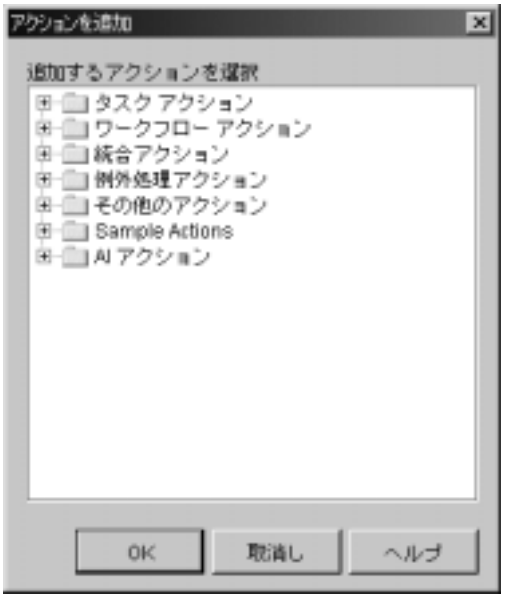
メッセージのロギング

B2B Integration プラグインは、協調的ワークフロー実行時の特定の時点でメッセージ（デバッグ メッセージ、実行時情報、アプリケーション メッセージなど）を B2B ログに送信する手段を提供します。この機能は、ワークフローの任意のノードに追加できる Log アクションを通じて提供されます。このアクションは、ワークフローの設計過程で便利なデバッグ ツールとして使用できます。

Log アクションをワークフローのノードに追加するには、次の手順を行います。

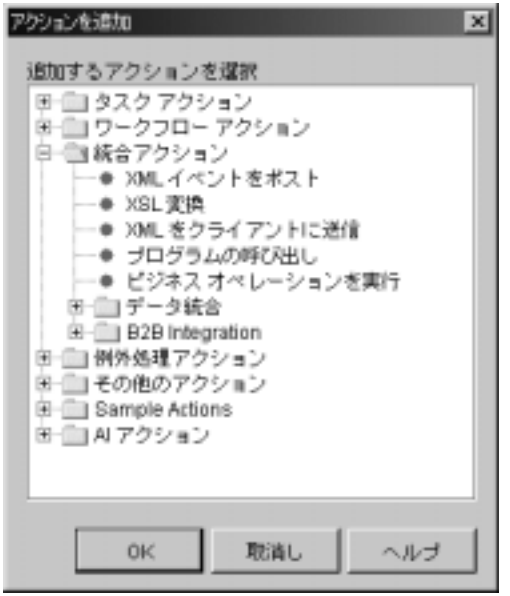
1. Log アクションを追加するワークフローを開きます。ワークフローはアクティブであり、有効でなければなりません。
2. アクションを指定できるダイアログ ボックスで、[追加] をクリックして [アクションを追加] ダイアログ ボックスを表示します。

図 2-4 [アクションを追加] ダイアログ ボックス



3. [統合アクション] フォルダをクリックして展開します。

図 2-5 [アクションを追加] ダイアログ ボックスとワークフロー アクション



4. [B2B Integration] フォルダをクリックして展開します。
5. [ログ] を選択します。[ログ] アクション ダイアログ ボックスが表示されます。

図 2-6　[ログ] アクション ダイアログ ボックス

6. B2B ログに送信するメッセージを入力するか、Expression Builder を使用してログに送信するメッセージを作成します。Expression Builder の使い方については、『*WebLogic Integration Studio ユーザーズ ガイド*』の「ワークフロー式の使用法」を参照してください。

ワークフロー変数の使用について

通常、ワークフロー変数は実行時にワークフローで必要とされるアプリケーション固有の情報を格納するために使用します。変数は主に、ワークフロー インスタンスの論理パスを制御するために作成して値を割り当てます。同じワークフロー テンプレート定義を複数回インスタンス化し、分岐ノードが含まれている場合は異なる方法で進行できます。

協調的ワークフロー以外のワークフローに適用される変数処理のガイドラインは、協調的ワークフローにも当てはまります（詳細は『*WebLogic Integration Studio ユーザーズガイド*』を参照）。ワークフローを作成するときには、以下のガイドラインに注意してください。

- 実行時に変数を必要とするワークフローは、Studio でワークフロー変数として定義する必要があります。ワークフローの実行時に、ワークフロー変数には以下の方法でアクセスできます。
  - ワークフロー インスタンスで（たとえば、分岐ノードで）
  - ワークフローを開始する、またはワークフローで送受信されるビジネス メッセージを操作するエンタープライズ JavaBean（EJB）または Java クラスで
- 構造化プログラミング言語の場合と同じように、変数はワークフローの作成前に定義するようにします。Studio を使用する場合、ワークフロー作成前に変数を定義することには、以下の 2 つのはっきりとした利点があります。
  - 作成前にワークフローをより綿密に設計するようになり、設計サイクルがより効率的かつ計画的になります。
  - Studio のさまざまなダイアログ ボックスやタブで変数を識別するときに、変数が選択ボックスに表示されます。変数の選択に便利なだけではなく、変数の型と名前をワークフローの特定のノードに関連付けるときにエラーのリスクが減少します。
- プログラム的にワークフローを開始する Java アプリケーションでは、ワークフローの開始時に取得したワークフロー インスタンス オブジェクトを参照することでワークフローに固有のワークフロー変数にいつでもアクセスできます。

注意： プログラム的にワークフローを開始するアプリケーションでは、Message Manipulator API（パッケージ `com.bea.b2b.wlpi`）を使用する必要があります。詳細については、3-21 ページの「会話開始者ワークフローを開始するアプリケーションの開発」を参照してください。

ワークフロー変数と Java データ型の関連付け

ビジネス オペレーションの中でワークフロー変数にアクセスする必要がある場合は、ワークフロー変数の型が Java データ型とどのように対応しているのかを知る必要があります。次の表は、それらの型がどのように関連しているのかを示しています。

表 2-3 ワークフロー変数と Java データ型

ワークフロー変数の型	Java データ型
ï¹éöóÒ	java.lang.String
Integer	java.lang.Long
Double	java.lang.Double
ìŽït	java.util.Date
Boolean	java.lang.Boolean
Complex Object	java.lang.Object （Serializable を実装する必要がある）
XML	org.w3c.dom.Node

ワークフロー変数を定義する際のルール

協調的ワークフローのワークフロー変数を定義する際のルールは、非協調的なワークフローの場合と同じです。

- ワークフロー変数には、以下の状況でプログラム的にアクセスできます。
  - インスタンスが作成された後で、それが開始される前（4-26 ページの「ステップ 5 : ワークフロー インスタンスから入力変数を取得する」を参照）
  - Manipulate Business Message アクションの実行時（4-16 ページの「Manipulate Business Message アクションの定義」を参照）
  - インスタンスの完了後は出力変数にアクセス可能

- アプリケーションでワークフロー インスタンスの入力変数を設定できるようにするには、まず Studio でそのワークフローの変数を定義する必要があります。このアプリケーションでは、Message Manipulator API（パッケージ `com.bea.b2b.wlpi`）を使用してワークフローに渡されるそれらの変数を設定する必要があります。
- アプリケーションによってワークフローが開始されている場合は、アプリケーションでワークフロー インスタンスを使用して変数を設定または取得できます。
- 出力変数（ワークフローから渡される変数）を定義する場合は、ワークフローが終了した後にその値にアクセスできます。
- ビジネス メッセージにアクセスするアクションでは、Java オブジェクト変数を使用してビジネス メッセージを操作、送信、または受信します。ただし、それらのアクションを通じて変数に格納されたオブジェクトはビジネス メッセージをカプセル化する内部クラスに属します。したがって、それらの変数にはメッセージ マニピュレータを使用してアクセスする必要があります。変数に直接アクセスした場合、動作は保証されません。詳細については、4-2 ページの「ビジネス メッセージのワークフロー変数の定義」を参照してください。
- Compose Business Message アクションおよび Extract Business Message Parts アクションで変数を設定する場合は、変数が以下の適切な型であることを確認してください。
  - XML ドキュメントを保持するビジネス メッセージの部分は、`XML` 型のワークフロー変数で格納または取得できます。
  - ビジネス メッセージのどの部分でもファイルに格納またはファイルから取得できます。そのファイル名は、`String` 型のワークフロー変数に格納されます。

  ビジネス メッセージのバイナリ添付ファイルを指定する変数には、ソース ファイルまたは送り先ファイルを適切に格納する必要があります。

変数の定義

Studio におけるワークフロー変数の定義に関する詳細については、『*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。

3　協調的ワークフローの開始

以下の節では、協調的ワークフローの開始アクションを定義する方法について説明します。

- 協調的ワークフローの開始について
- 会話開始者ワークフローの開始
- 会話参加者ワークフローの開始ノードの定義
- Start Public Workflow アクションの定義
- 会話開始者ワークフローを開始するアプリケーションの開発

注意： このドキュメントで説明する B2B 機能でサポートしている XOCP ビジネス プロトコルは、WebLogic Integration の本リリースより廃止されています。XOCP の代替となるビジネス プロトコル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。

協調的ワークフローの開始について

注意： `RosettaNet` プロトコルに基づく会話とワークフローを実装する場合、手順については『*B2B Integration RosettaNet の実装*』を参照してください。 これらの手順は `RosettaNet 1.1` および `2.0` プロトコルに基づいています。

ebXML ビジネス プロトコルは、このドキュメントで説明しているものよりシンプルな会話モデルを採用しています。このため、使用するのが ebXML ビジネス プロトコルのみであれば、このドキュメントではなく ebXML ドキュメントを直接参照されることをお勧めします。ebXML ビジネス プロトコルのワークフロー作成に関する情報については、『*B2B Integration ebXML の実装*』の「ワークフローで ebXML を使用する」を参照してください。

協調的ワークフローの開始作業は、以下の 2 つのタスクで構成されます。

- 協調的ワークフロー開始プロパティの定義
- 実行時の協調的ワークフローの開始

以降の 2 節では、各タスクの設計およびプログラミングの関係について説明します。

協調的ワークフロー開始プロパティの定義

協調的ワークフローは、以下の 3 つの方法で開始できます。

- ワークフローを開始する Java アプリケーションでプログラム的に
- 親ワークフローのノードで定義された Start Public Workflow アクションを通じてサブワークフローとして
- ビジネス メッセージで開始する状態で、ビジネス メッセージを受信したときに

ワークフローの開始プロパティは、設計するワークフローのタイプ（会話開始者か会話参加者か）に基づき、以下のルールに従って定義します。

- プログラム的に開始される会話開始者ワークフローの場合は、開始ノードのプロパティで [手動] を指定します。
- Start Public Workflow アクションを通じて開始される会話開始者ワークフローの場合は、開始ノードで開始状態を [呼び出し] に指定します。
- 会話参加者ワークフローの場合は、開始ノードでビジネス メッセージで開始するように定義します。

注意： ワークフローをインスタンス化するには、ワークフロー テンプレート定義がアクティブかつ有効である必要があります。

実行時の協調的ワークフローの開始

実行時に協調的ワークフローを開始する方法は、ワークフローが会話開始者または会話参加者のどちらなのかによって異なります。

- 会話開始者ワークフローは、以下の 2 つの方法のいずれかで開始できます。
  - 親ワークフローまたは他のローカル ワークフローが、親ワークフローのノードで定義されている Start Public Workflow アクションを使用して会話開始者ワークフローを開始できます。Start Public Workflow アクションの定義されているノードが実行されたときに、会話開始者ワークフローが開始されます。
  - ユーザ記述アプリケーションが実行時に会話開始者ワークフローを開始できます。WebLogic Integration では、アプリケーションでワークフローの開始や例外の処理に使用できる Message Manipulator API （パッケージ `com.bea.b2b.wlpi`）が提供されます。
- 会話参加者ワークフローは、そのワークフローを開始するように定義されたビジネス メッセージが WebLogic Integration プロセス エンジンで受信されたときに開始されます。

会話開始者ワークフローの開始

会話開始者ワークフローは、以下のいずれかの方法で開始します。

- Java アプリケーションを通じてプログラム的に
- 別のローカル ワークフローを通じて

プログラム的に開始される会話開始者ワークフローでは、開始ノードのプロパティで [手動] を定義する必要があります。プログラム的に会話開始者ワークフローを開始する方法の詳細については、3-21 ページの「会話開始者ワークフローを開始するアプリケーションの開発」を参照してください。

別のローカル ワークフローを通じて開始される会話開始者ワークフローでは、開始ノードで開始プロパティを [呼び出し] に定義する必要があります。ローカル ワークフロー（サンプル アプリケーションで使用する設計パターンではプライベート ワークフローと呼ぶ）では、会話開始者ワークフローをサブワークフローとして開始できます。サンプル アプリケーションの詳細については、『*WebLogic Integration チュートリアル*』および『*B2B Integration サンプルの使い方*』を参照してください。

会話開始者ワークフローの開始プロパティを定義するには、次の手順を行います。

1. 『*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」の説明に従って開始シェイプを表示または追加します。
2. 開始シェイプ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。

図 3-1 [開始のプロパティ] ダイアログ ボックス：手動開始

3. 必要に応じて、[説明] フィールドのテキストをユニークで識別可能な名前に変更します。
4. ワークフローがプログラム的に開始される場合は、[手動] を選択します。ワークフローが別のワークフローを通じて開始される場合は、[呼び出し] を選択します。

   これらのオプションに関する詳細については、『*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」を参照してください。
5. [OK] をクリックします。

会話参加者ワークフローの開始ノードの定義

会話参加者ワークフローは、トレーディング パートナから最初のビジネス メッセージを受信したときに開始されます。そのようなワークフローでは、ビジネス メッセージで開始するように定義する必要があります。

会話参加者ワークフローのビジネス メッセージで開始するように定義するには、次の手順を行います。

1. 『*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」の説明に従って開始シェイプを表示または追加します。
2. 開始シェイプ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。
3. [開始のプロパティ] ダイアログ ボックスで [イベント] を選択し、右側に表示されるドロップダウン リストで [Business Message] を選択します。

図 3-2 ビジネス メッセージで開始するように選択

[開始のプロパティ] ダイアログ ボックスが、次の図のようにフィールドを追加して再表示されます。これらのフィールドでは、協調的ワークフローに

固有の情報（特にワークフローが基づくビジネス プロトコルに関連する情報）を追加します。次の図は、XOCP プロトコルに基づくワークフローの [開始のプロパティ] ダイアログ ボックスを示しています。

図 3-3　会話参加者ワークフローの開始プロパティ

4. [説明] フィールドのテキストをユニークで識別可能な名前に変更します。
5. ワークフローが基づくビジネス プロトコルで必要な情報を入力します。詳細については、以下の節を参照してください。
   - XOCP 1.1 ベースのワークフローの場合は、3-8 ページの「XOCP 1.1 プロトコルに基づく会話参加者ワークフローの開始（非推奨）」を参照してください。

- RosettaNet 2.0 ベースのワークフローの場合は、3-9 ページの「RosettaNet 2.0 プロトコルに基づく会話参加者ワークフローの開始」を参照してください。
- RosettaNet 1.1 ベースのワークフローの場合は、3-9 ページの「RosettaNet 1.1 プロトコルに基づく会話参加者ワークフローの開始」を参照してください。

