BEA WebLogic Integration™

Creating Workflows for B2B Integration

Release 2.1 Service Pack 1
Document Date: January 2002

Creating Workflows for B2B Integration

Part Number	Date	Software Version
N/A	January 2002	2.1 Service Pack 1

Contents

3. Starting Collaborative Workflows

4. Working with Business Messages

5. Sending and Receiving Business Messages

6. Ending Collaborative Workflows

Index

About This Document

This document describes how to create the collaborative workflows that are a fundamental building block of the conversations defined in a collaboration agreement. To create collaborative workflows, you use the B2B integration plug-in. This extends the capabilities of the WebLogic Integration Studio, allowing you to create collaborative workflows that are bound to conversation definitions.

Note: The ebXML business protocol uses a conversation model that is simpler than the one presented here. Therefore, if you are using only the ebXML business protocol we recommend that you consult the ebXML documentation directly instead of using this document. For more information about creating workflows for the ebXML business protocol, see "Using Workflows with ebXML" in *Implementing ebXML for B2B Integration*.

This document is organized as follows:

- Chapter 1, "About Creating Workflows," provides an introduction to the B2B integration plug-in. It also describes basic concepts of the WebLogic Integration Studio and explains how the Studio ties into conversation definitions.
- Chapter 2, "Defining Conversation Properties for Workflow Templates," describes how to define templates for collaborative workflows meant to implement a role in a conversation definition.
- Chapter 3, "Starting Collaborative Workflows," explains how to configure the start properties of collaborative workflows and how to start collaborative workflows at run time.
- Chapter 4, "Working with Business Messages," explains how to assign, to tasks and events in a workflow, action properties that trigger the composition or extraction of data in the business documents that are exchanged between trading partners. This chapter also explains how to work with the message manipulator classes provided with the Message Manipulator API to compose or manipulate the contents of business messages.

- Chapter 5, “Sending and Receiving Business Messages,” explains how to assign, to tasks and events in a workflow, action properties that trigger the sending and receiving of business messages.
- Chapter 6, “Ending Collaborative Workflows,” explains how to handle the termination of conversations and end collaborative workflows.

What You Need to Know

This document is intended primarily for:

- Business analysts who use WebLogic Integration Studio to design collaborative workflows that implement the trading partner roles defined for a conversation in the WebLogic Integration environment.
- System administrators who set up and administer WebLogic Integration applications.

For an overview of the WebLogic Integration architecture, see *Introducing BEA WebLogic Integration*.

How to Print this Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the WebLogic Integration documentation CD. You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format.

If you do not have the Adobe Acrobat Reader installed, you can download it for free from the Adobe Web site at `http://www.adobe.com/`.

Related Information

For information about business process management, see the following documents:

- *Learning to Use BPM with WebLogic Integration*
- *Using the WebLogic Integration Studio*
- *Programming BPM Client Applications*
- *Using the WebLogic Integration Worklist*

Contact Us!

Your feedback on the WebLogic Integration documentation is important to us. Send us e-mail at **docsupport@bea.com** if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the WebLogic Integration documentation.

In your e-mail message, please indicate that you are using the documentation for the WebLogic Integration 2.1 release.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Examples*: LPT1 SIGNON OR

Convention	Item
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

`...`	Indicates one of the following in a command line:
	■ That an argument can be repeated several times in a command line
	■ That the statement omits additional optional arguments
	■ That you can enter additional parameters, values, or other information
	The ellipsis itself should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 About Creating Workflows

The following sections describe key concepts for using workflows in B2B integration applications:

- Architectural Overview
- Key Concepts
- Run-Time Prerequisites
- Summary of Workflow Integration Tasks

Architectural Overview

Note: The ebXML business protocol uses a conversation model that is simpler than the one presented here. Therefore, if you are using only the ebXML business protocol we recommend that you consult the ebXML documentation directly instead of using this document. For more information about creating workflows for the ebXML business protocol, see "Using Workflows with ebXML" in *Implementing ebXML for B2B Integration*.

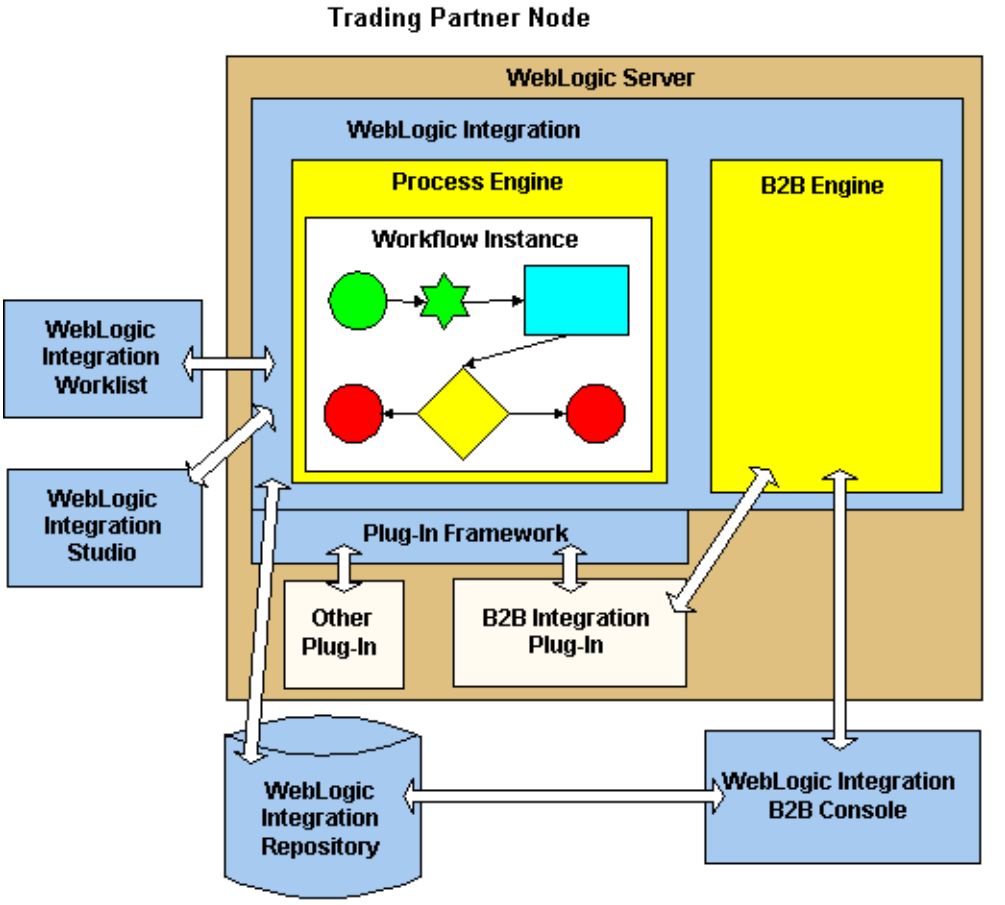
A fundamental step in defining a conversation is creating the workflow that executes each role in the conversation. The B2B integration component provides a plug-in that allows you to create such *collaborative workflows* (also known as *public workflows*). By using the B2B integration component of WebLogic Integration together with the business process management (BPM) component, you have access to the following:

- A visual design tool for designing workflows (process models), which reduces the time needed to develop business collaborations
- A run-time process engine for executing workflows
- Process monitoring capabilities

Creating workflows using the B2B integration plug-in requires a combination of design and administrative tasks, and optionally a set of programming tasks.

Architectural Components

The following figure shows how the WebLogic Integration architecture incorporates the B2B integration plug-in.



WebLogic Integration Components

The WebLogic Integration architecture includes the following components to facilitate the creation and execution of collaborative workflows.

Table 1-1 WebLogic Integration Components

Component	Description
Process Engine	Run-time controller and workflow engine that executes and manages workflows and tracks workflow instances.
Workflow Instance	A running instance of a workflow template definition, which implements a role of a conversation definition that is configured in the WebLogic Integration repository. A workflow instance is created when a workflow template definition is instantiated at run time.
WebLogic Integration Studio	Client application that is used at design time to define workflows and at run time to monitor running workflows.
WebLogic Integration Worklist	Client application that is used to view and perform tasks that are currently assigned to a user or to roles to which the user belongs. Examples of such include reassigning tasks to other users, marking tasks as done, unmarking tasks done, viewing a workflow status, manually starting a workflow, and so on.
WebLogic Integration Repository	Database in which the following are stored: ■ Workflow templates and instances. ■ Conversation definition information used by WebLogic Integration at run time, such as information about conversations, parties, conversation roles, and protocol bindings. The repository can reside locally on the trading partner node, or it can reside on a different node that is network accessible to the WebLogic Integration software. The repository can be deployed so that it is accessible to a single trading partner in one organization or to multiple trading partners in different organizations.
B2B Engine	Software that processes and routes messages among trading partners at run time.
WebLogic Integration B2B Console	Browser-based component that enables you to create, configure, administer, and monitor collaboration agreements, conversations, delivery channels, document exchanges, trading partners, and more.

Table 1-1 WebLogic Integration Components (Continued)

Component	Description
WebLogic Integration Plug-In Framework	Component that allows a specific set of BEA software—namely, the B2B integration, data integration, and application integration components of WebLogic Integration, as well as the BEA Java Application-to-Mainframe (JAM) product—and user-written software to extend the functionality of the WebLogic Integration Studio and the process engine.
B2B Integration Plug-In	The B2B integration plug-in that: ■ Extends the Studio to allow you to specify workflow nodes, actions, events, and other details that are geared toward creating collaborative workflows—that is, workflows that implement individual roles in a conversation based on one of the following business protocols: – XOCP – RosettaNet – cXML ■ Integrates the process engine with conversation flow.
Other Plug-Ins	The WebLogic Integration plug-in framework supports the incorporation of additional plug-ins, as explained in the preceding description of the WebLogic Integration Plug-In Framework.

For an introduction to the Studio, Worklist, process engine, and plug-in framework components, see "Introduction to the WebLogic Integration Studio" in *Using the WebLogic Integration Studio*.

Key Concepts

This section describes key concepts that you need to understand before using collaborative workflows in B2B integration applications.

Workflows, Collaborative Workflows, Workflow Templates, and Workflow Template Definitions

This section describes the following key business process management (BPM) concepts:

- WebLogic Integration provides a workflow automation tool in the process engine. A *workflow* is a business process. Workflow automation is the automation of a business process, in whole or in part, during which information of any type is passed to the right participant at the right time according to a set of intelligent business rules that allow computers to perform most of the work while humans have to deal only with exceptions.
- A *collaborative workflow* is a workflow that implements the role for a trading partner in a conversation definition. A conversation definition may have two or more roles, each of which is associated with a collaborative workflow.

  Collaborative workflows that implement public processes are often referred to as *public workflows*. Strictly speaking, a collaborative workflow is a public workflow only if the workflow has not been customized to execute private processes. However, for the sake of simplicity, the terms collaborative workflow and public workflow are considered interchangeable in this document. (In the Studio user interface, text that refers to the B2B integration plug-in refers only to the term public workflow. For example, the action that you define in a parent workflow to start a collaborative workflow is called Start Public Workflow.)

  For more details about public and private business processes, see "Overview" in *Introducing B2B Integration*.
- A *workflow template* is a folder (or a container) in the WebLogic Integration Studio. This workflow template represents a workflow and is given a meaningful workflow name, such as Order Processing or Billing. The workflow template

aggregates various definitions (or versions) of its implementation; these are referred to as workflow template definitions. Further, a workflow template is responsible for controlling which organizations can use the constituent workflow template definitions.