6. 『*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」の説明に従って開始オーガニゼーションを指定します。
7. [OK] をクリックします。

XOCP 1.1 プロトコルに基づく会話参加者ワークフローの開始（非推奨）

[開始のプロパティ] ダイアログ ボックスで、XOCP 1.1 プロトコルに基づくワークフローの以下の情報を入力します。

図 3-4 XOCP 1.1 開始プロパティ

1. [対象変数] は、内部型のインスタンスが格納される `object` 型のワークフロー変数です。内部型のインスタンスには、ビジネス メッセージが格納されます。
2. [ルータ式] は、ビジネス メッセージの送信側で設定されたルータ式が格納される `string` 型のワークフロー変数です。

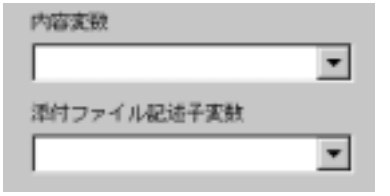
3. [送信側の名前] は、ビジネス メッセージを送信したトレーディング パートナの名前が格納される `string` 型のワークフロー変数です。

   送信側の名前を XPath 式に変換する場合は、[XPath 変換] ボタンを選択します。

RosettaNet 2.0 プロトコルに基づく会話参加者ワークフローの開始

[開始のプロパティ] ダイアログ ボックスで、RosettaNet 2.0 プロトコルに基づくワークフローの以下の情報を入力します。

図 3-5　RosettaNet 2.0 開始プロパティ



1. [内容変数] は、ビジネス メッセージの内容が格納される `XML` 型のワークフロー変数です。
2. [添付ファイル記述子変数] は、ビジネス メッセージの添付ファイルを記述する XML データが格納される `XML` 型のワークフロー変数（オプション）です。

RosettaNet 2.0 プロトコルに基づくワークフローの開始プロパティ作成の詳細については、『*B2B Integration RosettaNet の実装*』を参照してください。

RosettaNet 1.1 プロトコルに基づく会話参加者ワークフローの開始

[開始のプロパティ] ダイアログ ボックスで、RosettaNet 1.1 プロトコルに基づくワークフローの以下の情報を入力します。

図 3-6 RosettaNet 1.1 開始プロパティ

1. [内容変数] は、ビジネス メッセージの内容が格納される XML 型のワークフロー変数です。
2. [添付ファイル記述子変数] は、ビジネス メッセージの添付ファイルを記述する XML データが格納される XML 型のワークフロー変数（オプション）です。

RosettaNet 1.1 プロトコルに基づくワークフローの開始プロパティ作成の詳細については、『*B2B Integration RosettaNet の実装*』を参照してください。

Start Public Workflow アクションの定義

Start Public Workflow アクションを使用すると、親ワークフローから協調的ワークフローを開始できます。Start Public Workflow アクションを使用するときには、以下の事項を指定する必要があります。

- 呼び出されるワークフロー インスタンスと関連付けられたコラボレーション アグリーメントをユニークに識別するために必要な会話定義、ロール、および最小限のパーティ数の値
- 子ワークフローの入力変数の値
- 子ワークフローのインスタンス ID が格納される変数の名前
- 親ワークフローに関連を持つ子の出力変数、および子ワークフローの出力値が格納される親ワークフローの変数

Start Public Workflow アクションを通じて協調的ワークフローを開始する場合、その協調的ワークフローには親ワークフローと同期を取るための手段が必要です。

以降の節では、Start Public Workflow アクションを定義する方法、および呼び出される協調的ワークフローと親ワークフローを同期させる方法について説明します。

Start Public Workflow アクションの定義

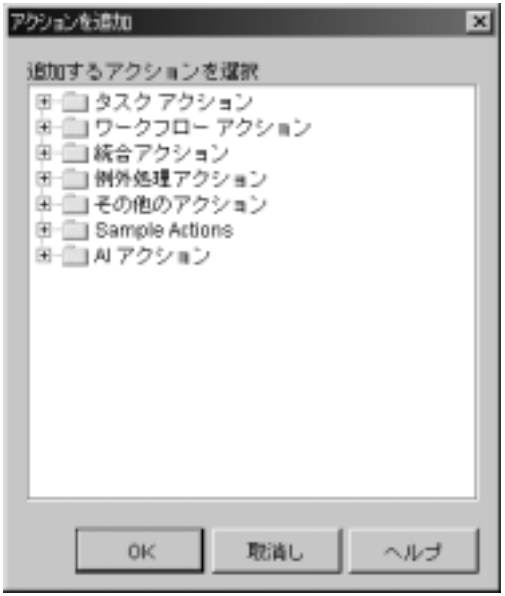
Start Public Workflow アクションはどのワークフロー ノードとでも関連付けることができますが、通常はタスク ノードと関連付けます。Start Public Workflow アクションは明示的にワークフロー テンプレート定義に追加する必要があります。

Start Public Workflow アクションの追加

Studio で親ワークフローの Start Public Workflow アクションを定義するには、次の手順を行います。

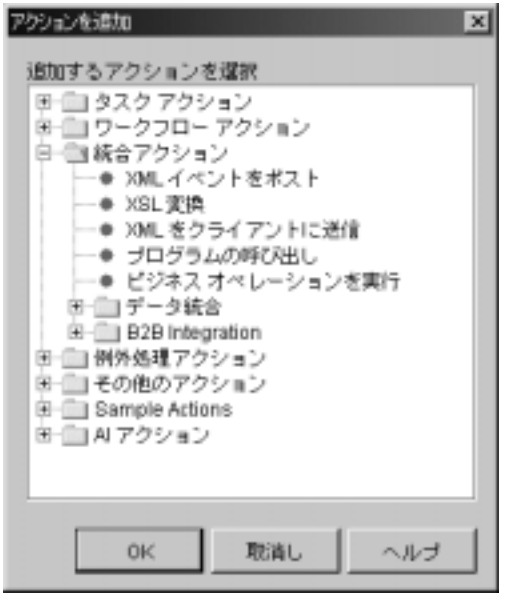
1. Start Public Workflow アクションを追加するワークフローを開きます。ワークフローはアクティブかつ有効でなければなりません。
2. （通常はタスク ノードで）アクションを指定できるダイアログ ボックスで、[追加] をクリックして [アクションを追加] ダイアログ ボックスを表示します。

図 3-7 [アクションを追加] ダイアログ ボックス



3. [統合アクション] フォルダをクリックして展開します。

図 3-8 [アクションを追加] ダイアログ ボックスと統合アクション



4. [B2B Integration] フォルダをクリックして展開します。
5. [Start Public Workflow] を選択します。
6. [OK] をクリックして、[パブリック ワークフローを開始] ダイアログ ボックスを表示します。

図 3-9 [パブリック ワークフローを開始 **] ダイアログ ボックス**

7. [パブリック ワークフローを開始] ダイアログ ボックスの [会話] ページの以下のセクションを設定します。このページで指定した値と変数は、会話で必要なワークフローが含まれるコラボレーション アグリーメントをユニークに識別します。

表 3-1 [会話] ページのセクション

ダイアログ ボックスのセクション	説明
[会話]	以下の情報を入力する。 ■ WebLogic Integration リポジトリの会話定義の名前 ■ 会話定義のバージョン ■ 開始される協調的ワークフローで表される会話ロール

表 3-1 [会話] ページのセクション (定義)

ダイアログ ボックスのセクション	説明
[パーティ]	ワークフローが適用される関連付けられている会話定義のコラボレーション アグリーメントでコンフィグレーションされたパーティのトレーディング パートナ名、ロール、および配信チャネル。すべてのパーティを指定する必要はないが、ワークフローと関連付けられたコラボレーション アグリーメントをユニークに識別できなければならない。 ロールおよび配信チャネルで入力する情報は、リポジトリの対応するトレーディング パートナでコンフィグレーションされた情報と正確に同じでなければならない。ロールと配信チャネルの指定はオプションであり、その情報がコラボレーション アグリーメントをユニークに識別するために必要かどうかによって異なる。 ■ トレーディング パートナ名を指定するには、[TP 名] カラムのセルをクリックしてトレーディング パートナの名前を入力する。 ■ ロールを指定するには、[ロール] カラムのセルをクリックしてロール名を入力する。 ■ 配信チャネルを指定するには、[配信チャネル] カラムのセルをクリックして配信チャネル名を入力する。 ■ パーティのリストに行を挿入するには、挿入を行う行を右クリックして [挿入] を選択する。クリックした行の上に空白の行が作成される。 ■ パーティのリストから行を削除するには、削除する行を右クリックして [削除] を選択する。 Expression Builder を使用すると、[パブリック ワークフローを開始] ダイアログ ボックスでパーティの値を指定できる。詳細については、3-20 ページの「[パブリック ワークフローを開始] ダイアログ ボックスでの Expression Builder を使用した値の指定」を参照してください。
[メモ]	説明テキスト（省略可能）

表 3-1 [会話] ページのセクション (定義)

ダイアログ ボックスのセクション	説明
[アクション]	ワークフロー アクションを追加できるタブ。[アクション] タブでは、通常は、パブリック ワークフローと親ワークフローの同期を取るために「タスクに完了マークを付ける」プロパティを追加する。詳細については、3-21 ページの「子ワークフローと親ワークフローの同期」を参照してください。

8. [ワークフロー] タブを選択します。次のページが表示されます。

図 3-10 [パブリック ワークフローを開始] ダイアログ ボックスの [ワークフロー] ページ

9. [ワークフロー] タブの以下のフィールドとセクションを設定します。子ワークフローのワークフロー インスタンス ID が格納される変数と親ワークフローと子ワークフローの間で渡される変数の両方を識別します。

表 3-2 [ワークフロー] タブのフィールドとセクション

フィールドまたはセクション	説明
[参照を媒介する変数]	子ワークフローのインスタンス ID が格納される `String` 型のワークフロー変数。この変数は、親ワークフローと子ワークフローの標準の通信メカニズムを実装する手段として使用する。
[パラメータ]	親ワークフローから子ワークフローに渡される変数の名前（[名前] カラム）と値（[式] カラム）。ここで指定する変数名は、子ワークフローに対する入力変数としても指定する必要がある。2-16 ページの「ワークフロー変数の使用について」で説明されているように、これらの変数はこのダイアログ ボックスで選択する前に親ワークフローと子ワークフローの両方で定義する必要がある。そうすることで、選択された変数が適切な名前と型を持つようになる。 Studio の Expression Builder を使用すると、各変数の値の式を作成できる。Expression Builder の使い方については、3-20 ページの「[パブリック ワークフローを開始] ダイアログ ボックスでの Expression Builder を使用した値の指定」を参照。
[結果]	子ワークフローの実行が完了した後に、子ワークフローから親ワークフローに返される出力変数の名前と値。入力変数と同じように、それらの出力変数はこのダイアログ ボックスで選択する前に親ワークフローと子ワークフローの両方で定義する必要がある。
[メモ]	説明テキスト（省略可能）

10. [OK] をクリックして変更を保存します。

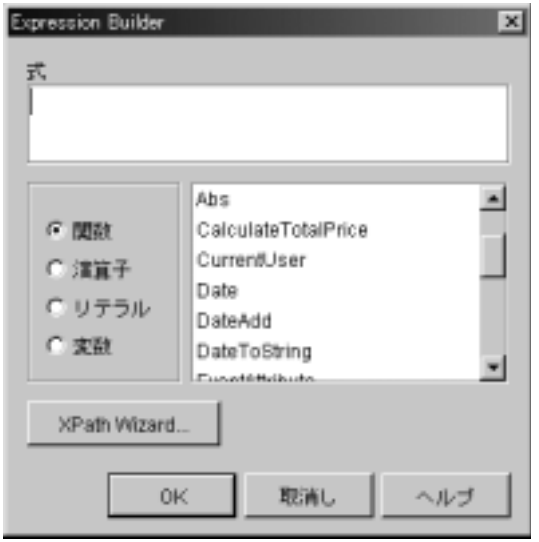
[パブリック ワークフローを開始] ダイアログ ボックスでの Expression Builder を使用した値の指定

Studio の Expression Builder を使用して、[パブリック ワークフローを開始] ダイアログ ボックスで以下の値を指定できます。

- トレーディング パートナ名
- Roles (ロール)
- 配信チャネル
- ワークフロー パラメータ式

Expression Builder を表示するには、上記のいずれかの値を指定するセルを右クリックして [式] を選択します。次の図のように Expression Builder が表示されます。

図 3-11 Studio の Expression Builder



Expression Builder の使い方については、『*WebLogic Integration Studio ユーザーズガイド*』の「ワークフロー式の使用法」を参照してください。

子ワークフローと親ワークフローの同期

3-10 ページの「Start Public Workflow アクションの定義」で説明したように、Start Public Workflow アクションを通じて協調的ワークフローを開始するためには、その協調的ワークフローと親ワークフローの同期を取る手段が必要です。同期は、以下のいずれかの方法で実現できます。

- ワークフローとの同期を実装するアプリケーションを使用します。
- Start Public Workflow アクションを定義する親ワークフローで、（Task アクションではなく）Start Public Workflow アクションでタスクに完了のマークを付けます。このようにすると、呼び出されたパブリック ワークフローで処理が終了するまで親ワークフローで次のノードに実行が移行しません。

会話開始者ワークフローを開始するアプリケーションの開発

会話開始者ワークフローは、Java アプリケーション（ワークフロー アプリケーション）を通じて実行時に開始できます。以下の節では、会話開始者ワークフローをプログラム的に開始するワークフロー アプリケーションの作成方法を説明します。

- Message Manipulator API
- 会話開始者ワークフローにアクセスするためのプログラミング手順

会話開始者ワークフローをプログラム的に開始するためには、3-3 ページの「会話開始者ワークフローの開始」で説明されているようにワークフロー テンプレートの開始ノードで [手動] 開始プロパティが必要です。

注意： WebLogic Integration リリース 2.0 以降では、Worklist を通じて協調的ワークフローを開始しようとするとエラーが発生します。

Message Manipulator API

ワークフロー アプリケーションでは、com.bea.b2b.wlpi パッケージを使用してワークフローを開始できます。このパッケージでは、以下のインタフェースとクラスが提供されます。

表 3-3 com.bea.b2b.wlpi パッケージの構成要素

オブジェクト	説明
MessageManipulator インタフェース	ワークフロー タスク内の Manipulate Message アクションで使用されるすべてのクラスで実装される。
WorkflowInstance クラス	実行中のワークフロー インスタンスを表す。
WLPIException クラス	処理エラーが発生した場合にワークフロー API のパブリック メソッドによって送出される。

このパッケージの詳細については、『*BEA WebLogic Integration Javadoc*』を参照してください。

会話開始者ワークフローにアクセスするためのプログラミング手順

会話開始者ワークフローにアクセスするために、ワークフロー アプリケーションでは次の手順を行います。

- 手順 1 : 必要なパッケージをインポートする
- 手順 2 : 特定のワークフロー テンプレートのワークフロー インスタンス オブジェクトを作成する
- 手順 3 : 入力変数を初期化する
- 手順 4 : ワークフロー インスタンスの開始
- 手順 5 : ワークフロー インスタンスの完了を待つ
- 手順 6 : 出力変数の結果を処理する

ワークフロー アプリケーションでは、3-26 ページの「例外処理」で説明されているように Message Manipulator API の各メソッド呼び出しで送出される例外も処理する必要があります。

注意： ワークフローを開始するアプリケーションを作成する前に、以下の事項を確認してください。

- ワークフローのテンプレート定義が作成されている
- ワークフローが手動開始可能であり、ワークフローがアクティブかつ有効である

手順 1：必要なパッケージをインポートする

ワークフローにアクセスするために、ワークフロー アプリケーションではまず必要なパッケージをインポートします。最低でも、ワークフロー アプリケーションでは次の例のように `com.bea.b2b.wlpi` パッケージをインポートする必要があります。

コード リスト 3-1　com.bea.b2b.wlpi パッケージのインポート

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.naming.*;

import com.bea.wlpi.common.*;
import com.bea.b2b.wlpi.*;

import com.bea.eci.logging.*;
```

手順 2：特定のワークフロー テンプレートのワークフロー インスタンス オブジェクトを作成する

次に、ワークフロー アプリケーションでは、次の例のように特定のワークフロー テンプレート定義のワークフロー インスタンス オブジェクトを作成する必要があります。

コード リスト 3-2 ワークフロー インスタンス オブジェクトの作成

```
Hashtable[] array = new Hashtable[2];
Hashtable party1  = new Hashtable();
Hashtable party2  = new Hashtable();

party1.put(WorkflowInstance.PARTYNAME_KEY, PARTY_SOURCE);
party2.put(WorkflowInstance.PARTYNAME_KEY, PARTY_DEST);

array[0] = party1;
array[1] = party2;

WorkflowInstance wi = new WorkflowInstance(BUSINESS_PROCESS_NAME,
                                BUSINESS_PROCESS_MAJOR,
                                BUSINESS_PROCESS_MINOR,
                                BUSINESS_PROCESS_ROLE,
                                 array);
```

手順 3：入力変数を初期化する

会話開始者ワークフローで入力変数が定義されている場合は、ワークフロー インスタンスを開始する前にそれらの変数を初期化して値を割り当てる必要があります。それらの入力変数は、2-16 ページの「ワークフロー変数の使用について」で説明されているように Studio においてテンプレート定義で最初に宣言する必要があります。

ワークフロー アプリケーションでは、ワークフロー インスタンスで `setVariable` メソッドを呼び出し、そのメソッドに変数の名前と値を渡してインスタンス変数を設定します。`setVariable` メソッドでは、以下のパラメータが必要です。

表 3-4 setVariable メソッドのパラメータ

パラメータ	説明
`name`	設定する変数の名前
`Value`	変数の値。2-18 ページの「ワークフロー変数と Java データ型の関連付け」で説明されているように、値は Java オブジェクトとして表す必要がある。

次の例は、setVariable メソッドを使用して、開始するワークフローの 2 つの変数の値を指定する方法を示しています。

コード リスト 3-3　入力変数の値の設定

```
wi.setVariable( INTEGER_ONE_VAR, new Long( 3 ) );
wi.setVariable( INTEGER_ONE_VAR, new Long( 5 ) );
```

手順 4 : ワークフロー インスタンスの開始

ワークフロー インスタンスを作成して入力変数を初期化した後、ワークフロー アプリケーションは次の例のようにインスタンス オブジェクトの start メソッドを呼び出してワークフロー インスタンスを開始します。

コード リスト 3-4　ワークフロー インスタンスの開始

```
wi.start();
```

手順 5 : ワークフロー インスタンスの完了を待つ

ワークフロー インスタンスが開始された後、ワークフロー アプリケーションはワークフロー インスタンスの waitForCompletion メソッドを呼び出してその完了を待つことができます。ワークフロー インスタンスが完了するまで処理はブロックされます。

コード リスト 3-5　ワークフロー インスタンスの完了を待つ

```
wi.waitForCompletion();
```

ワークフロー インスタンスの完了を待つ間、ワークフロー アプリケーションではワークフロー インスタンスの `isCompleted` メソッドを呼び出してワークフロー インスタンスの完了状態を確認できます。このメソッドは、ワークフローの実行が完了している場合は `true` を返し、それ以外の場合は `false` を返します。

手順 6：出力変数の結果を処理する

ワークフロー インスタンスが完了した後、ワークフロー アプリケーションでは出力変数に格納された情報を取り出してワークフロー インスタンスの結果を処理できます。それらの出力変数は、Studio においてテンプレート定義で最初に宣言する必要があります。

ワークフロー アプリケーションでは、次の例のように、ワークフロー インスタンスの `getVariable` メソッドを呼び出し、取り出す変数の名前をそのメソッドに渡してインスタンス変数の値を取り出します。

コード リスト 3-6　出力変数の結果の取り出し

```
product = (String)wi.getVariable(MULTIPLY_REPLY_VAR);
note = (String)wi.getVariable(NOTE_VAR);
```

`getVariable` メソッドは、適切な Java データ型にキャストする必要のある Java オブジェクトを返します。対応するワークフロー変数の型に応じて戻り値をキャストする Java の型の詳細については、『*WebLogic Integration Studio ユーザーズガイド*』を参照してください。

例外処理

ワークフロー アプリケーションの実行中にエラーが発生した場合は、`com.bea.b2b.wlpi.WLPIException` が送出されます。ワークフロー API の各メソッド呼び出しで、ワークフロー アプリケーションでは次の例のようにこの例外を捕捉して適切に処理できます。

コード リスト 3-7 ワークフロー アプリケーションでの WLPIExceptions の処理