- A *workflow template definition* is a definition (or version) of the workflow, distinguished by its Effective and Expiry dates. At design time, you use the Studio to define the properties that bind a workflow template definition to a role (such as *buyer* or *seller*) in a conversation. At run time, the process engine starts an instance (or session) of a workflow template definition, selecting the most effective (or current and active) definition.

For detailed information about these concepts, see "Introduction to the WebLogic Integration Studio" in *Using the WebLogic Integration Studio*.

Conversations and Business Messages

This section defines the following key B2B integration concepts:

- A *conversation* is a series of business message exchanges between trading partners that is predefined by a conversation definition. The receipt of a business message by a trading partner can cause any number of back-end processes to occur.
- A *business message* is the basic unit of communication in a conversation between trading partners. A business message consists of one or more business documents and, optionally, attachments.

  A *business document* is the XML-based payload part of a business message. An XML business document is normally associated with a corresponding document definition associated with it.

  One or more *attachments* may also be included with the business message and may or may not be written in XML; for example, an attachment can be written in binary format.

To construct outgoing business messages or to process incoming business messages, a workflow uses the Compose Business Message, Extract Business Message Parts, or Manipulate Business Message action, as appropriate.

For detailed information about these concepts, see *Introducing B2B Integration.*

Initiators and Participants

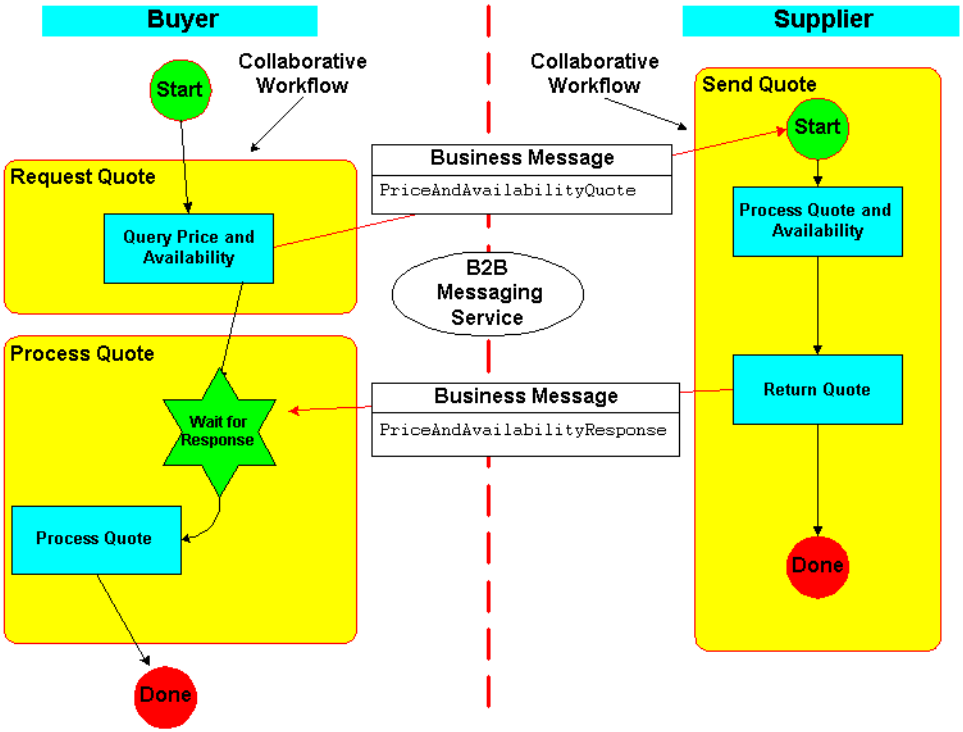
A conversation involves an *initiator* who starts the conversation and *participants* who participate in the conversation once it has started. Each perspective requires a different kind of collaborative workflow.

Table 1-2 Types of Workflows

Workflow Type	Description
Conversation initiator workflow	Defined to have conversation properties and a nonbusiness message start property. This type of workflow initiates and terminates the conversation.
Conversation participant workflow	Defined to have conversation properties and a business message start property. This type of workflow can join and exit the conversation but cannot initiate or terminate it.

In the context of a conversation, these two types of workflows are interlocking. For example, suppose a buyer wants to obtain bids from various suppliers. The resulting conversation might proceed as shown in the following figure.

Figure 1-1 Sample Business Collaboration with Two Workflows



1. In the Studio, the buyer (the initiating trading partner) starts a collaborative workflow instance (the business collaboration initiator workflow). This workflow constructs and sends the `PriceAndAvailabilityQuote` business message (containing a bid request in the form of an XML document) by way of the B2B engine to one or more qualified sellers (depending on the business protocol) and awaits a reply.

   Note: A workflow can be started in various ways; for example, programmatically or by a separate, local workflow.

2. Each qualified supplier (a participant trading partner) receives the business message, which triggers the start of an instance of a workflow (the conversation participant workflow) on that supplier's node.

   The collaborative workflow that implements the supplier role processes the incoming bid request, determines whether to submit a bid or not and, if it

decides to submit a bid, constructs and sends the `PriceAndAvailabilityResponse` business message (containing a bid reply in the form of an XML document), and ends the workflow. This workflow instance has a business message start property; that is, this workflow is started when the buyer's `PriceAndAvailabilityQuote` business message is received at the supplier's node.

Note: The collaborative workflow that implements the supplier role is defined with conversation properties and a Business Message Start property.

3. On the buyer's side, the workflow receives bid replies from all qualified suppliers, processes the results of the bid, and then terminates the conversation.

Sending and Receiving Business Messages

When trading partners exchange business messages, both initiator and participant workflows typically send and receive business messages.

It is important to keep in mind which parts of the workflow send business messages and which parts receive them. For example, a buyer might submit a bid request (a business message) to a seller. In this case, the buyer workflow is sending the business message and the seller workflow is receiving it. When the seller replies to the request with a bid (another business message), then the roles are reversed: the seller workflow is the sender and the buyer is the recipient workflow.

The design tasks differ depending on whether the workflow sends or receives business messages. However, in both cases, you must define certain properties in the workflow template definition. The B2B integration plug-in offers two means to work with business messages:

- Workflow actions that can compose or extract information from business messages that do not require any user-written application code to manipulate the business messages themselves

- Workflow actions that invoke application code that implements the `com.bea.b2b.wlpi.MessageManipulator` interface to either compose or extract information from business messages, which may be appropriate in situations where complex message handling is required.

For more information, see Chapter 4, "Working with Business Messages."

Run-Time Prerequisites

Before messages can be exchanged at run time, you must make sure the following prerequisites are met:

- WebLogic Integration is installed and configured, as described in "Administrative Tasks" on page 1-12.
- Workflows are defined and linked to conversation definitions, as described in "Design Tasks" on page 1-13.
- You have written and tested application code for manipulating messages and for starting the conversation initiator workflow, as described in "Programming Tasks" on page 1-15.
- For all trading partners, WebLogic Integration is configured and running.
- For each trading partner, all relevant workflows are active and stored in the WebLogic Integration repository:
  - For an initiating trading partner, the conversation initiator workflow is active and defined with the nonbusiness message start property. It awaits the invocation of the application that starts the workflow.
  - For all participating trading partners, the conversation participant workflow is active and defined with a Business Message start property. It awaits the receipt of the initial business message in the conversation.

Summary of Workflow Integration Tasks

Using collaborative workflows to exchange business messages in WebLogic Integration requires a combination of administrative, design, and programming tasks.

Administrative Tasks

To create collaborative workflows, you must complete the following administrative tasks:

1. Install WebLogic Integration as described in *Installing BEA WebLogic Integration*

   Note: The Samples realm in WebLogic Integration is preconfigured as described in the following step.

2. Configure WebLogic Integration according to the instructions in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

   Note: We strongly recommend that you run the WebLogic Integration samples after you install WebLogic Integration. By running the samples, you can verify that the installation has been successful. For more information about running the sample applications, see *Learning to Use BEA WebLogic Integration* and *Running the B2B Integration Samples*.

3. Using the B2B Console, create and configure the necessary entities in the WebLogic Integration repository, including collaboration agreements, delivery channels, trading partners, document exchanges, and so on. For more information, see *Administering B2B Integration*.

   Note: Every collaborative workflow template definition requires a conversation definition.

4. Using the Studio, specify the organizations, users, and roles in the WebLogic Integration repository, as described in "Administering Data" in *Using the WebLogic Integration Studio*.

Design Tasks

To create collaborative workflows, complete the following design tasks in the Studio:

1. Create and design collaborative workflows that implement each role in a conversation definition, in one of the following ways:
   - You can create workflows from scratch, as described in "Defining Workflow Templates" in *Using the WebLogic Integration Studio*.
   - Alternatively, you can import workflows created in previous versions of WebLogic Integration, as described in *Migrating to BEA WebLogic Integration Release 2.1*.
2. In addition to defining the standard workflow properties, you must also define properties that link the workflow to a conversation, which you do via the B2B integration plug-in. (The remaining tasks in this procedure apply to integrating workflows into conversations.)
3. For each workflow template, specify conversation properties as follows:
   - Explicitly link the workflow template to a role in a conversation definition in the WebLogic Integration repository, as described in "Linking Templates to Conversations" on page 2-6.
   - Optionally, specify other conversation properties, as described in "Defining the Quality of Service for XOCP Message Delivery in the Workflow Template" on page 2-9.
4. For each workflow template definition, define start actions depending on the type of workflow:
   - For conversation initiator workflows, define a nonbusiness message start property, as described in "Starting a Conversation Initiator Workflow" on page 3-3.
   - For conversation participant workflows, define a business message start property, as described in "Defining the Start Node for a Conversation Participant Workflow" on page 3-5.
5. For each workflow template definition, define how the workflow will end. To do so, add a done shape and define its properties. Conversation initiator workflows typically terminate the conversation in the Done node, as described in "Defining the Termination of Conversation Initiator Workflows" on page 6-1.

Note that conversation participant workflows do not have a custom Done node definition.

6. Define any input or output variables used in the workflow, or in applications that interact with the workflow, in the workflow template definition, as described in "About Using Workflow Variables" on page 2-15.

7. For each workflow template definition, define how business messages are processed:

   - For all collaborative workflows, define the workflow variables that are used to store business messages, as described in "About Using Workflow Variables" on page 2-15.
   - For all collaborative workflows, define task nodes that manipulate business messages, either by creating business messages to send or by processing business messages that are received, as described in Chapter 4, "Working with Business Messages."
   - For workflows that send business messages, define the Send Business Message action, as described in "Defining the Workflow to Send Business Messages" on page 5-1.
   - For conversation participant workflows, define the Start node as a business message start so that the collaborative workflow is started upon receipt of the initial business message from the conversation initiator workflow, as described in "Defining the Start Node for a Conversation Participant Workflow" on page 3-5. In addition, add an associated business message manipulation action to process the incoming business message.
   - For noninitial business messages received by conversation initiator or conversation participant workflows, define an Incoming Business Message Conversation Event, as described in "Defining Business Message Receive Events" on page 5-17. In addition, you must add an associated manipulate business message action to process the incoming business message.

8. Define how the conversation will terminate. The method of termination is protocol specific:

   - Conversation initiator workflows for XOCP-based conversations typically define a custom Done node that terminates the conversation. When an XOCP-based conversation initiator workflow sends a collaboration termination event to the intermediary (routing proxy), the intermediary sends a conversation termination event to all conversation participants.