```
catch (WorkflowException we){
      debug("Error starting workflow");
      we.printStackTrace();
}
```

4　ビジネス メッセージの使い方

以下の節では、ビジネス パートナ間で交換されるビジネス メッセージの内容を B2B Integration プラグインで作成および操作する方法について説明します。

- ビジネス メッセージの処理について
- ビジネス メッセージのワークフロー変数の定義
- 単純なメッセージ操作（非推奨）
- 複雑なメッセージ操作（非推奨）

注意： この章で説明する機能は、WebLogic Integration の本リリースから廃止された XOCP ビジネス プロトコルをベースにしています。RosettaNet ビジネス メッセージの使用に関する詳細については、『*B2B Integration RosettaNet の実装*』を参照してください。ebXMLRosettaNet ビジネス メッセージの使用に関する詳細については、『*B2B Integration ebXML の実装*』を参照してください。XOCP の代替となるビジネス プロトコル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。

ビジネス メッセージの処理について

ビジネス メッセージは、トレーディング パートナによって会話の中で交換される通信の基本単位です。ビジネス メッセージの構成要素は以下のとおりです。

- 1 つまたは複数の*ビジネス ドキュメント*。ビジネス メッセージの XML ベースのペイロード部分を表します。ペイロードは、ビジネス メッセージのビジネス コンテンツです。
- 1 つまたは複数の*添付ファイル*（オプション）。ビジネス メッセージの XML または非 XML ペイロード部分を表します。

B2B Integration プラグインでは、以下の 2 つの方法でビジネス メッセージの内容を作成または抽出できます。

- 単純なメッセージ操作が Compose Business Message アクションと Extract Business Message Parts アクションでサポートされています。これら 2 つのアクションでは、ビジネス メッセージの内容をプログラム的ではない方法で操作できます。現行の WebLogic Integration では、Compose Business Message アクションと Extract Business Message Parts アクションは XOCP メッセージでのみサポートされています。
- 複雑なメッセージ操作が Manipulate Business Message アクションでサポートされています。このアクションでは、Message Manipulator API（パッケージ `com.bea.b2b.wlpi`）の `MessageManipulator` インタフェースを実装するユーザ記述アプリケーションが開始されます。このインタフェースを使用すると、複雑なメッセージを作成および操作できます。現行の WebLogic Integration では、Manipulate Business Message アクションは XOCP メッセージでのみサポートされています。

ビジネス メッセージのワークフロー変数の定義

実行時に、ビジネス メッセージは送信される前または受信された後に（`Java Object` 型の）ワークフロー変数に格納されます。ビジネス メッセージが送信可能になると、ビジネス メッセージを作成するアクションと関連付けられているアプリケーション コードではそのビジネス メッセージを作成し、それをこの変数に格納してワークフロー インスタンスに返します。ビジネス メッセージが受信されると、Compose Business Message、Extract Business Message Parts、または Manipulate Business Message といったアクションと関連付けられているワークフロー ノードではワークフロー インスタンスからこの変数を取得し、それを利用して受信ビジネス メッセージを処理します。

注意： application integration では、XOCP ビジネス メッセージは `XML Document` 型のワークフロー変数には格納されません。

2-16 ページの「ワークフロー変数の使用について」で説明されているように、Studio で、ビジネス メッセージを格納する `Java Object` 変数はそれらのメッセージと関連するアクションを定義する前に定義する必要があります。

各ワークフロー テンプレート定義では、ワークフローで送信または受信されるビジネス メッセージごとに別々の変数を定義するか、すべてのメッセージ交換で利用する 1 つのオブジェクト変数を定義できます。

Studio でビジネス ドキュメントの変数を定義するには、次の手順を行います。

1. フォルダ ツリーで、適切なワークフロー テンプレート定義の [変数] を右クリックします。[変数を作成] を選択して、[変数プロパティ] ダイアログ ボックスを表示します。

図 4-1 [変数プロパティ] ダイアログ ボックス

2. この変数のユニークな名前を指定します。
3. `Java Object` 変数型を選択します。
4. [OK] をクリックします。

単純なメッセージ操作（非推奨）

B2B Integration プラグインでは、単純なメッセージ操作に使用できる以下の 2 つのアクションが提供されます。

- Compose Business Message ― ビジネス メッセージの送信 アクションで引き続き送信できるビジネス メッセージを作成します。
- Extract Business Message Parts ― ビジネス メッセージをさらに細かく処理するために構成要素を抽出します。通常は、ワークフローでビジネス メッセージが受信された後に行います。

これら 2 つのアクションでは、プログラミングの必要なくビジネス メッセージを操作できます。以降の節では、これら 2 つのワークフロー アクションの使用方法を説明します。

ビジネス メッセージの作成

Compose Business Message アクションは、開始、タスク、分岐、イベント、完了といった協調的ワークフローのどのノードでも定義できます。Compose Business Message アクションは、明示的にワークフロー テンプレート定義に追加する必要があります。実行時に、このアクションは [Compose Business Message] ダイアログ ボックスで変数として指定された 1 つまたは複数の XML ドキュメントと 1 つまたは複数の添付ファイルをビジネス メッセージ エンベロープに追加します。[Compose Business Message] ダイアログ ボックスでは、ビジネス メッセージの送信 アクションで送信可能なメッセージを簡単に作成できます。

ビジネス メッセージ作成の概要については、以下のいずれかを参照してください。

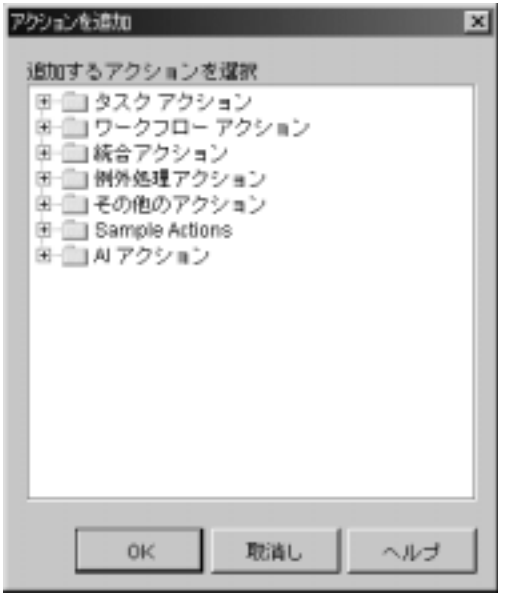
- 1-7 ページの「会話とビジネス メッセージ」
- 『*B2B Integration 入門*』の「概要」にある「ビジネス プロトコルのサポート」

Compose Business Message アクションの定義

Studio でワークフローの Compose Business Message アクションを定義するには、次の手順を行います。

1. アクションを指定できるダイアログ ボックス（[タスクのプロパティ]、[分岐のプロパティ]、[イベントのプロパティ]、[開始のプロパティ] ダイアログ ボックスなど）で、[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。

図 4-2 [アクションを追加] ダイアログ ボックス



2. [統合アクション] フォルダをクリックして展開します。

図 4-3 [アクションを追加] ダイアログ ボックスと統合アクション

3. [B2B Integration] フォルダをクリックして展開します。
4. [Compose Business Message] を選択します。
5. [OK] をクリックして、[Compose Business Message] ダイアログ ボックスを表示します。

図 4-4 [Compose Business Message] ダイアログ ボックス

6. [Compose Business Message] ダイアログ ボックスの以下のフィールドを設定します。

表 4-1 [Compose Business Message] ダイアログ ボックスのフィールド

フィールド	説明
[出力変数]	作成するビジネス メッセージを保持する `Java object` 型のワークフロー変数
パート番号	作成するビジネス メッセージの各構成要素の ID。[要素の割り当て] テーブルの各エントリにはユニークな ID が必須であり、それは正の整数でなければならない。 ID を割り当てる手順は次のとおり。 1. [要素の ID 番号] カラムで適切なセルをクリックする。 2. 要素 ID を入力する。
[要素の種類]	メッセージの要素の種類（XML ビジネス ドキュメントまたは添付ファイル）を識別する。 ■ XML ドキュメントは `XML` 型のワークフロー変数で定義しなければならない。 ■ 添付ファイルは `binary` 型で定義しなければならない。 要素の種類を選択する手順は次のとおり。 1. [要素] カラムで適切なセルをクリックする。 2. 表示されるドロップダウン リストから種類を選択する。

表 4-1 [Compose Business Message] ダイアログ ボックスのフィールド (続き

フィールド	説明
[ソース]	関連付けられている種類の要素のファイル名が格納されるワークフロー変数を識別する。 ■ 要素の種類が XML の場合、ソースは XML ビジネス ドキュメントのファイル名が格納された XML 変数でなければならない。 ■ 要素の種類が binary の場合、ソースは添付ファイルのファイル名が格納された String 変数でなければならない。 ソース ファイルを選択する手順は次のとおり。 1. [ソース] カラムで適切なセルをクリックする。 2. セルの右端に沿って表示される下矢印をクリックする。 3. 適切なメッセージ要素を保持するファイルの名前が格納された変数名を選択する。
[テキスト]	説明テキスト（省略可能）

4. 既に指定されている 2 つの要素の間にメッセージ要素を挿入するには、次のようにします。
   a. 挿入を行う行を右クリックします。
   b. [挿入] を選択します。クリックした行の上に空白の行が作成されます。
5. [Compose Business Message] ダイアログ ボックスの [要素の割り当て] テーブルのエントリを削除するには、次のようにします。
   a. 削除するメッセージ要素を指定する行をクリックします。
   b. 右クリックして [削除] を選択します。
6. [Compose Business Message] ダイアログ ボックスでの指定が終了したら、[OK] をクリックします。

ビジネス メッセージの情報の抽出

Extract Business Message Parts アクションは、開始、タスク、分岐、イベント、完了といった協調的ワークフローのどのノードでも定義できます。Extract Business Message Parts アクションは、明示的にワークフロー テンプレート定義に追加する必要があります。実行時に、このアクションは [ビジネス メッセージの要素を抽出] ダイアログ ボックスで変数として指定された XML ドキュメントと添付ファイルをビジネス メッセージ エンベロープから抽出します。[ビジネス メッセージの要素を抽出] ダイアログ ボックスでは、受信されたビジネス メッセージの内容を簡単に取得できます。

ビジネス メッセージ作成の概要については、以下のいずれかを参照してください。

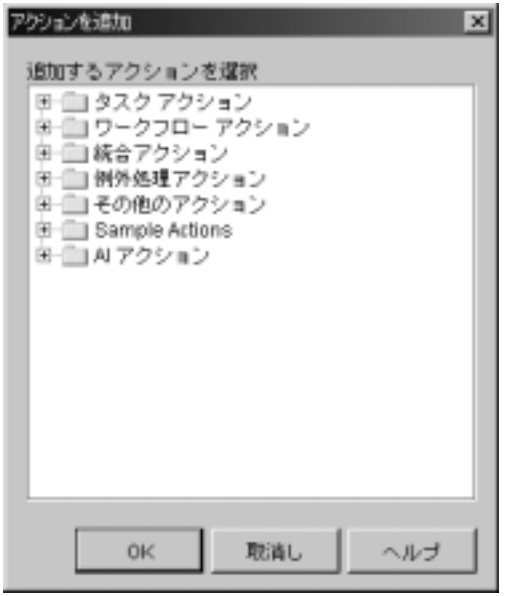
- 1-7 ページの「会話とビジネス メッセージ」
- 『*B2B Integration 入門*』の「概要」にある「ビジネス プロトコルのサポート」

Extract Business Message Parts アクションの定義

Studio でワークフローの Extract Business Message Parts アクションを定義するには、次の手順を行います。

1. アクションを指定できるダイアログ ボックス（[タスクのプロパティ]、[分岐のプロパティ]、[イベントのプロパティ]、[開始のプロパティ] ダイアログ ボックスなど）で、[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。

図 4-5 [アクションを追加] ダイアログ ボックス



2. [統合アクション] フォルダをクリックして展開します。

図 4-6 [アクションを追加] ダイアログ ボックスと統合アクション

3. [B2B Integration] フォルダをクリックして展開します。
4. [Extract Business Message Parts] を選択します。
5. [OK] をクリックして、[ビジネス メッセージの要素を抽出] ダイアログ ボックスを表示します。

図 4-7 [ビジネス メッセージの要素を抽出] ダイアログ ボックス

6. [ビジネス メッセージの要素を抽出] ダイアログ ボックスの以下のフィールドを設定します。

表 4-2 [ビジネス メッセージの要素を抽出] ダイアログ ボックスのフィールド

フィールド	説明
[入力変数]	要素を抽出するビジネス メッセージを保持する `Java object` 型のワークフロー変数
パート番号	ビジネス メッセージから抽出する各構成要素の ID。以下の点に注意。 ■ [要素の割り当て] テーブルで指定した各エントリにはユニークな ID が必要。その ID は正の整数であり、ビジネス メッセージから抽出する要素の ID と対応していなければならない。 ■ ビジネス メッセージのすべての要素を抽出する必要はなく、必要な要素だけで良い。 ID を割り当てる手順は次のとおり。 1. [要素の ID 番号] カラムで適切なセルをクリックする。 2. 要素 ID を入力する。
[要素の種類]	抽出するメッセージの要素の種類（XML ビジネス ドキュメントまたは添付ファイル）を識別する。 ■ XML ドキュメントは `XML` 型のワークフロー変数で定義しなければならない。 ■ 添付ファイルは `binary` 型で定義しなければならない。 要素の種類を選択する手順は次のとおり。 1. [要素] カラムで適切なセルをクリックする。 2. 表示されるドロップダウン リストから種類を選択する。

表 4-2 [ビジネス メッセージの要素を抽出] ダイアログ ボックスのフィールド (

フィールド	説明
[送り先]	関連付けられている種類の要素のファイル名が格納されるワークフロー変数を識別する。 ■ 要素の種類が XML の場合、送り先は XML ビジネス ドキュメントのファイル名が格納される XML 変数でなければならない。 ■ 要素の種類が binary の場合、送り先は添付ファイルのファイル名が格納される String 変数でなければならない。 送り先ファイルを選択する手順は次のとおり。 1. [送り先] カラムで適切なセルをクリックする。 2. セルの右端に沿って表示される下矢印をクリックする。 3. 抽出されたメッセージ要素を保持するファイルの名前が格納される変数名を選択する。
[テキスト]	説明テキスト（省略可能）

4. 既に指定されている 2 つの要素の間にメッセージ要素を挿入するには、次のようにします。
   a. 挿入を行う行を右クリックします。
   b. [挿入] を選択します。クリックした行の上に空白の行が作成されます。
5. [Extract Business Message Parts] ダイアログ ボックスの [要素の割り当て] テーブルのエントリを削除するには、次のようにします。
   a. 削除するメッセージ要素の ID セルをクリックします。
   b. 右クリックして [削除] を選択します。
6. [OK] をクリックします。

複雑なメッセージ操作（非推奨）

以下のタスクを行ってからでないと、MessageManipulator インタフェースを実装する Java アプリケーションでビジネス メッセージの内容を作成または抽出できません。

- 作成または操作されるビジネス メッセージの内容を保持するワークフロー変数を作成します（4-2 ページの「ビジネス メッセージのワークフロー変数の定義」を参照）。
- ワークフローの適切なノードで Manipulate Business Message アクションを定義します（4-16 ページの「Manipulate Business Message アクションの定義」を参照）。
- com.bea.b2b.wlpi.MessageManipulator インタフェースを実装し、その manipulate メソッドを使用してビジネス メッセージを処理する Java アプリケーションを記述します（4-22 ページの「ビジネス メッセージを操作するアプリケーションの記述」を参照）。

以降の節では、Manipulate Business Message アクションの定義方法および Java アプリケーションの記述方法について説明します。

Manipulate Business Message アクションの定義

実行時に、Manipulate Business Message アクションはビジネス メッセージを操作するために呼び出されます。ワークフローからビジネス メッセージ（要求など）が送信される場合、Manipulate Business Message アクションは関連付けられているアプリケーション コードを実行してビジネス メッセージを作成し、ビジネス メッセージの送信 アクションで送信される出力変数にそのメッセージを保存します。ワークフローでビジネス メッセージ（応答など）が受信される場合、Manipulate Business Message アクションは入力変数内のビジネス メッセージを取得し、それを処理するために関連付けられているアプリケーション コードに渡します。

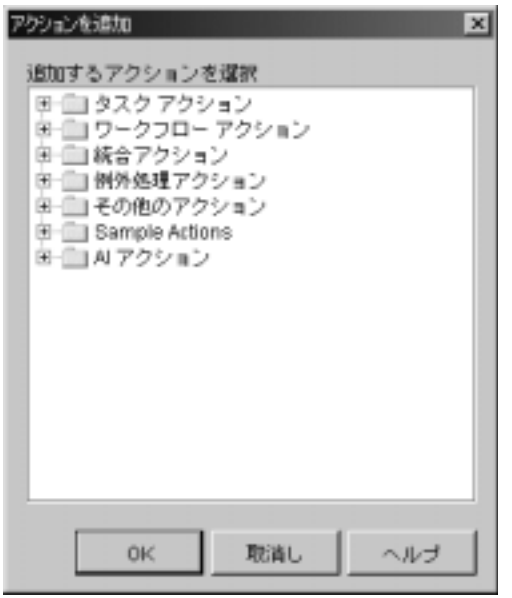
Manipulate Business Message アクションは、タスク、分岐、イベント、開始といったノードに関連付けることができます。Manipulate Business Message アクションは、明示的にワークフロー テンプレート定義に追加する必要があります。

Manipulate Business Message アクションの追加

Studio でワークフローの Manipulate Business Message アクションを定義するには、次の手順を行います。

1. アクションを指定できるダイアログ ボックス（[タスクのプロパティ]、[分岐のプロパティ]、[イベントのプロパティ]、[開始のプロパティ] ダイアログ ボックスなど）で、[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。

図 4-8 [アクションを追加] ダイアログ ボックス



2. [統合アクション] フォルダをクリックして展開します。

図 4-9 [アクションを追加] ダイアログ ボックスと統合アクション

3. [B2B Integration] フォルダをクリックして展開します。
4. [Manipulate Business Message] を選択します。
5. [OK] をクリックして、[ビジネス メッセージの操作] ダイアログ ボックスを表示します。

図 4-10 [ビジネス メッセージの操作] ダイアログ ボックス

6. [ビジネス メッセージの操作] ダイアログ ボックスの以下のフィールドで値を選択します。

表 4-3 [ビジネス メッセージの操作] ダイアログ ボックスのフィールド

フィールド	説明
[マニピュレータ クラス名]	`com.bea.b2b.wlpi.MessageManipulator` インタフェースを実装する Java クラスの名前（必須）。詳細については、4-22 ページの「ビジネス メッセージを操作するアプリケーションの記述」を参照。
[入力変数]	Receive Business Message アクションで受信されたメッセージなど、既存のビジネス メッセージが格納されたワークフロー変数の名前。 この変数の内容は `in` パラメータとして、`com.bea.b2b.wlpi.MessageManipulator` インタフェースを実装する指定された Java クラスの `manipulate` 処理に渡される。変数名が指定されていない場合、`in` パラメータの値は `null`。 指定される変数は、Java Object 型の既存のワークフロー変数と対応していなければならない。詳細については、2-16 ページの「ワークフロー変数の使用について」を参照してください。
[出力変数]	`com.bea.b2b.wlpi.MessageManipulator` インタフェースを実装する指定された Java クラスの `manipulate` 処理で返されたビジネス メッセージが格納されるワークフロー変数の名前。 指定される変数は、Java Object 型の既存のワークフロー変数と対応していなければならない。詳細については、2-16 ページの「ワークフロー変数の使用について」を参照してください。変数名が指定されていない場合、`manipulate` 処理の戻り値は無視される。
[メモ]	説明テキスト（省略可能）

入力フィールドまたは出力フィールドの変数を指定するときには、以下のガイドラインに従ってください。

- アクションが操作対象のビジネス メッセージを受信する場合は、入力変数フィールドの値を指定する必要があります。

- アクションがビジネス メッセージを送信する場合は、出力変数フィールドの値を指定する必要があります。
- 送信するビジネス メッセージを作成する場合は、メッセージ マニピュレータとともに、そのビジネス メッセージ オブジェクトを指定する出力フィールドを使用できます。

7. [OK] をクリックして変更を保存します。

Manipulate Business Message アクションの例

たとえば、[ビジネス メッセージの操作] ダイアログ ボックスで以下の設定を指定するとします。

表 4-4 [ビジネス メッセージの操作] ダイアログ ボックスのサンプル設定

フィールド	説明
[マニピュレータ クラス名]	`examples.wlpiverifier.ProcessRequest`
[入力変数]	`requestMsg`
[出力変数]	`replyMsg`

実行時に、WebLogic Integration プロセス エンジンが指定された設定でアクションを実行すると、以下のイベントが発生します。

1. `examples.wlpiverifier.ProcessRequest` クラスのオブジェクトがリフレクションとデフォルト コンストラクタを使用して作成されます。
2. `in` パラメータ（`requestMsg`）の値が取り出されます。
3. `manipulate` 処理が呼び出されます。
4. `manipulate` 処理の戻り値がワークフロー出力変数（`replyMsg`）に格納されます。

ビジネス メッセージを操作するアプリケーションの記述

トレーディング パートナ間で交換されるビジネス メッセージをワークフロー変数と Java コードを使用して操作する Java アプリケーションを記述します。Manipulate Business Message アクションでは、このアプリケーションを呼び出して、送信するビジネス メッセージを作成するか、受信したビジネス メッセージを処理します。この Java アプリケーションは、`com.bea.b2b.wlpi.MessageManipulator` インタフェースを実装します。この種のアプリケーションは、一般的にメッセージ マニピュレータと呼ばれます。

Manipulate Business Message アクションのメッセージ マニピュレータ クラスおよび入力と出力の変数を定義する方法については、4-16 ページの「Manipulate Business Message アクションの定義」を参照してください。
`com.bea.b2b.wlpi.MessageManipulator` インタフェースの詳細については、『*BEA WebLogic Integration Javadoc*』を参照してください。

メッセージ マニピュレータの機能

メッセージ マニピュレータでは、ビジネス メッセージを処理するための以下の処理を実装できます。

- ビジネス メッセージを送信前に作成します（4-24 ページの「MessageManipulator インタフェースを実装してビジネス メッセージを作成するアプリケーションの記述手順」を参照）。ワークフローは会話に参加するためにメッセージを送信する必要があります。
  - 実行時に Manipulate Business Message アクションが呼び出されます。Manipulate Business Message アクションは、Java アプリケーションを呼び出して（他のワークフロー変数の内容に基づいて）ビジネス メッセージを作成し、変数に格納するようにそのビジネス メッセージを返します。詳細については、4-16 ページの「Manipulate Business Message アクションの定義」を参照してください。
  - Send Business Message アクションは、この変数を取り出してビジネス メッセージを送信します。詳細については、5-1 ページの「ビジネス メッセージを送信するワークフローの定義」を参照してください。

- ビジネス メッセージを送信前に作成します（4-27 ページの「MessageManipulator インタフェースを実装して受信したビジネス メッセージの内容を処理するアプリケーションの記述手順」を参照）。ビジネス メッセージの受信後、呼び出されたビジネス メッセージ マニピュレータはメッセージの内容を抽出し、必要な要素を他のアクションで使用できるようにワークフロー変数に格納します。
- 受信したビジネス メッセージの特定の要素を置換し、そのメッセージを送信します。
- 独自のメッセージ抽出処理を実行します。

MessageManipulator インタフェース

会話のロール間で交換されるビジネス メッセージを処理するために、ワークフロー アプリケーションでは `com.bea.b2b.wlpi.MessageManipulator` インタフェースを実装する Java クラスを使用します。このインタフェースには、次のシグネチャを持つ `manipulate` という 1 つの処理があります。

```
XOCPMessage manipulate(WorkflowInstance instance, XOCPMessage in)
throws WLPIException;
```

`manipulate` 処理を呼び出すときに、ワークフローでは以下のパラメータを指定します。

表 4-5 manipulate 処理のパラメータ

パラメータ	説明
`instance`	変数の取得または設定に使用できる現在のワークフロー インスタンス。詳細については、4-16 ページの「Manipulate Business Message アクションの定義」を参照してください。
`in`	関連付けられている Manipulate Business Message アクションで入力変数として指定されたワークフロー変数に格納されている XOCP メッセージ。Manipulate Business Message アクションで入力変数が指定されていないか、変数が空の場合は、`null` が渡される。

manipulate 処理では、メッセージ マニピュレータで生成された XOCP メッセージが返されます。実行時に、この XOCP メッセージは関連付けられている Manipulate Business Message アクションで指定された出力変数に格納されます。この出力変数が指定されていない場合、戻り値は無視されます。

パブリック デフォルト コンストラクタ

メッセージ マニピュレータ インタフェースを実装するクラスでは、パブリック デフォルト コンストラクタ（引数のないコンストラクタ）が必要です。プロセス エンジンでは、Java リフレクションを使用してそのクラスのオブジェクトを作成し、したがってデフォルト コンストラクタを呼び出します。

MessageManipulator インタフェースを実装してビジネス メッセージを作成するアプリケーションの記述手順

この節で示す ChannelMasterMessageFactoryII クラスは、ビジネス メッセージを作成するメッセージ マニピュレータの例です。このマニピュレータは、ワークフローで発生する Manipulate Business Message アクションから呼び出されます。このマニピュレータは、送信するビジネス メッセージとしてワークフローに渡される応答メッセージ（xocpmsg 変数）を返します。

ステップ 1 : 必要なパッケージをインポートする

次のリストは、XOCP メッセージを作成するための XOCP メッセージング オブジェクトなど、ChannelMasterMessageFactoryII クラスがインポートするパッケージを示しています。

コード リスト 4-1 必要なパッケージのインポート

```
package wlcsamples.channelmaster;

import java.io.*;

import org.apache.xerces.dom.*;
import org.w3c.dom.*;

import com.bea.eci.logging.*;
import com.bea.b2b.wlpi.MessageManipulator;
import com.bea.b2b.wlpi.WorkflowInstance;
import com.bea.b2b.wlpi.WLPIException;
```

```
import com.bea.b2b.protocol.conversation.ConversationType;
import com.bea.b2b.enabler.*;
import com.bea.b2b.enabler.xocp.*;
import com.bea.b2b.protocol.messaging.*;
import com.bea.b2b.protocol.xocp.conversation.local.*;
import com.bea.b2b.protocol.xocp.messaging.*;
```

ステップ 2 : MessageManipulator インタフェースを実装する

次のリストは、`MessageManipulator` インタフェースを実装する `ChannelMasterMessageFactoryII` クラス宣言を示しています。

コード リスト 4-2 MessageManipulator インタフェースの実装

```
public class ChannelMasterMessageFactoryII implements MessageManipulator
```

ステップ 3 : デフォルト コンストラクタを実装する

次のリストは、`ChannelMasterMessageFactoryII` クラスで使用するデフォルト コンストラクタを示しています。

コード リスト 4-3 デフォルト コンストラクタの実装

```
public ChannelMasterMessageFactoryII(){};
```

ステップ 4 : manipulate メソッドの呼び出しを実装する

次のコードは、`manipulate` メソッドを呼び出します。このメソッドは、受信ビジネス メッセージだけでなく現在のワークフロー インスタンス オブジェクトを取り出します。

コード リスト 4-4 manipulate メソッドの呼び出し

```
public XOCPMessage manipulate(WorkflowInstance instance,
                                        XOCPMessage in)
    throws WLPIException{
```

ステップ 5 : ワークフロー インスタンスから入力変数を取得する

次のコードは、getVariable メソッドを使用して、ビジネス メッセージの作成に使用する現在のワークフロー インスタンスから値を取得します。

コード リスト 4-5 入力変数の取得

```
String content = (String) instance.getVariable(MESSAGE_CONTENT_VAR);
```

ステップ 6 : ビジネス メッセージの作成

次のコードは、以下の処理を実行します。

1. XML ドキュメントの作成に使用する XML データを表す DOM オブジェクトを作成します。
2. temp-xml-transporter.dtd という DTD を使用して XML ドキュメント オブジェクトを作成します。
3. XML ドキュメントの各 XML 要素を作成します。
4. XML ドキュメントを追加して XOCP メッセージを作成します。

コード リスト 4-6 XML ドキュメントの作成

```
DOMImplementationImpl domi = new DOMImplementationImpl();
DocumentType dType =
  domi.createDocumentType( "temp-xml-transporter",
    "temp-xml-transporter", "temp-xml-transporter.dtd" );

org.w3c.dom.Document rq = new DocumentImpl(dType);
Element root = rq.createElement("temp-xml-transporter");
rq.appendChild(root);
```

```
Element elementContent = rq.createElement("content");
Text t1 = rq.createTextNode(content);
elementContent.appendChild(t1);
root.appendChild(elementContent);

XOCPMessage xocpmsg = new XOCPMessage("");
xocpmsg.addPayloadPart(new BusinessDocument(rq));
```

ステップ 7 : 要求メッセージを返す

次のコードは、変数 xocpmsg（XOCPMessage 型）の要求メッセージを返します。戻り値は、ビジネス メッセージを送信するためにワークフローの出力変数（Java Object 型）に割り当てられます。

コード リスト 4-7　要求メッセージを返す

```
return xocpmsg;
```

MessageManipulator インタフェースを実装して受信したビジネス メッセージの内容を処理するアプリケーションの記述手順

この節で説明する ChannelMasterMessageExtractorII クラスは、ビジネス メッセージを受信して処理するメッセージ マニピュレータの例です。このマニピュレータは、（ビジネス メッセージで開始するように定義された）開始イベントと関連付けられている Manipulate Business Message アクションから呼び出されます。その開始イベントは、最初のビジネス メッセージが会話開始者ワークフローから受信されたときにトリガされます。このマニピュレータは、送信するビジネス メッセージとしてワークフローに渡される応答メッセージ（replyMsg 変数）を返します。

ステップ 1 : 必要なパッケージをインポートする

次のコードは、XOCP メッセージを作成するための XOCP メッセージング オブジェクトなど、ChannelMasterMessageExtractorII クラスがインポートするパッケージを示しています。

コード リスト 4-8 必要なパッケージのインポート

```
package wlcsamples.channelmaster;

import java.io.*;

import org.apache.xerces.dom.*;
import org.w3c.dom.*;

import com.bea.eci.logging.*;
import com.bea.b2b.wlpi.MessageManipulator;
import com.bea.b2b.wlpi.WorkflowInstance;
import com.bea.b2b.wlpi.WLPIException;

import com.bea.b2b.protocol.conversation.ConversationType;
import com.bea.b2b.enabler.*;
import com.bea.b2b.enabler.xocp.*;
import com.bea.b2b.protocol.messaging.*;
import com.bea.b2b.protocol.xocp.conversation.local.*;
import com.bea.b2b.protocol.xocp.messaging.*;
```

ステップ 2 : MessageManipulator インタフェースを実装する

次のリストは、MessageManipulator インタフェースを実装する ChannelMasterMessageExtractorII クラス宣言を示しています。

コード リスト 4-9 MessageManipulator インタフェースの実装

```
public class ChannelMasterMessageExtractorII implements MessageManipulator
```

ステップ 3 : デフォルト コンストラクタを実装する

次のリストは、ChannelMasterMessageExtractorII クラスで使用するデフォルト コンストラクタを示しています。

コード リスト 4-10　デフォルト コンストラクタの実装

```
public ChannelMasterMessageExtractorII(){};
```

ステップ 4 : manipulate メソッドを呼び出す

次のコードは、manipulate メソッドを呼び出します。このメソッドは、受信ビジネス メッセージ（要求）だけでなく現在のワークフロー インスタンス オブジェクトを取り出します。

コード リスト 4-11　manipulate メソッドの呼び出し

```
public XOCPMessage manipulate(WorkflowInstance instance,
                                        XOCPMessage in)
    throws WLPIException{
```

ステップ 5 : 要求メッセージを処理する

次のコードは、以下の処理を実行します。

1. ビジネス メッセージからペイロード部分を抽出します。
2. 各ペイロード部分について、DOM ドキュメントを抽出します。
3. DOM オブジェクトの DOM API を使用して、XML コンテンツを抽出します。

コード リスト 4-12　要求メッセージの処理

```
PayloadPart[] payload = in.getPayloadParts();
Document rq = null;

if (payload != null && payload.length > 0){//
  BusinessDocument bd = (BusinessDocument)payload[0];
  rq = bd.getDocument();
  if (rq == null)
    throw new WLPIException("Did not get a reply document");
}
Element root = rq.getDocumentElement();
debug(root.toString());
```