- RosettaNet-based workflows define their own means to terminate conversations. For more information, see *Implementing RosettaNet for B2B Integration*.
- Conversation participant workflows for XOCP-based conversations may optionally have an event node that listens for an end-of-conversation event, which gives the workflow an opportunity to perform any housekeeping operations before passing control to a Done node. For more information about defining an end-of-conversation listener event node in a conversation participant workflow, see "Defining the End of Conversation Participant Workflows" on page 6-4.

Note: You can run workflows in the WebLogic Integration environment even if they are not integrated with B2B features. For example, you can run workflows created in the Studio without specifically adapting them to integrate with B2B functionality.

Programming Tasks

Programming tasks depend on the specific needs of each application that makes use of a workflow. The following tasks are required:

- For conversation initiator workflows that are not instantiated via the Start Public Workflow action defined for a parent workflow, write an application that starts the workflow. This application uses the `com.bea.b2b.wlpi.WorkflowInstance` class to pass information about the conversation to which the workflow is bound, such as the following:
  - Conversation definition name and version
  - Conversation role of the initiator
  - The identities of one or more parties that participate in the conversation

  This information allows the workflow to become bound to the unique collaboration agreement required for the conversation. The application can perform other tasks, such as assigning values to workflow variables, but it must, at a minimum, pass information about the conversation to which the workflow is bound. For more information, see "Developing Applications that Start Conversation Initiator Workflows" on page 3-19.

Note: For information about starting a workflow as a subworkflow, see "Defining the Start Public Workflow Action" on page 3-9.

- For both conversation initiator workflows and conversation participant workflows that require complex message handling, write application code that implements the `com.bea.b2b.wlpi.MessageManipulator` interface. Using this class, applications invoked by the workflow can perform complex manipulation, extraction, receiving, and sending processes on the business messages that a workflow sends and receives. For more information, see "Complex Message Manipulation" on page 4-15.

CHAPTER

2 Defining Conversation Properties for Workflow Templates

The following sections describe how to define templates and template definitions for workflows that implement roles in conversation definitions:

- About Using the B2B Integration Plug-In to Define Templates
- About Templates and Template Definitions
- Creating a Workflow Template
- Linking Templates to Conversations
- Linking Templates to Collaboration Protocols
- Logging Messages
- About Using Workflow Variables

About Using the B2B Integration Plug-In to Define Templates

Note: The ebXML business protocol uses a conversation model that is simpler than the one presented here. Therefore, if you are using only the ebXML business protocol we recommend that you consult the ebXML documentation directly instead of using this document. For more information about creating workflows for the ebXML business protocol, see "Using Workflows with ebXML" in *Implementing ebXML for B2B Integration*.

To create workflows for conversation roles in WebLogic Integration, you use the B2B integration plug-in. The first step in creating these workflows is defining workflow templates and template definitions for those workflows. In addition to the standard properties for templates and template definitions described in "Defining Workflow Templates" in *Using the WebLogic Integration Studio*, you must define additional properties not described in that document that allow the workflows based on those template definitions to be used in conversations.

For example, suppose you link a workflow template definition to a particular role in a conversation definition and version that is bound to a collaboration agreement in the WebLogic Integration repository. You must also define some additional attributes, such as the protocol to which the conversation role implemented by the workflow is bound, and additional configuration data that might be required by the protocol.

For general information about defining workflow templates, see "Defining Workflow Templates" in *Using the WebLogic Integration Studio*.

About Templates and Template Definitions

The topic "Defining Workflow Templates" in *Using the WebLogic Integration Studio* contains a comprehensive discussion about templates and template definitions, specifically about the differences between the types of information you provide in the template and the template definition.

The only data required in a template definition that you cannot define in a template is the expiration date and whether the template definition is active. However, it is good practice to set all conversation properties in the template, such as the following:

- Conversation name and version
- Conversation role
- Conversation protocol

Then, when you create a template definition, you have the flexibility of overriding any conversation properties defined in the template, as appropriate. However, the primary advantage to assigning these properties in the template instead of the template definition is that you can specify properties for all template definitions contained in a given template.

Notes: Due to a limitation in WebLogic Integration, template properties are not exported when you export a template definition. That is, the properties that a template definition inherits from a template are not exported. You should consider your export requirements when you determine the properties to define in a template versus a template definition.

Template definition properties are exportable, but are specific to only one template definition.

The sections that follow show how to define conversation properties in the template; however, it is possible to define all these properties in the template definition, instead.

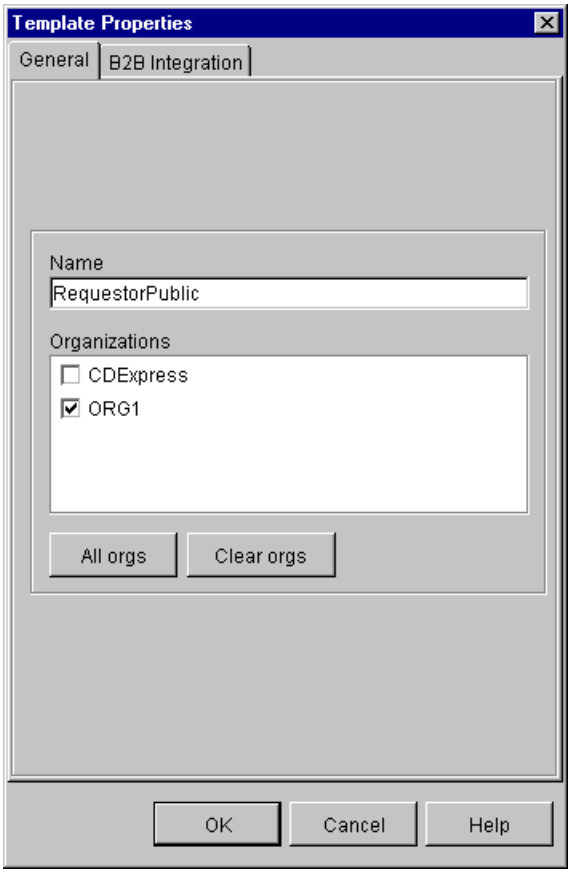
Creating a Workflow Template

To create a workflow template in the Studio using the B2B integration plug-in, complete the following steps:

1. Do one of the following:
   - To create a new template, right-click the template that will contain the new template in the folder tree, and select Create Template from the pop-up menu.
   - To open an existing template, right-click the template in the folder tree and select Open from the pop-up menu.

2. Right-click the template name and select Properties from the pop-up menu to display the Template Properties dialog box, shown in the following figure.

Figure 2-1 Template Properties Dialog Box



3. Complete the fields on the General page, as described in "Defining Workflow Templates" in *Using the WebLogic Integration Studio*.

4. Click OK to save your changes.

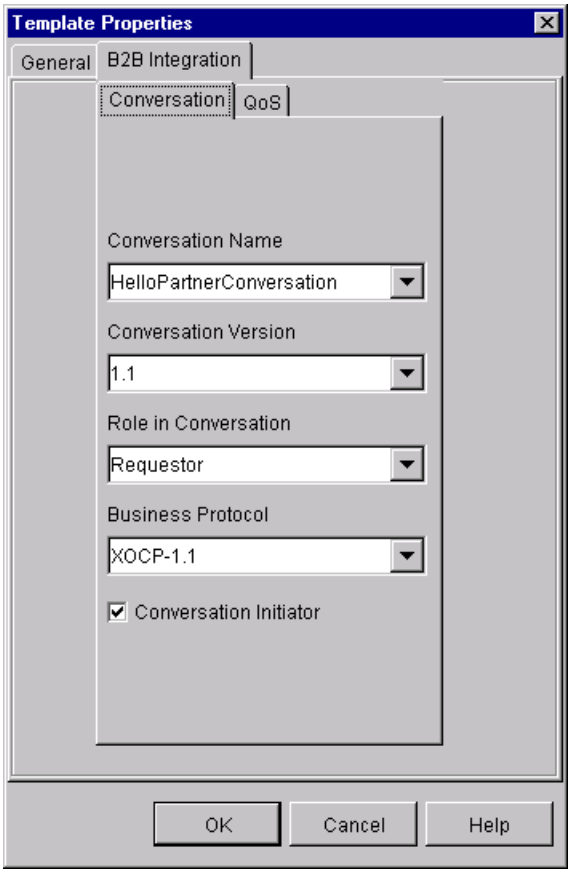
Linking Templates to Conversations

Before you use a collaborative workflow to exchange business messages in WebLogic Integration, you must first link the workflow template to a particular conversation definition (a conversation name, version, role, and protocol) in the repository.

To link a workflow template to a conversation definition:

1. Open the Template Properties dialog box, as described in "Creating a Workflow Template" on page 2-4.
2. In the Template Properties dialog box, select the B2B Integration tab, shown in the following figure.

Figure 2-2 B2B Integration Page



3. Complete the following fields on the page titled Conversation.

Table 2-1 Fields on the Conversation Page

Field	Description
Conversation Name	Name of the conversation definition in the repository to link with this workflow template definition.
Conversation Version	Version number of the conversation definition in the repository to link with this workflow template definition.

Table 2-1 Fields on the Conversation Page (Continued)

Field	Description
Role in Conversation	Role in the conversation definition to link with this workflow template definition. In order for a trading partner to receive messages in this conversation, it must be registered in this role in the conversation at run time.
Business Protocol	Business protocol for the conversation definition linked with this workflow template definition. The following business protocols are available: ■ XOCP-1.1 ■ RosettaNet-2.0 ■ RosettaNet-1.1 If you choose XOCP, an additional subtab becomes available so you can choose Quality of Service (QoS) settings for the business messages exchanged via the XOCP protocol. If you choose one of the RosettaNet protocols, see *Implementing RosettaNet for B2B Integration* for information about how to define conversation properties for workflows used in RosettaNet-based conversations.
Conversation Initiator	Select this option if the workflow starts a conversation.

4. Click OK to save your changes.

Linking Templates to Collaboration Protocols

In general, linking a template to a specific protocol affects what you see in many of the B2B integration plug-in dialog boxes and tabs throughout the Studio. As this document explains how to use various components of the B2B integration plug-in to perform various tasks, such as defining workflow start properties, sending and receiving

business messages, and terminating conversations, you will see how the appearance of each plug-in component varies, depending on the protocol chosen for the workflow template.

For example, Table 2-1 explains that if you choose the XOCP protocol while configuring a workflow template, an additional tab is displayed in which you can specify Quality of Service (QoS) settings for business messages exchanged in workflows based on that template.

With respect to linking a template to a protocol, do the following:

- If you link a workflow template to one of the supported RosettaNet protocols, click OK in the Template Properties dialog box. The task of defining the template is complete.
- If you link the template to the XOCP protocol, you can optionally specify QoS settings in the QoS tab that is displayed, as explained in the following section.

Defining the Quality of Service for XOCP Message Delivery in the Workflow Template

The Quality of Service (QoS) is a set of attributes that are defined for reliable business message publishing. You can set QoS attributes only for business messages sent via the XOCP protocol.

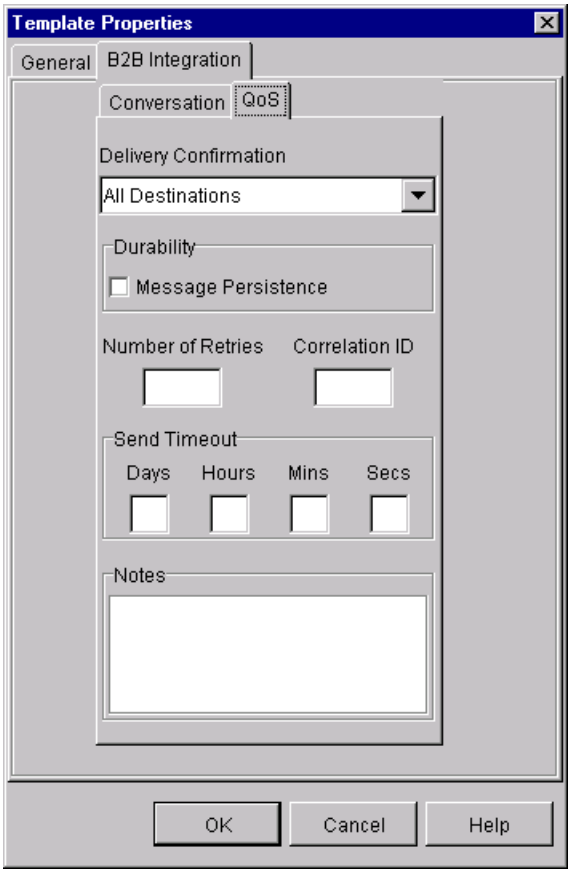
In the Studio, you can define the QoS in the following:

- In the workflow template, where the settings apply to all Send Business Message actions, unless the QoS is specifically overridden by the definition of the action.
- In the workflow template definition, where the settings also apply to all Send Business Message actions unless specifically overridden by the definitions of the action.
- In the Send Business Message action, where the settings apply to the specific action only, but override the settings specified in either the template or the template definition. For more information, see “Defining the Quality of Service for Message Delivery for a Send Business Message Action” on page 5-10.

To specify the Quality of Service in the workflow template:

1. Select the tab labeled QoS, shown in the following figure.

Figure 2-3 Quality of Service Properties Page



2. Complete the following fields on the Quality of Service page.

Table 2-2 Quality of Service Properties

Field	Description
Delivery Confirmation	Degree to which message delivery confirmation is required: Up to XOCP Hub (the default), Up to XOCP Hub Router, or All Destinations. Your selection determines which options are available in the Message Token Assignments dialog box, as described in "Assigning Message Token Information to Workflow Variables" on page 5-6. You can choose one of the following values: ■ Up to XOCP Hub—Delivery confirmation is required when a message reaches the intermediary (default). Select this option to provide basic delivery confirmation with maximum run-time performance. ■ Up to XOCP Hub Router—Delivery confirmation is required when a message reaches the router in the intermediary. This option provides the list of trading partners selected by the router to receive the message. ■ All Destinations—Delivery confirmation is required from all destinations. Select this option to provide the maximum delivery confirmation details. May affect run-time performance.
Durability	If you select the box labeled Message Persistence, messages are saved in a persistent state. This option increases the likelihood of recovery from a system failure but requires additional processing that might affect run-time performance. If Message Persistence is not selected, messages are not saved in a persistent state. This option improves run-time performance but reduces the likelihood of recovery from a system failure.
Number of Retries	Maximum number of retries for sending a message (default is 0). The WebLogic Integration process engine repeatedly attempts to send a message until it either successfully sends the message or exceeds the maximum number of retries. An exception is thrown if the maximum number of retries is exceeded.

Table 2-2 Quality of Service Properties (Continued)

Field	Description
Correlation ID	Message identification string that can be used to correlate the message with other business messages in the application (default is none). For example, a trading partner might want to specify a correlation ID in a request so that replies to that request can be matched to the original request. The B2B messaging service includes this property with the message.
Send Timeout	Timeout value in days, hours, minutes, and seconds for sending a message (default is 0, which means no timeout). The WebLogic Integration process engine waits until either a delivery confirmation is received or the timeout period is exceeded.
Notes	Optional descriptive text.

3. Click OK to save your settings.

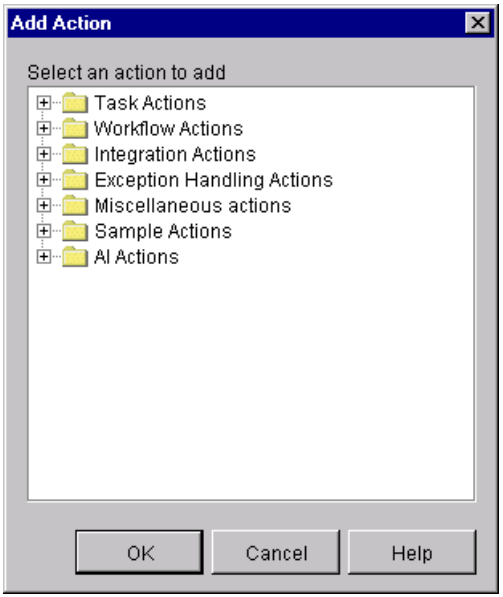
Logging Messages

The B2B integration plug-in provides a means to send messages— for example, debug messages, run-time information, or application messages—to the B2B log at any point during the execution of a collaborative workflow. This functionality is provided via the Log action, which you can add to any node in the workflow. This action can provide a useful debugging tool during the workflow design process.

To add a Log action to a node in the workflow, complete the following steps:

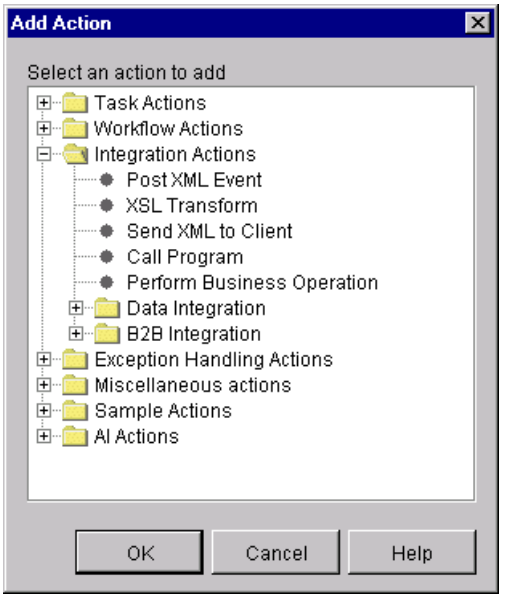
1. Open the workflow in which you want to add the Log action, making sure that the workflow is active and not expired.
2. In any dialog box where you can specify an action, click Add to display the Add Action dialog box.

Figure 2-4 Add Action Dialog Box



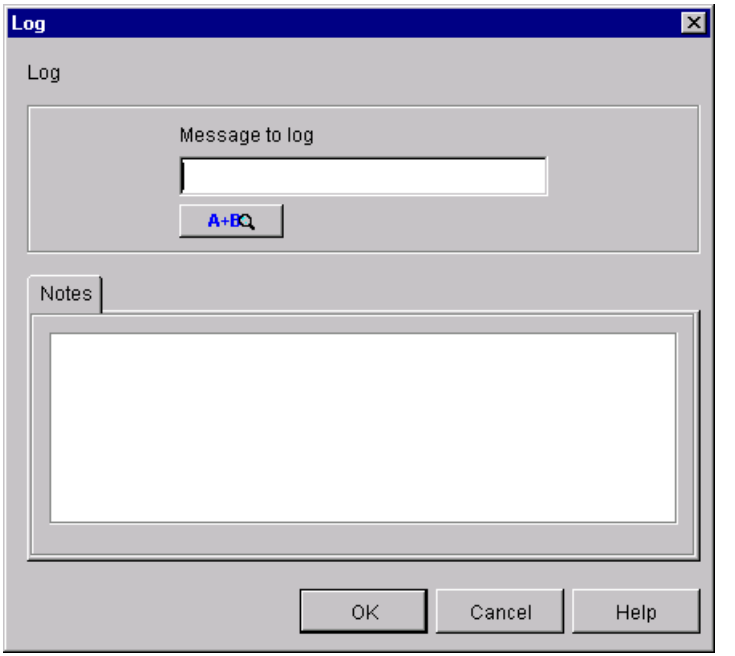
3. Click the Integration Actions folder to expand it.

Figure 2-5 Add Action Dialog Box with Workflow Actions



4. Click the B2B Integration folder to expand it.
5. Select Log. The Log action dialog box is displayed.

Figure 2-6 Log Action Dialog Box



6. Enter the message you want to send to the B2B log, or use the Expression Builder to build the message to be sent to the log. For information about using the Expression Builder, see "Using Workflow Expressions" in *Using the WebLogic Integration Studio*.

About Using Workflow Variables

A workflow variable is typically used to store application-specific information required by the workflow at run time. Variables are created and assigned values largely to control the logical path through a workflow instance. The same workflow template definition is instantiated multiple times and can be traversed in different ways if the flow contains decision nodes.

The guidelines for working with variables that apply to noncollaborative workflows also apply to collaborative workflows. (They are fully explained in *Using the WebLogic Integration Studio*.) When creating workflows, keep the following guidelines in mind:

- You must define workflows that require variables during run time in the Studio as workflow variables. During workflow execution, you can access a workflow variable in the following ways:
  - Within the workflow instance, such as in a decision node
  - Within an Enterprise Java Bean (EJB) or Java class that starts a workflow or manipulates business messages sent or received by the workflow
- As with structured programming languages, it is always good practice to define your variables *before* you create your workflows. With the Studio, defining variables before creating workflows provides two distinct benefits:
  - It encourages you to plan your workflows more thoroughly before you create them, leading to a more efficient and deliberate design cycle.
  - When you identify variables in various dialog boxes and tabs in the Studio, the variables appear in selection boxes. This is not only convenient for choosing variables; it also reduces the risk of error in associating a variable type and name with a given node in a workflow.
- Java applications that start workflows programmatically can access workflow variables specific to the workflow at any time by referencing the workflow instance object that you obtain when you start the workflow.

Note: Applications that start workflows programmatically must use the Message Manipulator API (package `com.bea.b2b.wlpi`). For more information, see "Developing Applications that Start Conversation Initiator Workflows" on page 3-19.

Associations Between Workflow Variables and Java Data Types

If you want to access a workflow variable within a business operation, you need to know how workflow variable types correspond to Java data types. The following table shows how they are related.

Table 2-3 Workflow Variables and Java Data Types

Workflow Variable Type	Java Data Type
`String`	`java.lang.String`
`Integer`	`java.lang.Long`
`Double`	`java.lang.Double`
`Date`	`java.util.Date`
`Boolean`	`java.lang.Boolean`
`Complex Object`	`java.lang.Object` (must implement `Serializable`)
`XML`	`org.w3c.dom.Node`

Rules for Defining Workflow Variables

The rules for defining Workflow variables for collaborative workflows are the same as those for noncollaborative workflows:

- You can access workflow variables programmatically in the following situations:
  - After the instance is created but before it is started, as described in "Step 5: Get Input Variables from the Workflow Instance" on page 4-24.
  - During the execution of a Manipulate Business Message action, as described in "Defining the Manipulate Business Message Action" on page 4-15.
  - After the instance completes, you can access output variables.
- Before an application can set an input variable for a workflow instance, you must first define the variable for that workflow in the Studio. (Note that this application must use the Message Manipulator API (package `com.bea.b2b.wlpi`) in the application to set those variables that are passed to the workflow.)
- For a workflow that has been started by an application, you can use the workflow instance in that application to set or retrieve variables.

- If you define an output variable (a variable that is passed out of the workflow), you can access its value after the workflow has finished.
- Actions for accessing business messages use Java object variables for manipulating, sending, or receiving business messages. However, the objects stored in the variables through these actions belong to an internal class that encapsulates the business message. As a consequence, these variables should be accessed only through message manipulators. Directly accessing these variables will result in undefined behavior. For more information, see "Defining Workflow Variables for Business Messages" on page 4-2.
- When setting variables in the Compose Business Message and Extract Business Message Parts actions, make sure the variables have the following appropriate types:
  - The part of the business message that holds an XML document may be stored in or retrieved by a workflow variable of type `XML`.
  - Any business message part may be stored in or retrieved from a file. A workflow variable of type `String` contains the filename.