```
String name = root.getNodeName();
if (!name.equals("temp-xml-transporter")) {
  throw new WLPIException(
    debug("Did not get temp-xml-transporter, found " + name));
}
if (!root.hasChildNodes()){
  throw new WLPIException(
    debug("No child nodes in temp-xml-transporter"));
}
Node childContent = root.getFirstChild();
if (childContent == null){
  throw new WLPIException(
    debug("No child nodes inside temp-xml-transporter"));
}
String content = ((Text)childContent.getFirstChild()).getData();
debug("Content is " + content);
```

5　ビジネス メッセージの送受信

以下の節では、協調的ワークフローでビジネス メッセージを送受信する方法について説明します。

- ビジネス メッセージを送信するワークフローの定義
- ビジネス メッセージを受信するワークフローの定義

注意： RosettaNet プロトコルに基づく会話とワークフローを実装する場合、手順については『*B2B Integration RosettaNet の実装*』を参照してください。これらの手順は RosettaNet 1.1 および 2.0 プロトコルに基づいています。

このドキュメントで説明する B2B 機能でサポートしている XOCP ビジネス プロトコルは、WebLogic Integration の本リリースより廃止されています。XOCP の代替となるビジネス プロトコル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。

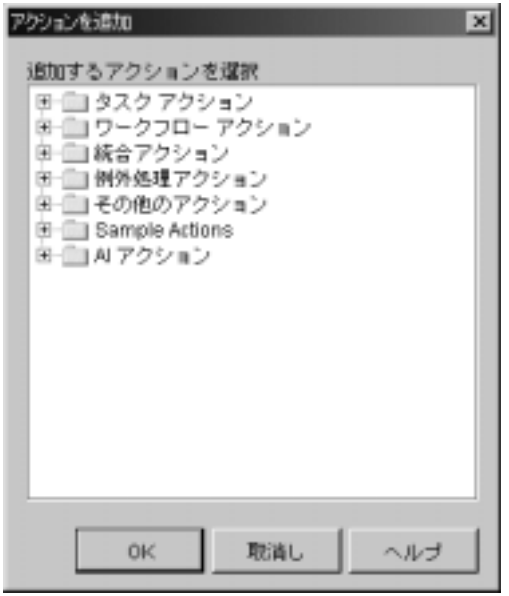
ビジネス メッセージを送信するワークフローの定義

WebLogic Integration Studio で Compose Business Message アクションまたは Manipulate Business Message アクションを使用してビジネス メッセージを作成した後は、Send Business Message アクションを使用してそのビジネス メッセージを送信します。この節では、トレーディング パートナにビジネス メッセージが送信されるように、協調的ワークフローで Send Business Message アクションを定義する方法について説明します。

Send Business Message アクションを定義するには、次の手順を行います。

1. アクションを指定できるダイアログ ボックス（[タスクのプロパティ]、[分岐のプロパティ]、[イベントのプロパティ]、[開始のプロパティ] ダイアログ ボックスなど）で、[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。

図 5-1 [アクションを追加] ダイアログ ボックス



2. [統合アクション] フォルダをクリックして展開します。

図 5-2 [アクションを追加] ダイアログ ボックスと統合アクション

3. [B2B Integration] フォルダをクリックして展開します。
4. [Send Business Message] を選択します。[OK] をクリックして、[ビジネス メッセージの送信] ダイアログ ボックスを表示します。

[ビジネス メッセージの送信] ダイアログ ボックスのどの部分が表示されるかは、ワークフロー テンプレートがコンフィグレーションされているプロトコルによって異なります。以降の節では、B2B Integration プラグインでサポートされているプロトコルごとに、このダイアログ ボックスの内容を指定する方法について説明します。

XOCP 1.1 プロトコルを使用したビジネス メッセージの送信（非推奨）

XOCP 1.1 プロトコルでコンフィグレーションされたワークフロー テンプレートの [アクション] ダイアログ ボックスで ビジネス メッセージの送信 アクションを選択すると、次のダイアログ ボックスが表示されます。

図 5-3 XOCP メッセージの [ビジネス メッセージの送信] ダイアログ ボックス

[ビジネス メッセージの送信] ダイアログ ボックスでは以下のタブが表示されます。

- [メッセージ] ― 入力メッセージ変数、ルータ式の内容、および対象ロールを指定できます。
- [トークン] ― ワークフロー変数に対するメッセージ トークン情報を指定できます（5-7 ページの「メッセージ トークン情報のワークフロー変数への割り当て」を参照）。
- [QoS] ― この ビジネス メッセージの送信 アクション レベルでサービス品質を指定できます（5-12 ページの「Send Business Message アクションでのメッセージ配信のサービス品質の定義（非推奨）」を参照）。

[ビジネス メッセージの送信] ダイアログ ボックスの以下のフィールドを設定します。

表 5-1 [ビジネス メッセージの送信] ダイアログ ボックスのフィールド

フィールド	説明
[入力メッセージ変数]	送信するビジネス メッセージが格納される `Java Object` 型のワークフロー変数の名前

表 5-1 [ビジネス メッセージの送信] ダイアログ ボックスのフィールド (続き)

フィールド	説明
[ルータ式の型]	ルータ式 フィールドの内容（トレーディング パートナ名、XPath 式、または変数名）。ルータ式は WebLogic Integration リポジトリで指定されたルータ式でオーバーライドされる場合がある。ルータの詳細については、『*B2B Integration 管理者ガイド*』の「高度なコンフィグレーション タスク」を参照。 この選択ボックスでは、以下の値を選択できる。 ■ [トレーディング パートナ名] — ルータ式 フィールドには 1 つのトレーディング パートナの名前が格納される。 ■ [XPath 式] — ルータ式 フィールドには Xpath 式が格納される。 ■ [変数名] — ルータ式 フィールドには Xpath 式の内容を持つ（`String` 型の）ワークフロー変数が格納される。変数はドロップダウン リストから選択し、その値は Receive Business Message イベントによって割り当てられている場合がある。
[ルータ式]	メッセージの送信時に使用されるルータ式。 ■ [トレーディング パートナ] が選択されている場合、メッセージは指定されたトレーディング パートナに送信される。 ■ [XPath 式] が選択されている場合、メッセージは指定された XPath 式に基づいて送信される。 このフィールドが空白の場合は、NULL フィルタが使用される。ルータ式の詳細については、『*B2B Integration 管理者ガイド*』の「高度なコンフィグレーション タスク」を参照。
[対象ロール]	メッセージが送信される会話の中のロール（必須フィールド）

表 5-1 [ビジネス メッセージの送信] ダイアログ ボックスのフィールド (続き)

フィールド	説明
[会話 QoS を使用]	ワークフロー テンプレートが XOCP プロトコルでコンフィグレーションされている場合は、このチェック ボックスが表示される。このチェック ボックスでは、テンプレート レベルまたはこの Send Business Message アクション レベルで定義されたサービス品質のどちらを使用するのかを指定できる。 ■ チェックボックスを選択すると、Application Integration ではワークフロー テンプレート定義のレベルで定義された QoS 情報が使用される（2-9 ページの「ワークフロー テンプレートでの XOCP メッセージ配信のサービス品質の定義（非推奨）」を参照）。 ■ チェック ボックスを選択しない場合、Application Integration ではこの Send Business Message アクション レベルで定義された QoS 情報が使用される（5-12 ページの「Send Business Message アクションでのメッセージ配信のサービス品質の定義（非推奨）」を参照）。
[メモ]	説明テキスト（省略可能）

メッセージ トークン情報のワークフロー変数への割り当て

ビジネス メッセージが B2B メッセージング サービスによって送信されるときには、メッセージ トークンがプログラミング レベルで Java オブジェクトとして返されます。メッセージ トークンは、メッセージについての情報を提供します。その情報には、メッセージ ID、会話 ID、送信の成否、配信ステータス、および B2B エンジンによる最終選択（ルータおよびフィルタの評価）の後の受信側数などがあります。アプリケーションでは、`getVariable` メソッドを呼び出してこの変数にアクセスします。

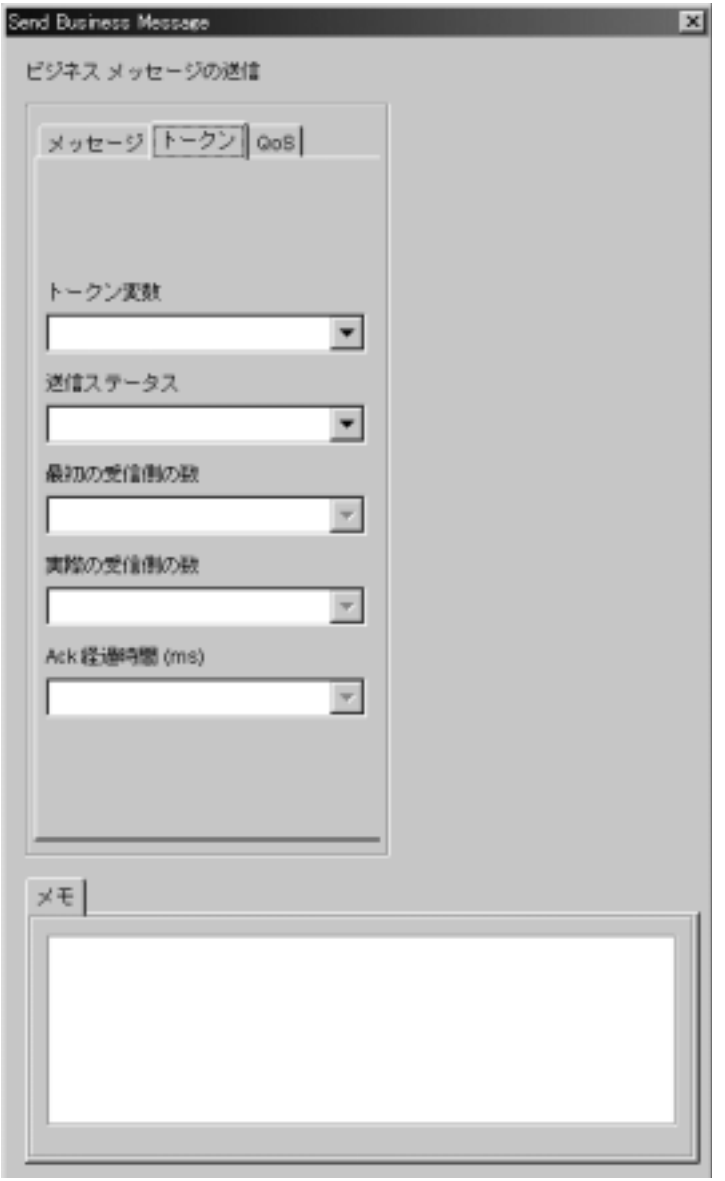
この変数は、ワークフロー終了後に処理される出力変数として定義できます。メッセージ トークンは、`com.bea.b2b.protocol.messaging.MessageToken` クラスで表されます。このクラスの詳細については、『*BEA WebLogic Integration Javadoc*』を参照してください。

協調的ワークフローは、トークンとその情報をワークフロー変数に割り当てて メッセージ トークンにアクセスできるようにコンフィグレーションできます。実行時において、ビジネス メッセージの送信 アクションが完了した後に値がワークフロー インスタンス変数に割り当てられます。

メッセージ トークンと関連情報をワークフロー変数に割り当てるには、次の手順を行います。

1. 5-1 ページの「ビジネス メッセージを送信するワークフローの定義」の説明に従って [ビジネス メッセージの送信] ダイアログ ボックスを開きます。
2. [ビジネス メッセージの送信] ダイアログ ボックスで [トークン] タブをクリックします。

図 5-4 [トークン] タブ


Send Business Message
ビジネス メッセージの送信
メッセージ
トークン
QoS
トークン変数
送信ステータス
最初の受信側の数
実際の受信側の数
Ack 経過時間 (ms)
メモ

注意： このダイアログ ボックスで利用可能なオプションは、選択したサービス品質の設定によって異なります。詳細については、2-9 ページの「ワークフロー テンプレートでの XOCP メッセージ配信のサービス品質の定義（非推奨）」を参照してください。

3. [トークン] タブの以下のフィールドを設定します。

表 5-2 [トークン] タブのフィールド

フィールド	説明
[トークン変数]	返されたメッセージ トークンを `Java Object` 型のワークフロー変数に割り当てる。このオブジェクトは、処理を目的としてビジネス オペレーションにのみ渡すことができる。
[送信ステータス]	メッセージが正常に送信されたかどうかを示す（成功の場合は true、失敗の場合は false）。その値は、ワークフローからアクセスできるか、ビジネス オペレーションに渡すことができる `Boolean` 型のワークフロー変数に割り当てる。

表 5-2 [トークン] タブのフィールド (続き)

フィールド	説明
[最初の受信側の数]	メッセージがルータを経由した後に B2B エンジンによって割り当てられた受信側の数。その値は、ワークフローからアクセスできるか、ビジネス オペレーションに渡すことができる `Integer` 型のワークフロー変数に割り当てる。 このフィールドは、[QoS メッセージ配信確認] 設定が以下のいずれかである場合にのみ表示される。 ■ [XOCP ハブまで]— メッセージが仲介機能に達したときに配信確認を必要とする（デフォルト）。このオプションを選択すると、最高の実行時パフォーマンスを維持して基本的な配信確認が提供される。 ■ [XOCP ハブ ルータまで]— メッセージが仲介機能のルータに達したときに配信確認を必要とする。このオプションでは、メッセージを受信するためにルータによって選択されたトレーディング パートナのリストが提供される。 ■ [すべての送り先]— すべての送り先から配信確認を必要とする。このオプションを選択すると、最高レベルの配信確認が提供される。実行時のパフォーマンスが低下する恐れがある。
[実際の受信側の数]	受信側の実際の数。その値は、ワークフローからアクセスできるか、ビジネス オペレーションに渡すことができる `Integer` 型のワークフロー変数に割り当てる。 このフィールドは、[QoS メッセージ配信確認] 設定が [すべての送り先] である場合にのみ表示される。
[Ack 経過時間 (ms)]	確認応答がすべての受信側から送信されるのに要する時間（ミリ秒単位）。このフィールドの値は、ワークフローからアクセスできるか、ビジネス オペレーションに渡すことができる `Long` 型のワークフロー変数に割り当てる。 このフィールドは、[QoS メッセージ配信確認] 設定が [すべての送り先] である場合にのみ表示される。

表 5-2 [トークン] タブのフィールド (続き)

フィールド	説明
[メモ]	説明テキスト（省略可能）

ビジネス メッセージが同期送信の配信オプションを使用して送信された場合、メッセージ トークンを使用して応答確認を待機することはできません。この目的で使用した場合は、メソッドが即座に返されます。

Send Business Message アクションでのメッセージ配信のサービス品質の定義（非推奨）

サービス品質（QoS）は、XOCP プロトコルを使用したビジネス メッセージの信頼性を高めるために定義する属性のセットです。QoS は、Studio を使用して以下のレベルで定義できます。

- テンプレート レベル。アクションの定義によってオーバーライドされない限り、設定はすべての Send Business Message アクションに適用されます。詳細については、2-9 ページの「ワークフロー テンプレートでの XOCP メッセージ配信のサービス品質の定義（非推奨）」を参照してください。
- Send Business Message アクション レベル。設定は特定のアクションのみに適用されますが、テンプレート レベルで指定された設定をオーバーライドします。

Send Business Message レベルで QoS を定義するには、次の手順を行います。

1. [ビジネス メッセージの送信] ダイアログ ボックスで [会話 QoS を使用] チェック ボックスが選択されていないことを確認します。
2. [ビジネス メッセージの送信] ダイアログ ボックスで [QoS] タブを選択します。

   [QoS] タブが表示されます。

図 5-5 [ビジネス メッセージの送信] ダイアログ ボックスの [QoS] タブ

3. 表 2-3 の説明に従って [QoS] タブのフィールドを設定します。
4. [OK] をクリックします。

注意： ここで指定した定義は、この会話のすべての送信アクションにではなくこの Send Message アクションにのみ適用されます。

RosettaNet ビジネス メッセージの送信

WebLogic Integration ではリリース 7.0 より、すべてのビジネス メッセージが同期的に伝送されます。ビジネス メッセージの同期的送信には次の利点があります。ビジネス メッセージを送信するタスク ノードが実行を完了したら、必要に応じて即時にワークフローのプロセスを次のノードへ進めることができます。ビジネス メッセージの返信ステータスを待っている間、ワークフローの実行を待機する必要がありません。

ビジネス メッセージを同期的に送信するワークフローを設計する場合は、次の定義を行います。

- ビジネス メッセージを（以前のように）送信するタスク ノード
- HTTP ステータス イベントのタイムアウト値、期日を指定するタスク ノード
- HTTP ステータス イベントを受け取るイベント ノード

7.0 以前のバージョンの WebLogic Integration を使用して作成された RosettaNet をベースとするワークフローをご使用の場合は、ビジネス メッセージの送信方法を変更する必要があります。既存の RosettaNet ベースのワークフローから現在の WebLogic Integration リリースへ移行する詳しい方法については、『*WebLogic Integration 移行ガイド*』を参照してください。

RosettaNet プロトコルをベースとするワークフローを作成する手順については、『*B2B Integration RosettaNet の実装*』を参照してください。

次のセクションでは、RosettaNet 1.1 および 2.0 プロトコルを使用してコンフィグレーションされたワークフローのビジネス メッセージを送信する方法について説明します。

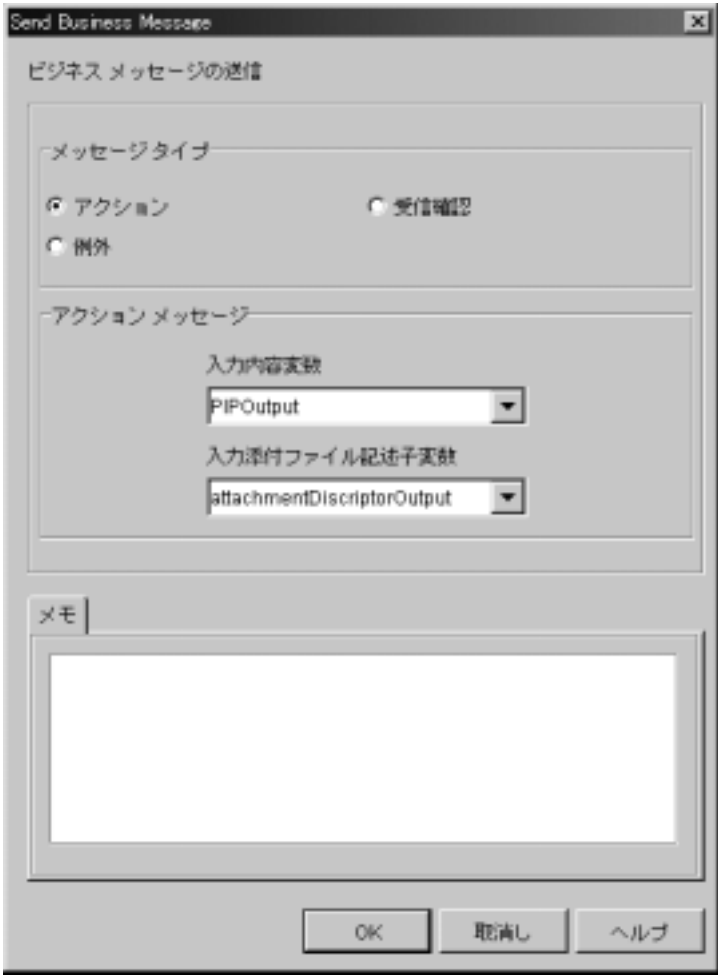
- RosettaNet 2.0 プロトコルを使用したビジネス メッセージの送信方法を指定する

- RosettaNet 1.1 プロトコルを使用したビジネス メッセージの送信方法を指定する
- HTTP ステータス イベント タイムアウト値を指定するタスク ノードを定義する
- HTTP ステータス イベントを受け取るイベント ノードを定義する

RosettaNet 2.0 プロトコルを使用したビジネス メッセージの送信方法を指定する

RosettaNet 2.0 プロトコルでコンフィグレーションされたワークフロー テンプレートのビジネス メッセージを送信するタスク ノードで、[アクション] ダイアログボックスから [ビジネス メッセージの送信] を選択します。次のダイアログボックスが表示されます。

図 5-6 RosettaNet 2.0 メッセージの [ビジネス メッセージの送信] ダイアログ ボックス



[ビジネス メッセージの送信] ダイアログ ボックスの以下のフィールドを設定します。

表 5-3 [ビジネス メッセージの送信] ダイアログ ボックスのボタンとフィールド

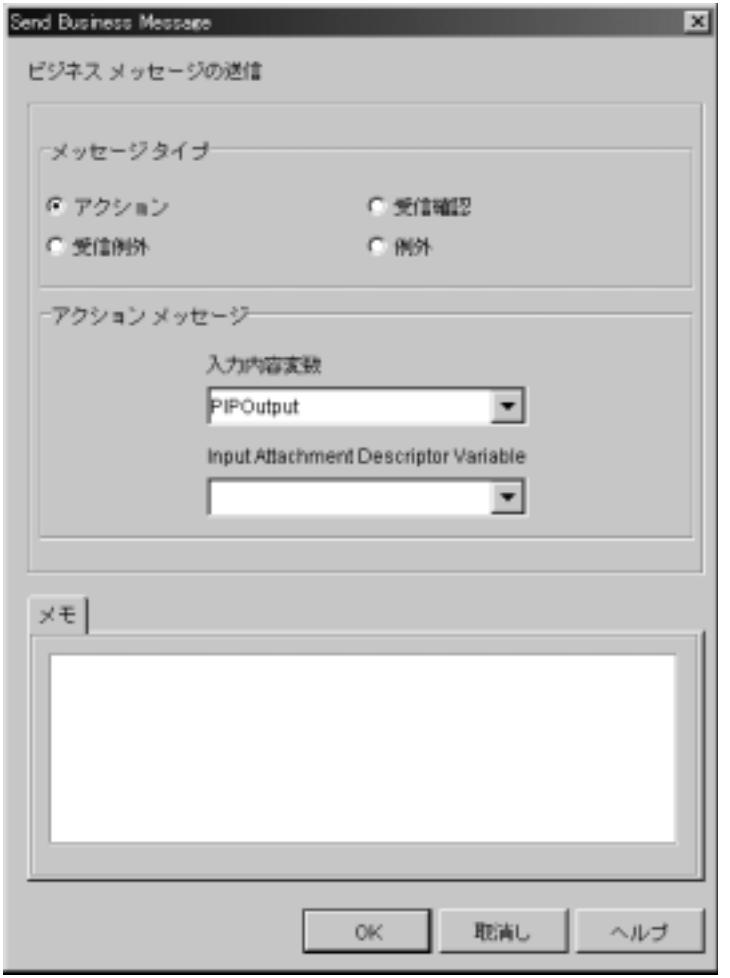
フィールド	説明
メッセージ タイプ	送信する RosettaNet 2.0 メッセージのタイプを指定する。ビジネス メッセージを送信するには、[アクション] をクリックする。
[入力内容変数]	サービス内容を表わす XML データが格納される XML 型のワークフロー変数の名前を指定する。
[入力添付ファイル記述子変数]	RosettaNet メッセージの一部として送信される添付ファイルを説明する XML データが格納される XML 型のワークフロー変数の名前を指定する。
[メモ]	説明テキスト（省略可能）

RosettaNet 2.0 メッセージの送信の詳細については、『*B2B Integration RosettaNet の実装*』を参照してください。

RosettaNet 1.1 プロトコルを使用したビジネス メッセージの送信方法を指定する

RosettaNet 1.1 プロトコルでコンフィグレーションされたワークフロー テンプレートのビジネス メッセージを送信するタスク ノードで、[アクション] ダイアログボックスから [Send Business Message] を選択します。次のダイアログ ボックスが表示されます。

図 5-7　RosettaNet 1.1 メッセージの [ビジネス メッセージの送信] ダイアログ ボックス



[ビジネス メッセージの送信] ダイアログ ボックスの以下のフィールドを設定します。

表 5-4 [ビジネス メッセージの送信] ダイアログ ボックスのボタンとフィールド

フィールド	説明
メッセージ タイプ	送信する RosettaNet 1.1 メッセージのタイプを指定する。ビジネス メッセージを送信するには、[アクション] をクリックする。
[入力内容変数]	サービス内容を表わす XML データが格納される XML 型のワークフロー変数の名前を指定する。
[入力添付ファイル記述子変数]	RosettaNet メッセージの一部として送信される添付ファイルを説明する XML データが格納される XML 型のワークフロー変数の名前を指定する。
[メモ]	説明テキスト（省略可能）

RosettaNet 1.1 メッセージの詳細情報や例については、『*B2B Integration RosettaNet の実装*』を参照してください。

HTTP ステータス イベント タイムアウト値を指定するタスク ノードを定義する

HTTP ステータス イベント受信のタイムアウト値を指定するタスク ノードを定義するには、次のステップを実行します。

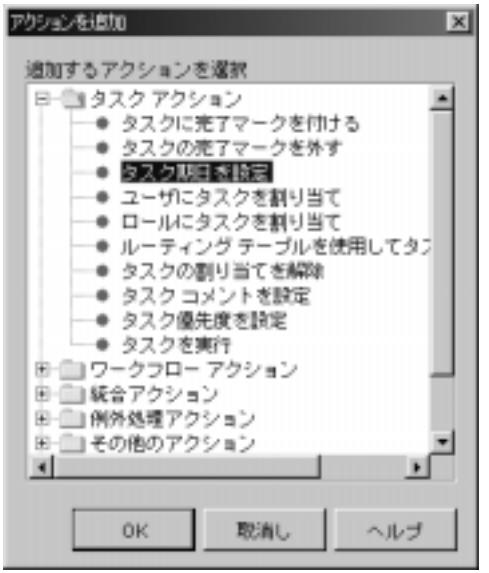
1. Studio ツールバーで、[タスク作成] をクリックし、次にビジネス メッセージを送信するタスク ノードの近くでワークフロー ダイアグラムの任意の位置をクリックします。
2. 作成したタスク ノードをダブルクリックします。

   [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 5-8 [タスクのプロパティ] ダイアログ ボックス

3. [タスク名] フィールドに内容を表わすタスク名をつけます。
4. ウィンドウの [アクション] セクションで [作成時] 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。

図 5-9 [アクションを追加] ダイアログ ボックス



5. [タスク アクション] という名前のフォルダを右クリックし、[タスク期日を設定] を選択します。[タスク期日を設定] ダイアログ ボックスが表示されます。

図 5-10 [タスク期日を設定] ダイアログ ボックス

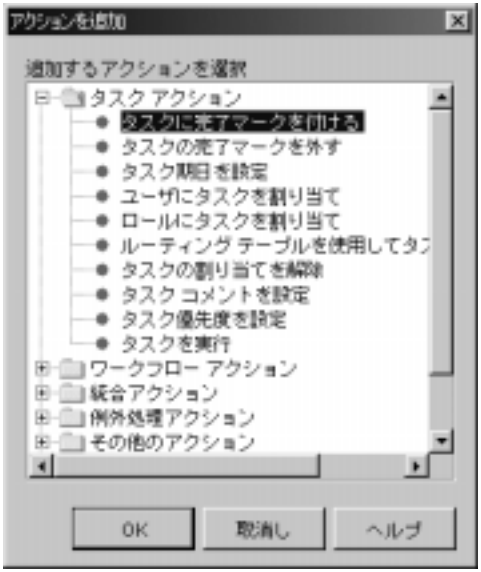
6. [タスク期日を設定] ダイアログ ボックスの上にあるメニューから [Wait for timeout] を選択します。
7. ダイアログ ボックスの下の方にある [期日] タブを選択します。[式に設定] フィールドに適切なタイムアウト値を入力します。たとえば図 5-10 では、タイムアウト値は 2 時間に設定されています。

   タスク期日の指定に関する詳細については、『*WebLogic Integration Studio ユーザーズガイド*』の「アクションの定義」にある「タスク期日を設定する」を参照してください。
8. 次の図で示されている [期日に実行するアクション] タブを選択します。

図 5-11 [期日に実行するアクション] タブ

9. [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。
10. [アクションを追加] ダイアログ ボックスで [タスク アクション] フォルダを拡張し、次の図にあるように [タスクに完了マークを付ける] を選択します。

図 5-12 [タスクに完了マークを付ける] アクション



11. [OK] をクリックします。次の図に示す [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 5-13 [タスクに完了マークを付ける] ダイアログ ボックス

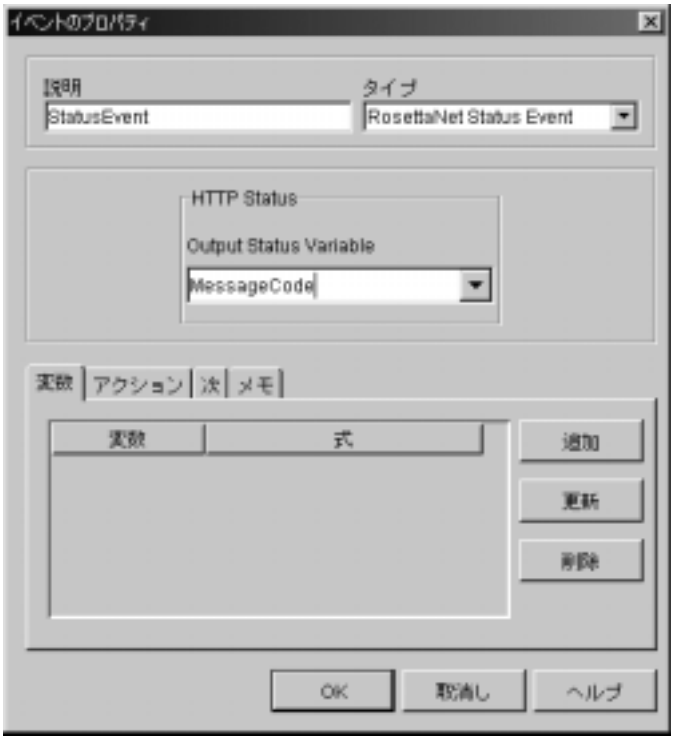
12. 現在のタスクを選択し、[OK] をクリックします。
13. [タスク期日を設定] ダイアログ ボックスで [OK] をクリックします。
14. [タスクのプロパティ] ダイアログ ボックスで [OK] をクリックます。
15. ワークフローを保存します。

HTTP ステータス イベントを受け取るイベント ノードを定義する

HTTP ステータス イベントを受け取るイベント ノードを作成するには、次のステップを実行します。

1. Studio ツールバーで、[イベント作成] をクリックし、次にビジネス メッセージを送信するタスク ノードの近くでワークフロー ダイアグラムの任意の位置をクリックします。
2. [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 5-14 [イベントのプロパティ] ダイアログ ボックス



3. [イベントのプロパティ] ダイアログ ボックスで、[説明] フィールドに、たとえば StatusEvent のように適切なイベント名を入力します。
4. [タイプ] フィールドで [RosettaNet Status Event] を選択します。
5. [Output Status Variable] ドロップダウン リストから、HTTP ステータス値の格納に適した変数を選択します。
6. [アクション] タブから [追加] を選択します。

   [アクションを追加] ダイアログ ボックスが表示されます。
7. [タスク アクション] フォルダを開き、[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。

8. 5-25 ページの「HTTP ステータス イベントを受け取るイベント ノードを定義する」で作成したタスクを選択し、[OK] をクリックします。
9. [イベントのプロパティ] ダイアログ ボックスで [OK] をクリックます。

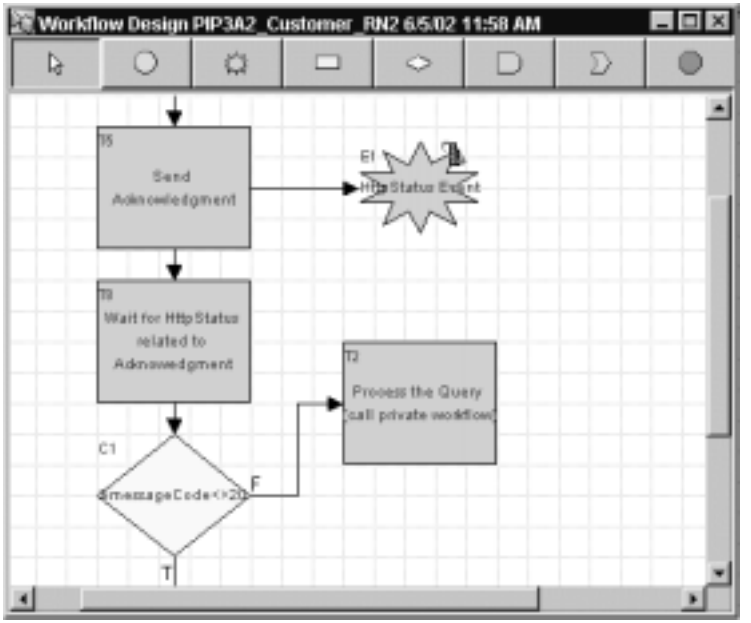
[ビジネス メッセージの送信] に関連付けられているノードを接続する

[ビジネス メッセージの送信] タスクに関連付けられているノードを定義したら、次は接続です。Studio には、ワークフローのノードを接続する方法が 2 つあります。

- Studio のツールバーから
  a. [接続] ボタンをクリックします。
  b. 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。
- 特定のノードには [プロパティ] ダイアログ ボックスから
  a. 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。[プロパティ] ダイアログ ボックスが表示されます。
  b. [次] タブで、接続先のノードを選択します。

次の図は、RosettaNet ベースのワークフローにおける、同期的 ビジネス メッセージの送信 タスクと 4 つのノード（T5、T8、C1 および E1）の関連付けの例です。

図 5-15 ビジネス メッセージの送信 ワークフロー



[ビジネス メッセージの送信] タスクに関連付けられているノードの定義と接続が終了したら、タイムアウト値を指定するタスクノードから分岐ノードに移行します。分岐ノードでは、HTTP ステータスを評価し、必要に応じて実行のフローを定義します。

ビジネス メッセージを受信するワークフローの定義

ワークフローは、以下の状況でビジネス メッセージを受信できます。

- 会話参加者ワークフローが、会話開始者ワークフローから送信される最初のビジネス メッセージを待っているとき。最初のビジネス メッセージは、ビジネス メッセージの開始として定義された会話の開始ノードを開始します。詳

細については、3-6 ページの「会話参加者ワークフローの開始ノードの定義」を参照してください。

- 会話開始者ワークフローまたは会話参加者ワークフローが、イベント ノードで別のメッセージ（要求に対する応答など）を待っているとき（5-29 ページの「Business Message Receive イベントの定義」を参照）。

以降の節では、B2B Integration プラグインでサポートされている各プロトコルで、ビジネス メッセージを受信するようにワークフローを設定する手順を説明します。

Business Message Receive イベントの定義

ワークフローがビジネス メッセージ（要求に対する応答や、それ以降の要求など）を受信するために待機する場合は、Business Message Receive イベントを定義する必要があります。このイベントは、実行時において適切なビジネス メッセージが会話で受信されたときにトリガされます。

Business Message Receive イベントを定義するには、次の手順を行います。

1. 『*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」の説明に従ってイベント ノードを表示または追加します。
2. イベント シェイプをダブルクリックするか、フォルダ ツリーでイベント シェイプを右クリックし、[プロパティ] コマンドを選択して [イベントのプロパティ] ダイアログ ボックスを表示します。

図 5-16 [イベントのプロパティ] ダイアログ ボックス

3. [説明] フィールドで、Receive Business Message イベントの意味のわかる名前を入力します。
4. [タイプ] ドロップダウン リストで [Collaboration event] を選択します。

[イベントのプロパティ] ダイアログ ボックスの表示が更新されます。どの内容が表示されるかは、ワークフロー テンプレートがコンフィグレーションされているプロトコルによって異なります。以降の節では、B2B Integration プラグインでサポートされているプロトコルごとに、Receive Business Event を定義する方法を説明します。

XOCP 1.1 プロトコルでの Business Message Receive イベントの定義