  Note that the variables that specify binary attachments to a business message must contain a source file or destination file, as appropriate.

Defining Variables

For complete details about defining workflow variables in the Studio, see "Using Workflow Expressions" in *Using the WebLogic Integration Studio*.

CHAPTER

3 Starting Collaborative Workflows

The following sections explain how to define start actions for collaborative workflows:

- About Starting Collaborative Workflows
- Starting a Conversation Initiator Workflow
- Defining the Start Node for a Conversation Participant Workflow
- Defining the Start Public Workflow Action
- Developing Applications that Start Conversation Initiator Workflows

About Starting Collaborative Workflows

Note: The ebXML business protocol uses a conversation model that is simpler than the one presented here. Therefore, if you are using only the ebXML business protocol we recommend that you consult the ebXML documentation directly instead of using this document. For more information about creating workflows for the ebXML business protocol, see "Using Workflows with ebXML" in *Implementing ebXML for B2B Integration*.

The job of starting collaborative workflows comprises two tasks:

- Defining Collaborative Workflow Start Properties
- Starting the Collaborative Workflow at Run Time

The following two sections describe the design and programming implications of each task.

Defining Collaborative Workflow Start Properties

There are three ways in which you can start a collaborative workflow:

- Programmatically, via a Java application that starts the workflow
- As a subworkflow, via the Start Public Workflow action defined on a node in the parent workflow
- When a business message is received, via the business message start state

You define the start properties for a workflow based on the type of workflow being designed (conversation initiator versus conversation participant) and according to the following rules:

- For a conversation initiator workflow that is started programmatically, you must specify a manual start property in the Start node.
- For a conversation initiator workflow that is started via the Start Public Workflow action, you must specify a called start state in the Start node.
- For a conversation participant workflow, you must define a business message start state in the Start node.

Note: To instantiate any workflow, the workflow template definition must be active and not expired.

Starting the Collaborative Workflow at Run Time

How you start a collaborative workflow at run time typically depends on whether the workflow is a conversation initiator or conversation participant:

- Conversation initiator workflows can be started in either of two ways:
  - A parent, or other local workflow, can start the conversation initiator workflow by using the Start Public Workflow action defined on one of the

parent workflow's nodes. When the node with the Start Public Workflow action is executed, the conversation initiator workflow is started.

- A user-written application can start the conversation initiator workflow at run time. WebLogic Integration provides the Message Manipulator API (package `com.bea.b2b.wlpi`) that the application uses to start the workflow and handle exceptions.

- Conversation participant workflows are started when the business message defined to start that workflow is received by the WebLogic Integration process engine.

Starting a Conversation Initiator Workflow

A conversation initiator workflow is started in one of the following ways:

- Programmatically via a Java application
- Via another local workflow

Conversation initiator workflows that are started programmatically must have a manual start property defined in the Start node. For more information about starting a conversation initiator workflow programmatically, see "Developing Applications that Start Conversation Initiator Workflows" on page 3-19.

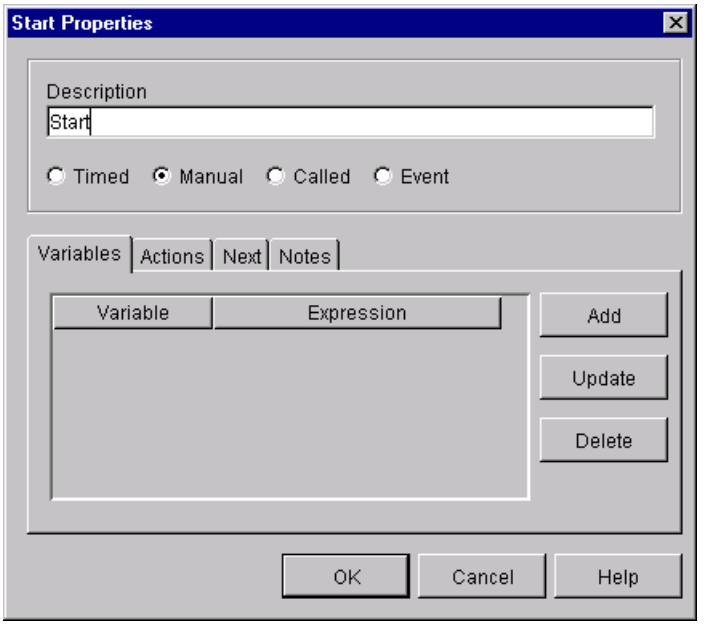
Conversation initiator workflows that are started via another local workflow must have the called start property defined in the Start node. The local workflow—in the design pattern used in the sample applications, this local workflow is called a private workflow—starts the conversation initiator workflow as a subworkflow. For more information about the sample applications, see *Learning to Use BEA WebLogic Integration* and *Running the B2B Integration Samples*.

To define the start property for a conversation initiator workflow:

1. Display or add the start shape, as described in "Defining Workflow Templates" in *Using the WebLogic Integration Studio*.

2. Double-click the start shape to display the Start Properties dialog box.

Figure 3-1 Start Properties Dialog Box: Manual Start



3. Change the text in the Description field to a unique, identifiable name, if desired.
4. If the workflow will be started programmatically, select Manual. If the workflow will be started via another workflow, select Called.

   For more information about these options, see "Defining Workflow Templates" in *Using the WebLogic Integration Studio*.
5. Click OK.

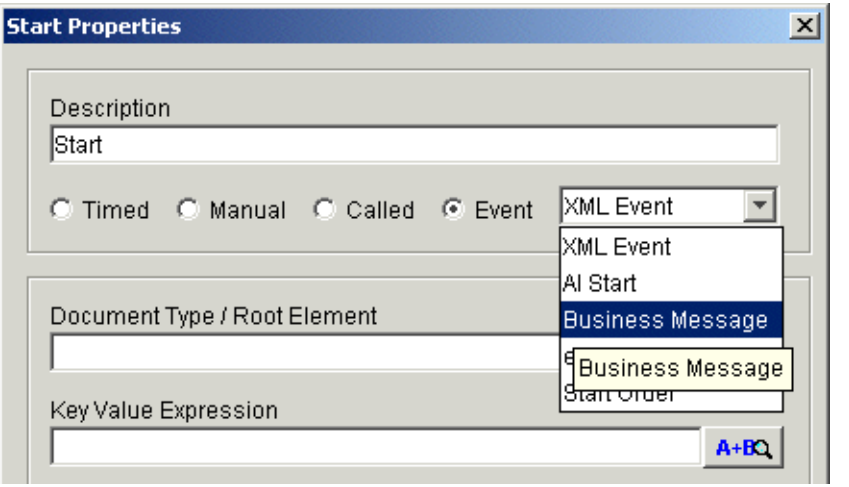
Defining the Start Node for a Conversation Participant Workflow

A conversation participant workflow is started when it receives an initial business message from a trading partner. You must define a Business Message start state for such workflows.

To define the Business Message start state for a conversation participant workflow:

1. Display or add the start shape, as described in "Defining Workflow Templates" in *Using the WebLogic Integration Studio*.
2. Double-click the start shape to display the Start Properties dialog box.
3. In the Start Properties dialog box, select Event, and in the drop-down list that appears to the right, select Business Message.

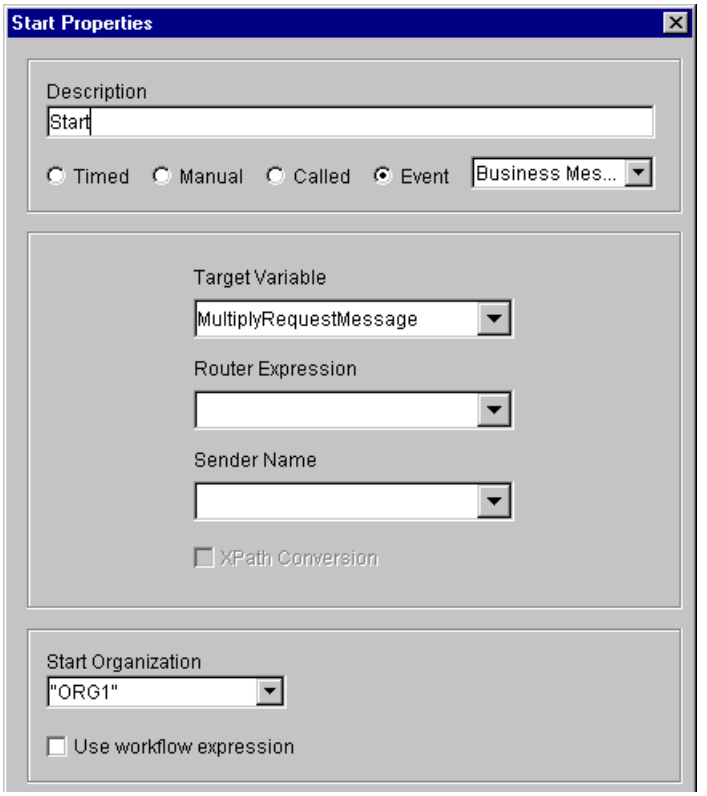
Figure 3-2 Choosing a Business Message Start State



The Start Properties dialog box is displayed again with the additional fields, shown in the following figure. In these fields you add information specific to the collaborative workflow, particularly information related to the business protocol

on which the workflow is based. (The following figure shows the Start Properties dialog box for a workflow based on the XOCP protocol.)

Figure 3-3 Start Properties for a Conversation Participant Workflow



4. Change the text in the Description field to a unique, identifiable name.
5. Provide the information required by the business protocol on which the workflow is based, described in the following sections:
   - For XOCP 1.1-based workflows, see “Starting Conversation Participant Workflows Based on the XOCP 1.1 Protocol” on page 3-7.
   - For RosettaNet 2.0-based workflows, see “Starting Conversation Participant Workflows Based on the RosettaNet 2.0 Protocol” on page 3-8.

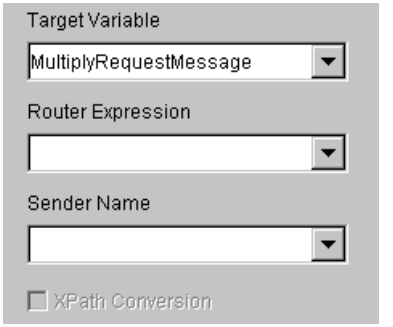
- For RosettaNet 1.1-based workflows, see "Starting Conversation Participant Workflows Based on the RosettaNet 1.1 Protocol" on page 3-8.

6. Specify a start organization, as described in "Defining Workflow Templates" in *Using the WebLogic Integration Studio*.
7. Click OK.

Starting Conversation Participant Workflows Based on the XOCP 1.1 Protocol

In the Start Properties dialog box, enter the following information for workflows based on the XOCP 1.1 protocol.

Figure 3-4 XOCP 1.1 Start Properties



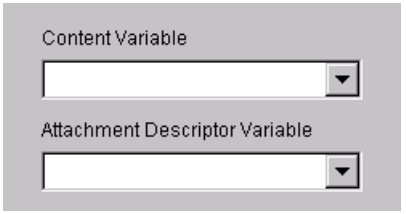
1. The target variable is a workflow variable of type `object` that will contain an instance of an internal type that, in turn, contains the business message.
2. The router expression is a workflow variable of type `string` containing the router expression that was set by the sender of the business message.
3. The sender name is a workflow variable of type `string` that contains the name of the trading partner who sent the business message.

   If you want to specify that the sender name is to be converted to an XPath expression, select the XPath Conversion button.