ワークフローテンプレートが XOCP 1.1 プロトコルでコンフィグレーションされている場合は、[Conversation event] タイプを選択すると、[イベントのプロパティ] ダイアログ ボックスが次のように更新されます。

図 5-17 XOCP メッセージ受信イベントを定義するための [イベントのプロパティ] ダイアログ ボックス

XOCP メッセージの Business Message Receive イベントをコンフィグレーションするには、[イベントのプロパティ] ダイアログ ボックスの以下のフィールドで値を選択します。

表 5-5 [イベントのプロパティ] ダイアログ ボックスのフィールド

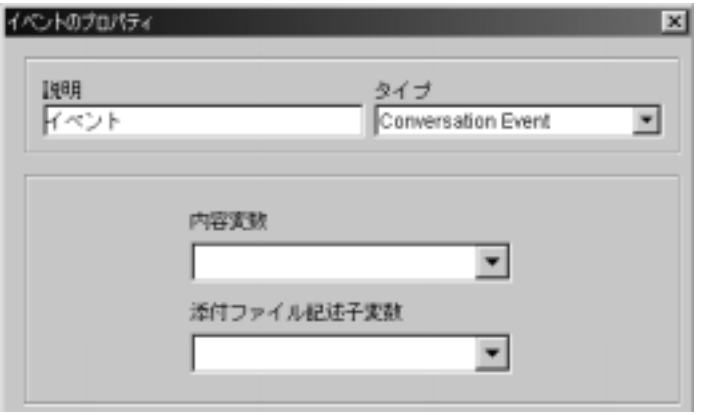
変数	説明
[イベントのタイプ]	[着信ビジネス メッセージ] を選択して、受信ビジネス メッセージを待つイベントを定義する。会話終了の方法の選択については、6-5 ページの「会話参加者ワークフローの終了の定義」を参照。
[対象変数]	ビジネス メッセージを格納する `Java Object` 型の対象ワークフロー変数の名前（必須フィールド）
[ルータ式]	XPath 式を格納する `String` 型のワークフロー変数の名前。その値は、メッセージを送信するために送信側で使用される XPath 式を表す。この XPath 式は、現在のメッセージに対する応答を発行するためにルータ式で使用できる。このフィールドはオプション。
[送信側の名前]	メッセージを送信したトレーディング パートナの名前を格納する `String` 型のワークフロー変数の名前。[Xpath 変換] チェック ボックスが選択されている場合、この名前は XPath 式に変換される。
[XPath 変換]	このチェック ボックスが選択されている場合、[送信側の名前] フィールドで指定された変数の内容が Send Business Message アクションでの使用に適した XPath 式に変換される。 選択されていない場合は、[送信側の名前] 変数は送信側の トレーディング パートナの実際の名前として使用されます。

実行時にビジネス メッセージが受信されると、イベントがトリガされて、対象変数が受信したばかりのビジネス メッセージに設定されます。ルータ変数が指定されている場合、その変数には送信側への応答で使用できる XPath 式が格納されます。詳細については、5-1 ページの「ビジネス メッセージを送信するワークフローの定義」を参照してください。

RosettaNet 2.0 プロトコルでの Business Message Receive イベントの定義

ワークフロー テンプレートが RosettaNet 2.0 プロトコルでコンフィグレーションされている場合は、[Conversation event] タイプを選択すると、[イベントのプロパティ] ダイアログ ボックスの表示が次のように更新されます。

図 5-18 RosettaNet 2.0 メッセージ受信イベントを定義するための [イベントのプロパティ] ダイアログ ボックス



RosettaNet 2.0 メッセージの Business Message Receive イベントをコンフィグレーションするには、[イベントのプロパティ] ダイアログ ボックスの以下のフィールドで値を選択します。

表 5-6 [イベントのプロパティ] ダイアログ ボックスのフィールド

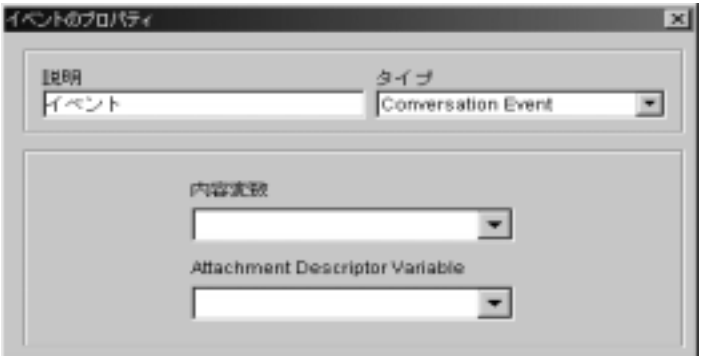
変数	説明
[内容変数]	RosettaNet 2.0 XML ビジネス ドキュメントの内容を格納する XML 型の対象ワークフロー変数の名前
[添付ファイル記述子変数]	RosettaNet 2.0 メッセージの添付ファイルの内容を格納する XML 型の対象ワークフロー変数の名前

実行時にビジネス メッセージが受信されると、イベントがトリガされて、対象変数が受信したばかりのビジネス メッセージに設定されます。RosettaNet 2.0 プロトコルでコンフィグレーションされたワークフロー テンプレートの Business Message Receive イベントを定義する方法の詳細情報と例については、『*B2B Integration RosettaNet の実装*』を参照してください。

RosettaNet 1.1 プロトコルでの Business Message Receive イベントの定義

ワークフロー テンプレートが RosettaNet 1.1 プロトコルでコンフィグレーションされている場合は、[Conversation event] タイプを選択すると、[イベントのプロパティ] ダイアログ ボックスの表示が次のように更新されます。

図 5-19 RosettaNet 1.1 メッセージ受信イベントを定義するための [イベントのプロパティ] ダイアログ ボックス



RosettaNet 1.1 メッセージの Business Message Receive イベントをコンフィグレーションするには、RosettaNet 1.1 XML ビジネス ドキュメントの内容を格納する XML 型の対象ワークフロー変数の名前を選択します。

実行時にビジネス メッセージが受信されると、イベントがトリガされて、対象変数が受信したばかりのビジネス メッセージに設定されます。RosettaNet 1.1 プロトコルでコンフィグレーションされたワークフロー テンプレートの Business Message Receive イベントを定義する方法の詳細情報と例については、『*B2B Integration RosettaNet の実装*』を参照してください。

6　協調的ワークフローの終了

以下の節では、協調的ワークフローの終了と関連する主要なタスクについて説明します。

- 会話終了の定義
- ワークフロー終了の定義

注意： この章で説明する機能の多くは、WebLogic Integration の本リリースから廃止された XOCP ビジネス プロトコルをベースにしています。RosettaNet プロトコル ベースの会話の終了については、『*B2B Integration RosettaNet の実装*』を参照してください。ebXML プロトコル ベースのワークフローの終了については、『*B2B Integration ebXML の実装*』を参照してください。XOCP の代替となるビジネス プロトコル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。

会話終了の定義

会話は、会話開始者ワークフローが完了状態に達したときに終了します。会話参加者ワークフローは、会話が終了する前にその会話への参加を終了できます。

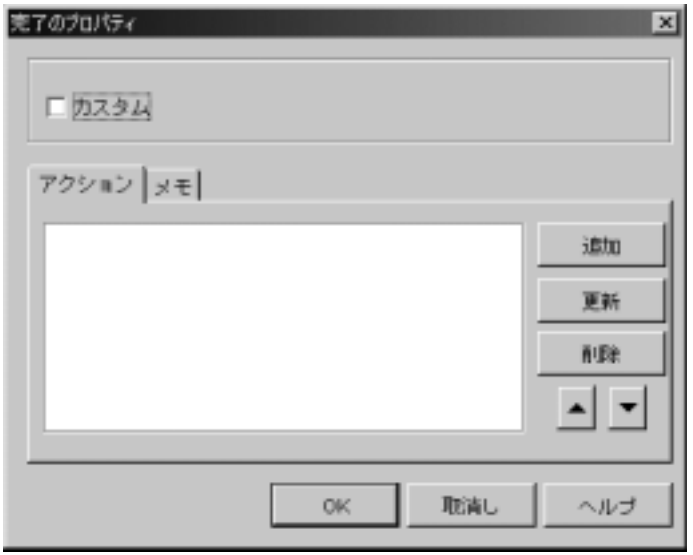
会話開始者ワークフローの終了の定義

会話開始者ワークフローについては、ワークフローの任意の完了ノードで会話終了プロパティ（成功または失敗での終了）を定義できます。ワークフローで完了ノードに達すると、アクティブなワークフローがすべての完了ノードに達しているかどうかに関係なく、ワークフローの実行中のインスタンスに完了のマークが付けられます。会話開始者ワークフローは会話を終了できますが、会話の他の参加者は終了できません。

会話開始者ワークフローの終了を定義するには、次の手順を行います。

1. 『*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」の説明に従って完了シェイプを追加または表示します。
2. 完了シェイプをダブルクリックするか、フォルダ ツリーでそれを右クリックします。[プロパティ] を選択して、[完了のプロパティ] ダイアログ ボックスを表示します。

図 6-1 [完了のプロパティ] ダイアログ ボックス



3. [カスタム] をクリックします。Studio でインストールされているプラグインごとのドロップダウン リストが表示されます。

図 6-2 [カスタム完了のプロパティ] ドロップダウン リスト

4. [Collaboration Termination] を選択します。

[完了のプロパティ] ダイアログ ボックスの内容は、ワークフロー テンプレートで定義されているプロトコルによって異なります。

サポートされている各プロトコルの完了プロパティ

以下の節では、サポートされている各プロトコルの [完了のプロパティ] ダイアログ ボックスを説明します。

- XOCP 1.1 プロトコルの [完了のプロパティ] ダイアログ ボックス（非推奨）
- RosettaNet 1.1 または 2.0 プロトコルの [完了のプロパティ] ダイアログ ボックス

XOCP 1.1 プロトコルの [完了のプロパティ] ダイアログ ボックス（非推奨）

XOCP 1.1 プロトコルで定義されているワークフロー テンプレートでは、[会話終了] プロパティを使用してカスタム完了プロパティを選択した後に次の図のような [完了のプロパティ] ダイアログ ボックスが表示されます。

図 6-3 XOCP 1.1 プロトコルの [完了のプロパティ] ダイアログ ボックス

[完了のプロパティ] ダイアログ ボックスで会話終了オプションを選択します。

表 6-1 [完了のプロパティ] ダイアログ ボックスの交換終了オプション

フィールド	説明
[完了]	このオプションは、会話が結果 SUCCESS で終了する場合に選択する（デフォルト）。会話は、この状態のアクションが完了した後に終了する。 結果 SUCCESS は、ワークフロー インスタンスが正常に完了したことを示す。会話の参加者には、会話が終了することが通知される（可能な場合）。
[失敗]	このオプションは、会話が結果 FAILURE で終了する場合に選択する。会話は、この状態のアクションが完了した後に終了する。 結果 FAILURE は、ワークフロー インスタンスで特定の会話またはアプリケーションのエラーが発生したことを示す。会話の参加者には、会話が終了することが通知される（可能な場合）。

RosettaNet 1.1 または 2.0 プロトコルの [完了のプロパティ] ダイアログ ボックス

RosettaNet 1.1 または 2.0 プロトコルでワークフローがコンフィグレーションされている場合、会話終了はワークフローの完了ノードで処理されません。[会話終了] プロパティを選択すると、次のメッセージが表示されます。

現在の会話設定では、この機能は利用できません。

RosettaNet ベースの会話の終了については、『*B2B Integration RosettaNet の実装*』を参照してください。

会話参加者ワークフローの終了の定義

会話参加者ワークフローでは、会話プロパティ、ビジネス メッセージ開始プロパティ、および（XOCP ベースのワークフローで必要に応じて）会話終了イベントが定義されています。会話終了イベントは、会話開始者からの会話終了信号を待つために参加者ワークフローで使用できます。会話終了イベントを使用すると、参加者ワークフローでは会話終了のステータスに基づいて追加処理（ハウスキーピング処理など）を実行できます。

注意： このイベントの使用はオプションであり、XOCP プロトコルでのみ使用できます。このイベントを待たないワークフローは、ワークフローを終了するだけで会話から退出できます（完了ノード）。

ワークフロー イベント シェイプは、通知ノードを表します。ワークフローは、イベントをトリガする会話終了メッセージを待ちます。そのトリガの時点で、イベントで定義されているアクションを実行したり、ワークフロー変数を設定したりできます。

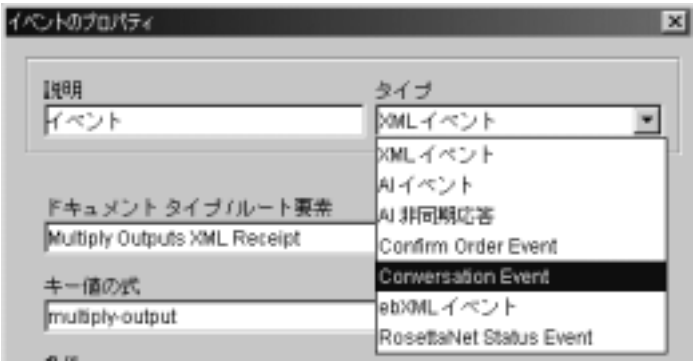
会話参加者ワークフローに会話終了イベントを追加するには、次の手順を行います。

1. 『*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」の説明に従ってイベント シェイプを表示または追加します。
2. イベント シェイプをダブルクリックするか、フォルダ ツリーでそれを右クリックします。[プロパティ] コマンドを選択して、[イベントのプロパティ] ダイアログ ボックスを表示します。

図 6-4 [イベントのプロパティ] ダイアログ ボックス

3. [説明] フィールドにテキストを入力し、[タイプ] ドロップダウン リストから [Conversation Event] を選択します。

図 6-5 コラボレーション イベントの選択



[イベントのプロパティ] ダイアログ ボックスでイベント タイプとして [Conversation Event] を選択した後、[イベントのプロパティ] ダイアログ ボックスの表示が次の図のように更新されます。

図 6-6　XOCP 1.1 プロトコルの会話コラボレーション終了イベント

4. [イベントのタイプ] 選択ボックスで [会話終了] を選択します。
5. 終了ステータス変数を格納できる Boolean 型のワークフロー変数を選択します。この変数は、次の表のいずれかの値に設定されます。

表 6-2 [イベントのプロパティ] ダイアログ ボックスの終了ステータス オプション

オプション	説明
True	会話が SUCCESS 値で終了したことを示す。
False	会話が FAILURE 値で終了したことを示す。

注意 : この Boolean 変数は、このダイアログ ボックスで選択する前に明示的に作成する必要があります。詳細については、2-16 ページの「ワークフロー変数の使用について」を参照してください。

6. [OK] をクリックして変更を保存します。

Studio では、会話終了ステータスの値が Boolean 型のワークフロー変数に割り当てられます。この変数は、ワークフローからアクセスするか、ビジネス オペレーションに渡すことができます。ワークフローの開発者は、この値に基づいて適切なアクションを行う必要があります。

ワークフロー終了の定義

すべての協調的ワークフローでは、少なくとも 1 つの完了ノードが必要です。会話開始者ワークフローで完了ノードを定義する方法については、6-1 ページの「会話開始者ワークフローの終了の定義」を参照してください。会話参加者ワークフローの場合、会話終了イベントの受信はワークフローの終了の定義とまったく関連がありません。通常、会話参加者ワークフローにはただ単にワークフローを終了するだけの完了ノードが少なくとも 1 つあります。プロトコルに関係なく、会話参加者ワークフローの完了ノードでコンフィグレーションできる B2B 固有のアクションやイベントは存在しません。

6-5 ページの「会話参加者ワークフローの終了の定義」で説明したように、会話参加者ワークフローでは会話終了イベントを処理する必要がありません。それらのワークフローは単純に終了でき、終了すると、参加していた会話から自動的に切り離されます。

完了ノードの設定に関する詳細については、『*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」を参照してください。

索引

R

S

T

W

X

あ

い

え

お

か

き

さ

し

せ

そ

た

て

と

な

に

は

ひ

ふ