Starting Conversation Participant Workflows Based on the RosettaNet 2.0 Protocol

In the Start Properties dialog box, enter the following information for workflows based on the RosettaNet 2.0 protocol.

Figure 3-5 RosettaNet 2.0 Start Properties



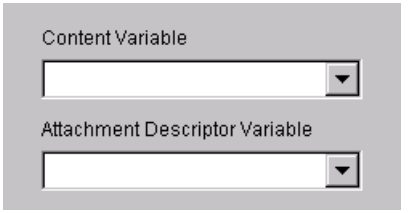
1. The content variable is a workflow variable of type `XML` that contains the business message content.
2. The attachment descriptor variable is an optional workflow variable of type `XML` that contains XML data that describes the attachment files to the business message.

For complete details about creating start properties for workflows based on the RosettaNet 2.0 protocol, see *Implementing RosettaNet for B2B Integration*.

Starting Conversation Participant Workflows Based on the RosettaNet 1.1 Protocol

In the Start Properties dialog box, enter the following information for workflows based on the RosettaNet 1.1 protocol.

Figure 3-6 RosettaNet 1.1 Start Properties



1. The content variable is a workflow variable of type `XML` that contains the business message content.
2. The attachment descriptor variable is an optional workflow variable of type `XML` that contains XML data that describes the attachment files to the business message.

For complete details about creating start properties for workflows based on the RosettaNet 1.1 protocol, see *Implementing RosettaNet for B2B Integration.*

Defining the Start Public Workflow Action

You can use the Start Public Workflow action to start a collaborative workflow from a parent workflow. When you use the Start Public Workflow action, you need to specify the following:

- The values for the conversation definition, role, and a minimum number of parties needed to uniquely identify the collaboration agreement associated with the called workflow instance
- Values of the child workflow input variables
- A variable name to contain the instance ID of the child workflow
- Child output variables in which the parent workflow is interested and the variables in the parent workflow that will contain the child workflow output values

When you start a collaborative workflow via the Start Public Workflow action, that collaborative workflow must include a means to synchronize the collaborative workflow with the parent workflow.

The sections that follow explain how to define the Start Public Workflow action and how to synchronize the called collaborative workflow with the parent workflow.

Defining the Start Public Workflow Action

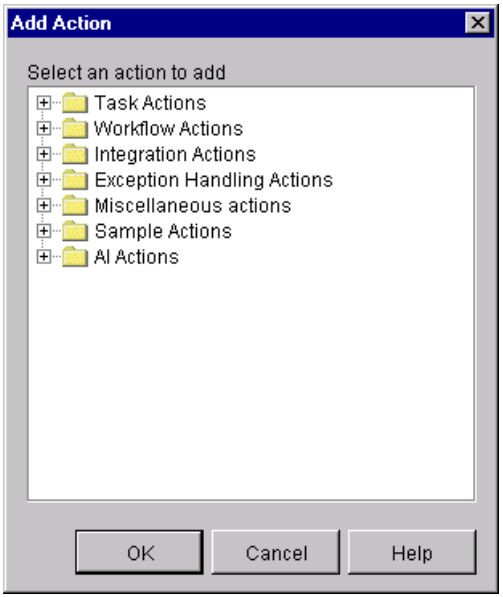
The Start Public Workflow action can be associated with any workflow nodes, but typically is associated with a task node. You must explicitly add the Start Public Workflow action to the workflow template definition.

Adding a Start Public Workflow Action

To define the Start Public Workflow action for a parent workflow in the Studio:

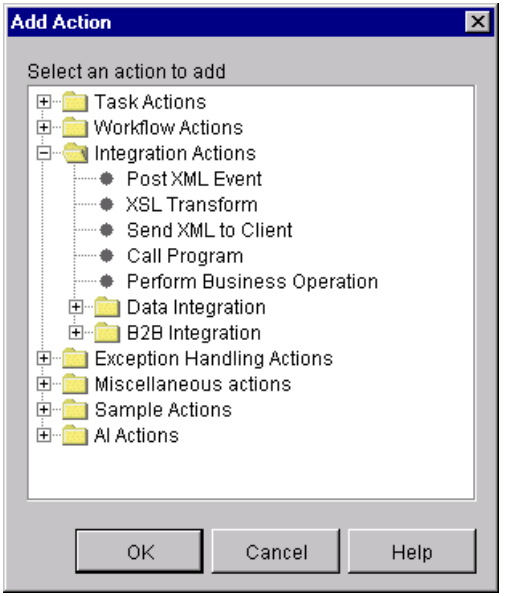
1. Open the workflow in which you want to add the Start Public Workflow action, making sure that the workflow is active and not expired.
2. In any dialog box where you can specify an action (typically in a task node), click Add to display the Add Action dialog box.

Figure 3-7 Add Action Dialog Box



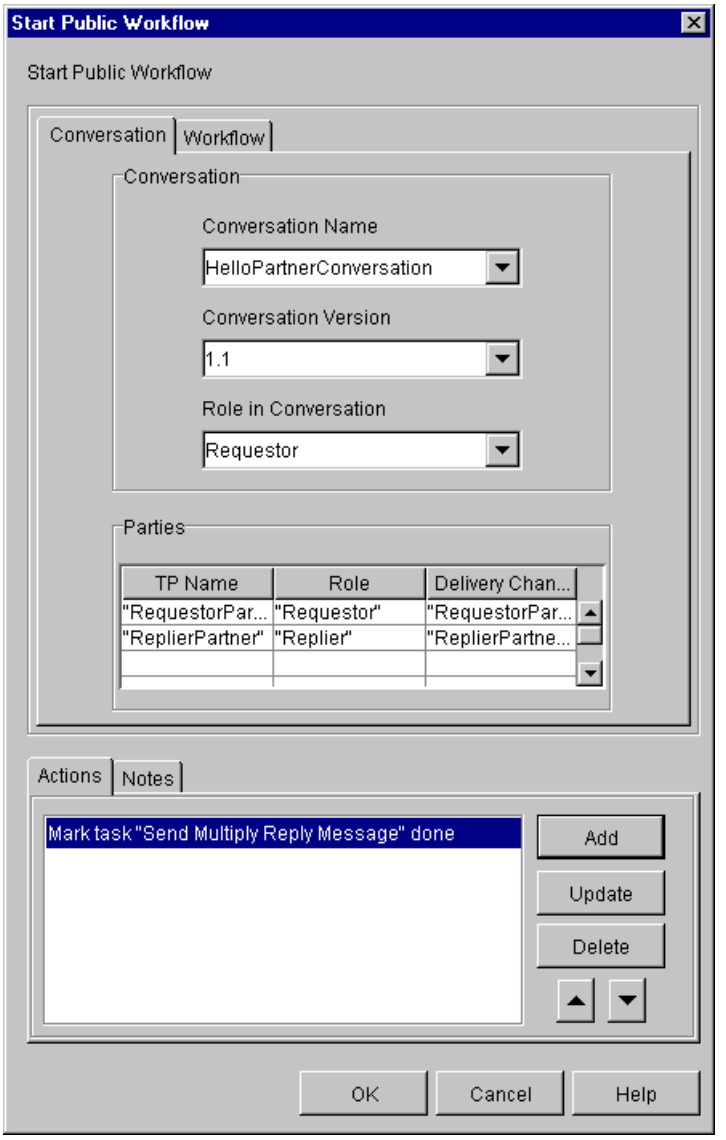
3. Click the Integration Actions folder to expand it.

Figure 3-8 Add Action Dialog Box with Integration Actions



4. Click the B2B Integration folder to expand it.
5. Select Start Public Workflow.
6. Click OK to display the Start Public Workflow dialog box.

Figure 3-9 Start Public Workflow Dialog Box



7. Complete the following sections on the Conversation page of the Start Public Workflow dialog box. The values and variables you specify in this page uniquely identify the collaboration agreement that contains the workflow required for the conversation.

Table 3-1 Sections on the Conversation Page

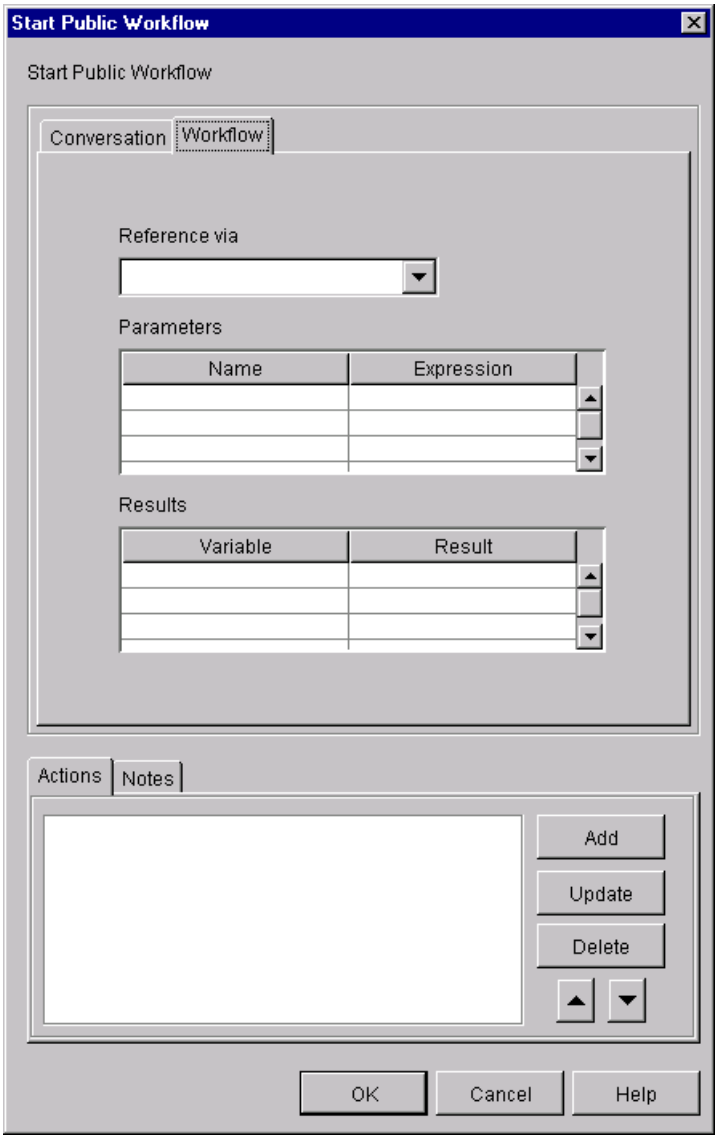
Dialog Box Section	Description
Conversation	Enter the following: ■ Name of the conversation definition in the WebLogic Integration repository ■ Conversation definition version ■ Conversation role represented by the collaborative workflow to be started

Table 3-1 Sections on the Conversation Page (Continued)

Dialog Box Section	Description
Parties	The trading partner name, role, and delivery channel of the parties configured in the collaboration agreement for the associated conversation definition to which the workflow applies. You do not need to specify all the parties, but you must identify enough to uniquely identify the collaboration agreement associated with the workflow. The information you provide for role and delivery channel must match the information configured for the corresponding trading partner in the repository. Specifying the role and delivery channel is optional, depending on whether that informaiton is required to uniquely identify the collaboration agreement. ■ To specify a trading partner name, click on the cell under the TP Name column, and enter the name of trading partner. ■ To specify a role, click on the cell under the Role column and enter the role name. ■ To specify a delivery channel, click on the cell under the Delivery Channel column and enter the delivery channel name. ■ To insert a row into a list of parties, right-click the row where you want to make the insertion and choose Insert. A blank row is created above the row in which you clicked. ■ To remove a row from the list of parties, right-click the row you want to remove and select Delete. You can use the Expression Builder to specify party values in the Start Public Workflow dialog box. For more information, see "Using the Expression Builder to Specify Values in the Start Public Workflow Dialog Box" on page 3-17.
Notes	Optional descriptive text.
Actions	Tab in which you can add workflow actions. In the Actions tab, you typically add the "Mark Task as Done" property to synchronize the public workflow with the parent workflow. For more information, see "Synchronizing the Child Workflow With the Parent Workflow" on page 3-18.

8. Select the Workflow tab. The following page is displayed.

Figure 3-10 Workflow Page of the Start Public Workflow Dialog Box



9. Complete the following fields and sections of the Workflow tab, which identify both the variable that will contain the workflow instance ID of the child workflow and the variables passed between the parent and child workflows.

Table 3-2 Fields and Sections of the Workflow Tab

Field or Section	Description
Reference via	The workflow variable of type `String` that will contain the instance ID of the child workflow. You use this variable as a means to implement a standard communication mechanism between the parent and child workflows.
Parameters	The variable names (in the Name column) and their respective values (in the Expression column) to be passed from the parent workflow to the child workflow. The variable names that you specify here must also be specified as input variables to the respective child workflow. As explained in "About Using Workflow Variables" on page 2-15, you should define these variables in both the parent and child workflows before you select them in this dialog box. This ensures that the variables chosen have the correct name and type. You can use the Studio Expression Builder to create an expression for each variable's value. For more information about using the Expression Builder, see "Using the Expression Builder to Specify Values in the Start Public Workflow Dialog Box" on page 3-17.
Results	Output variable names and their respective values to be passed from the child workflow back to the parent after the child workflow has completed its execution. As with input variables, these output variables should be defined in both the parent and child workflows before you select them in this dialog box.
Notes	Optional descriptive text.

10. Click OK to save your changes.

Using the Expression Builder to Specify Values in the Start Public Workflow Dialog Box

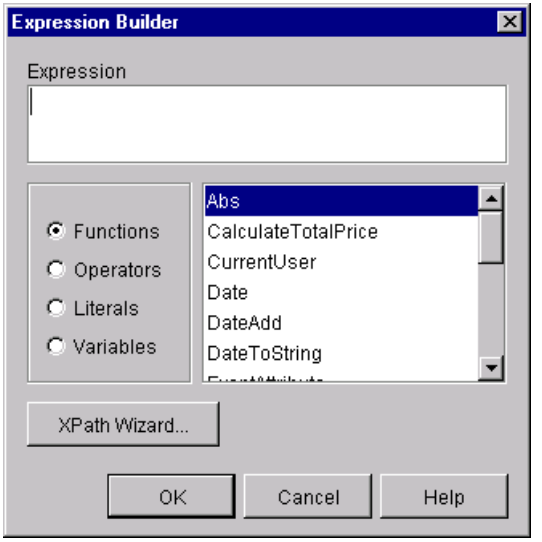
You can use the Studio Expression Builder to specify the following values in the Start Public Workflow dialog box:

- Trading partner names

- Roles
- Delivery channels
- Workflow parameter expressions

To display the Expression Builder, right-click in the cell in which you specify one of the preceding values and choose Expression. The Expression Builder is displayed, as shown in the following figure.

Figure 3-11 Studio Expression Builder



For information about how to use the Expression Builder, see "Using Workflow Expressions" in *Using the WebLogic Integration Studio*.

Synchronizing the Child Workflow With the Parent Workflow

As mentioned in "Defining the Start Public Workflow Action" on page 3-9, when you start a collaborative workflow via the Start Public Workflow action, you must have a means of synchronizing that collaborative workflow with the parent workflow. You can do this in either of two ways:

- Use an application that implements its own synchronization with the workflows.
- In the parent workflow where you define the Start Public Workflow action, mark the task as done in the Start Public Workflow actions (not in the Task actions). This prevents the parent workflow from passing execution to the next node in its workflow until the called public workflow has finished processing.

Developing Applications that Start Conversation Initiator Workflows

You can start a conversation initiator workflow at run time via a Java application, known as a workflow application. The following sections describe how to create a workflow application to start a conversation initiator workflow programmatically:

- Message Manipulator API
- Programming Steps for Accessing Conversation Initiator Workflows

To start a conversation initiator workflow programmatically, the start node for the workflow template must have a Manual start property, as described in "Starting a Conversation Initiator Workflow" on page 3-3.

Note: In WebLogic Integration release 2.0 and later, attempting to start a collaborative workflow via the Worklist results in an error.

Message Manipulator API

Workflow applications can use the `com.bea.b2b.wlpi` package to start workflows. This package provides the following interface and classes.

Table 3-3 Components of the com.bea.b2b.wlpi Package

Object	Description
`MessageManipulator` interface	Implemented by all classes that are used in a Manipulate Message action inside a workflow task
`WorkflowInstance` class	Represents a running workflow instance
`WLPIException` class	Thrown by a public method of the workflows API if an error occurs with processing

For details about this package, see the *BEA WebLogic Integration Javadoc*.

Programming Steps for Accessing Conversation Initiator Workflows

To access a conversation initiator workflow, a workflow application completes the following steps:

- Step 1: Import the Necessary Packages
- Step 2: Create a Workflow Instance Object for a Specific Workflow Template
- Step 3: Initialize Input Variables
- Step 4: Start a Workflow Instance
- Step 5: Wait for the Workflow Instance to Complete
- Step 6: Handle Results in Output Variables

The workflow application also needs to handle exceptions thrown by each of the method invocations on the Message Manipulator API, as explained in "Handling Exceptions" on page 3-24.

Note: Before creating an application that starts a workflow, verify the following:

- The template definition for the workflow has been created.
- The workflow has a manual start state, is active, and has not expired.

Step 1: Import the Necessary Packages

To access a workflow, a workflow application begins by importing the necessary package(s). At a minimum, the application must import the `com.bea.b2b.wlpi` package, as shown in the following example.

Listing 3-1 Importing the com.bea.b2b.wlpi Package

```
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.*;
```

Step 2: Create a Workflow Instance Object for a Specific Workflow Template

The workflow application next needs to create a workflow instance object of a specific workflow template definition, as shown in the following example.

Listing 3-2 Creating a Workflow Instance Object

```
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);
```

Step 3: Initialize Input Variables

If input variables are defined for a conversation initiator workflow, you must initialize and assign values to them in the application before starting the workflow instance. These input variables must first be declared in the template definition in the Studio, as described in "About Using Workflow Variables" on page 2-15.

A workflow application sets instance variables by invoking the `setVariable` method on the workflow instance and passing it the name and value of the variable. The `setVariable` method requires the following parameters.

Table 3-4 Parameters for the setVariable Method

Parameter	Description
`name`	Name of the variable to be set.
`value`	Value for the variable. The value must be represented as a Java object, as described in "Associations Between Workflow Variables and Java Data Types" on page 2-16.

The following example shows how to use the `setVariable` method to specify values for two variables in the workflow to be started.

Listing 3-3 Setting the Value of Input Variables

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

Step 4: Start a Workflow Instance

After creating a workflow instance and initializing input variables, a workflow application starts the workflow instance by invoking the `start` method on the instance object, as shown in the following example.

Listing 3-4 Start a Workflow Instance

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

Step 5: Wait for the Workflow Instance to Complete

Once a workflow instance has been started, a workflow application can wait for its completion by calling the `waitForCompletion` method on the workflow instance. The operation blocks until the workflow instance has completed.

Listing 3-5 Waiting for Completion of the Workflow Instance

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

While waiting for the workflow instance to complete, a workflow application can determine the completion state of the workflow instance by invoking the `isCompleted` method on the workflow instance. This method returns a Boolean `true` if the execution of the workflow is complete, or `false` if it is not.

Step 6: Handle Results in Output Variables

After a workflow instance has completed, a workflow application can handle the results of the workflow instance by retrieving the information stored in output variables. These output variables must first be declared in the template definition in the Studio.

A workflow application retrieves the value of an instance variable by calling the `getVariable` method on the workflow instance and passing it the name of the variable to be retrieved, as shown in the following example listing.

Listing 3-6 Retrieving the Results in Output Variables

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

The `getVariable` method returns a Java object that should be cast to the appropriate Java data type. For information about the Java types to which you must cast the return values, depending on the corresponding workflow variable types, see *Using the WebLogic Integration Studio*.

Handling Exceptions

If an error occurs while a workflow application is running, a `com.bea.b2b.wlpi.WLPIException` is thrown. With each method invocation on the workflows API, workflow applications can catch this exception and process it as appropriate, as shown in the following listing.

Listing 3-7 Handling WLPIExceptions in Workflow Applications

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

4 Working with Business Messages

The following sections explain how to use the B2B integration plug-in to construct and manipulate the contents of business messages that are exchanged among business partners:

- About Working with Business Messages
- Defining Workflow Variables for Business Messages
- Simple Message Manipulation
- Complex Message Manipulation

About Working with Business Messages

A business message is the basic unit of communication exchanged by trading partners in a conversation. A business message consists of the following:

- One or more *business documents*, which represent the XML-based payload part of a business message. The payload is the business content of a business message.
- Optionally, one or more *attachments*, which represent additional XML or nonXML payload parts of a business message.

The B2B integration plug-in provides two methods to either create or extract the contents from business messages:

- Simple message manipulation is supported by the Compose Business Message and Extract Business Message Parts actions. These two actions provide a means to manipulate the contents of business messages nonprogrammatically. In the current release of WebLogic Integration, the Compose Business Message and Extract Business Message Parts actions are supported for XOCP messages only.
- Complex message manipulation is supported by the Manipulate Business Message action. This action starts a user-written application that implements the `MessageManipulator` interface from the Message Manipulator API (package `com.bea.b2b.wlpi`). This interface enables you to create and manipulate complex messages. In the current release of WebLogic Integration, the Manipulate Business Message action is supported for XOCP messages only.

Defining Workflow Variables for Business Messages

At run time, a business message is stored in a workflow variable (of type `Java Object`) when it is ready to be sent or when it has been received. When a business message is ready to be sent, the application code associated with the action that composes the business message constructs the business message and returns it in this variable to the workflow instance. When a business message has been received, the workflow node associated with any of three actions (Compose Business Message, Extract Business Message Parts, or Manipulate Business Message) obtains this variable from the workflow instance and uses it to process the incoming business message.

Note: In WebLogic Integration, XOCP business messages are *not* stored in workflow variables of type `XML Document`.

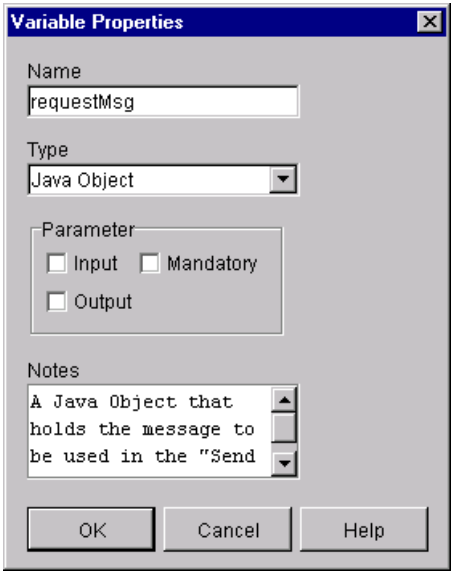
In the Studio, you must define the `Java Object` variables used to store business messages *before* you define any actions that refer to them, as described in "About Using Workflow Variables" on page 2-15.

For each workflow template definition, you can define a separate variable for each business message that the workflow sends or receives, or you can define a single object variable to contain all your message exchanges.

To define a variable for a business document in the Studio:

1. In the folder tree, right-click Variables under the appropriate workflow template definition. Select New Variable to display the Variable Properties dialog box.

Figure 4-1 Variable Properties Dialog Box



2. Specify a unique name for this variable.
3. Select the `Java Object` variable type.
4. Click OK.

Simple Message Manipulation

The B2B integration plug-in provides two actions that you can use for simple message manipulation:

- Compose Business Message—for creating a business message that you can subsequently send via the Send Business Message Action
- Extract Business Message Parts—for extracting the components of a business message for further processing, which you typically do after your workflow has received a business message

The primary ease-of-use feature provided by these two actions is that they provide a means to manipulate business messages entirely nonprogrammatically. The sections that follow explain how to use these two workflow actions.

Creating Business Messages

You can define the Compose Business Message action on any node in a collaborative workflow—start, task, decision, event, or done. You must explicitly add the Compose Business Message action to the workflow template definition. At run time, this action adds one or more XML documents and one or more attachments to a business message envelope, specified as variables in the Compose Business Message dialog box, which gives you an easy means to create a message that you can subsequently send via the Send Business Message action.

For overview information about the composition of a business message, refer to either of the following:

- “Conversations and Business Messages” on page 1-7
- “Supporting Business Protocols” in “Overview” in *Introducing B2B Integration*.

Defining the Compose Business Message Action

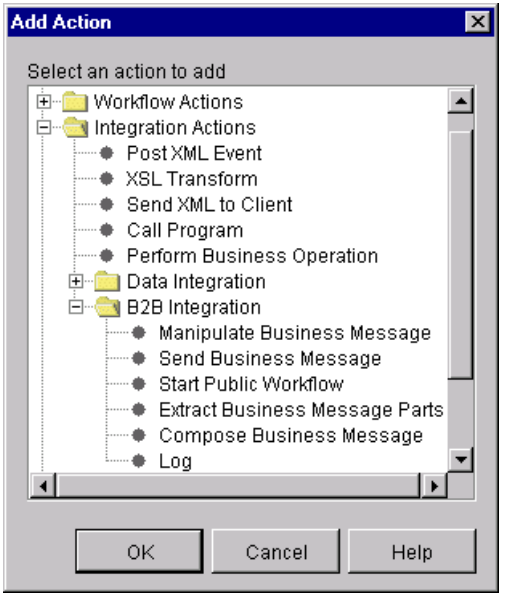
To define the Compose Business Message action for a workflow in the Studio:

1. In any dialog box where you can specify an action (such as the Task, Decision, Event, or Start Properties dialog box), click Add to display the Add Action dialog box.

Figure 4-2 Add Action Dialog Box

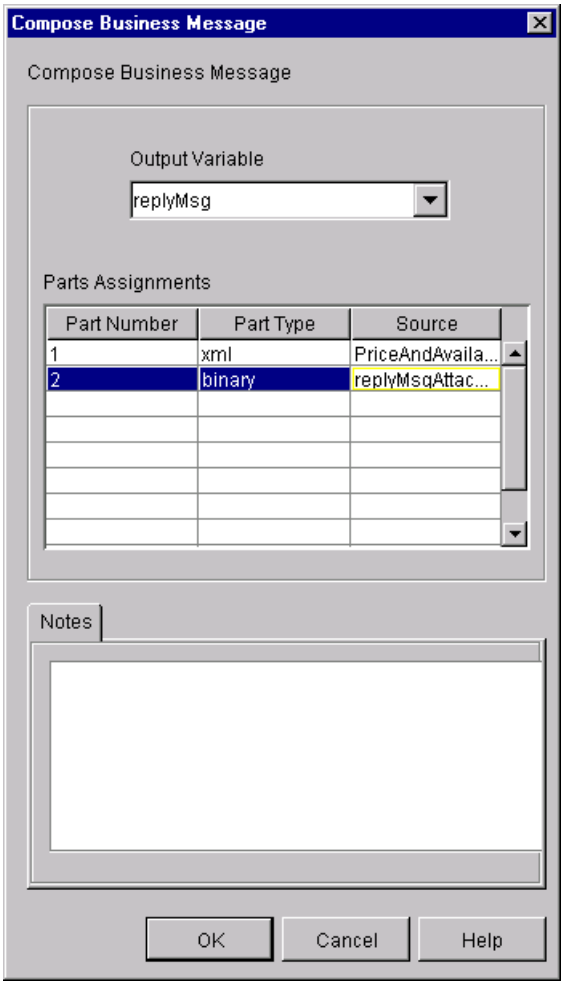
2. Click the Integration Actions folder to expand it.

Figure 4-3 Add Action Dialog Box with Integration Actions



3. Click the B2B Integration folder to expand it.
4. Select Compose Business Message.
5. Click OK to display the Compose Business Message dialog box.

Figure 4-4 Compose Business Message Dialog Box



6. Complete the following fields in the Compose Business Message dialog box.

Table 4-1 Fields in the Compose Business Message Dialog Box

Field	Description
Output Variable	The workflow variable of type `Java object` that holds the business message being composed.
Part Number	The ID for each component in the business message being composed. Each entry in the Parts Assignment table must have a unique ID, and it must be a positive integer. To assign an ID: 1. Click in the appropriate cell under the Part Number column. 2. Enter the part ID.
Part Type	Identifies the type of message part: XML business document, or attachment. ■ XML documents must be defined with the workflow variable of type `XML`. ■ Attachments must be defined with `binary` type. To choose a part type: 1. Click in the appropriate cell under the Part column. 2. Select the part type from the drop-down list that is displayed.
Source	Identifies the workflow variable containing the filename of the associated part type. ■ If the part type is `XML`, the source should be an `XML` variable containing the filename of the XML business document. ■ If the part type is `binary`, the source should be a `String` variable containing the filename of the attachment. To choose a source file: 1. Click in the appropriate cell under the Source column. 2. Click the down-arrow that is displayed along the right edge of the cell. 3. Select the variable name containing the name of the file that holds the appropriate message part.
Text	Optional descriptive text.

7. To insert a message part between two parts that are already specified:
   a. Right-click the row where you want to make the insertion.
   b. Choose Insert. A blank row is created above the row on which you clicked.
8. To delete an entry in the Parts Assignments table of the Compose Business Message dialog box:
   a. Click the row specifying the message part you want to delete.
   b. Right-click and choose Delete.
9. When you have finished specifying the Compose Business Message dialog box, click OK.

Extracting Information from Business Messages

You can define the Extract Business Message Parts action on any node in a collaborative workflow—start, task, decision, event, or done. You must explicitly add the Extract Business Message Parts action to the workflow template definition. At run time, this action extracts the XML document(s) and attachment(s) to a business message envelope, specified as variables in the Extract Business Message Parts dialog box, which gives you an easy means to obtain the contents of a business message that has been received.

For overview information about the composition of a business message, refer to either of the following:

- “Conversations and Business Messages” on page 1-7
- “Supporting Business Protocols” in “Overview” in *Introducing B2B Integration*.

Defining the Extract Business Message Parts Action

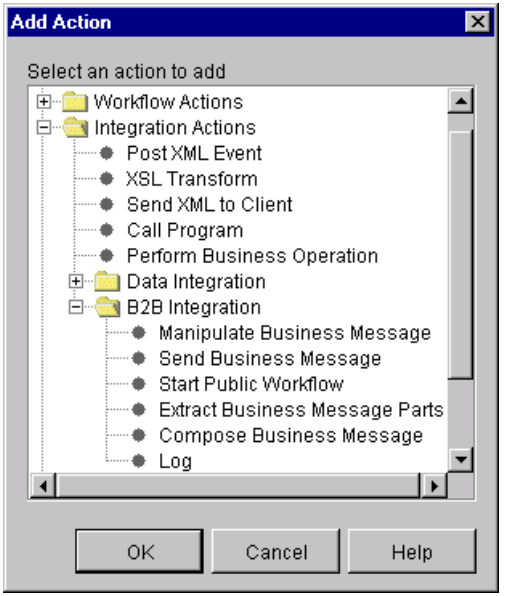
To define the Extract Business Message Parts action for a workflow in the Studio:

1. In any dialog box where you can specify an action (such as the Task, Decision, Event, or Start Properties dialog box), click Add to display the Add Action dialog box.

Figure 4-5 Add Action Dialog Box

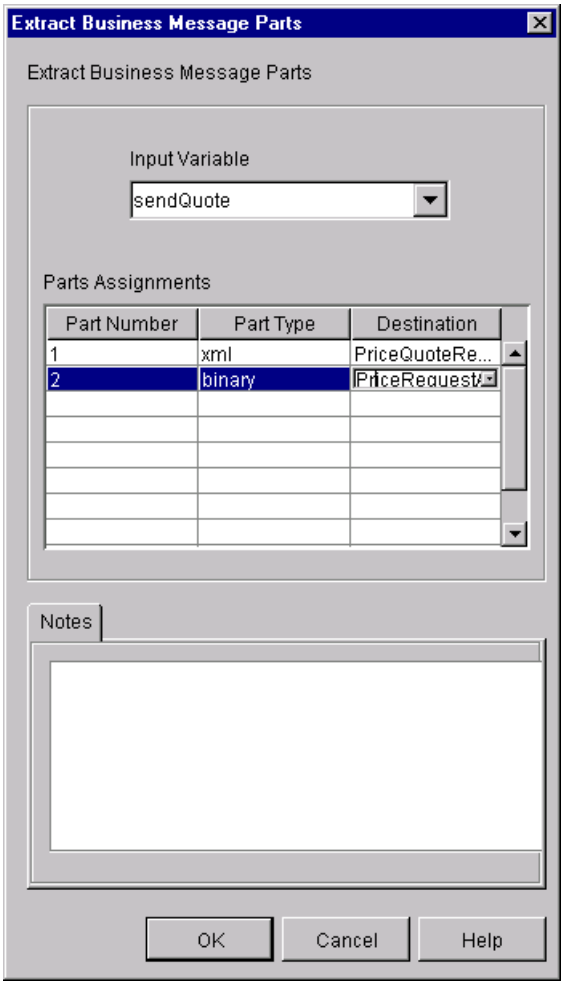
2. Click the Integration Actions folder to expand it.

Figure 4-6 Add Action Dialog Box with Integration Actions



3. Click the B2B Integration folder to expand it.
4. Select Extract Business Message Parts.
5. Click OK to display the Extract Business Message Parts dialog box.

Figure 4-7 Extract Business Message Parts Dialog Box



6. Complete the following fields in the Extract Business Message Parts dialog box.

Table 4-2 Fields in the Extract Business Message Parts Dialog Box

Field	Description
Input Variable	The workflow variable of type `Java object` that holds the business message from which the parts are being extracted.
Part Number	The ID for each component to be extracted from the business message. Note the following: ■ Each entry you specify in the Parts Assignment table must have a unique ID, it must be a positive integer, and it must correspond to the ID of the message part you want to extract from the business message. ■ You do not need to extract every message part in the business message, only the parts in which you are interested. To assign an ID: 1. Click in the appropriate cell under the Part Number column. 2. Enter the part ID.
Part Type	Identifies the type of message part being extracted: XML business document or attachment. ■ XML documents must be defined with the workflow variable of type `XML`. ■ Attachments must be defined with the `binary` type. To choose a part type: 1. Click in the appropriate cell under the Part column. 2. Select the part type from the drop-down list that is displayed.

Table 4-2 Fields in the Extract Business Message Parts Dialog Box (Continued)

Field	Description
Destination	Identifies the workflow variable containing the filename of the associated part type. ■ If the part type is `XML`, the destination should be an `XML` variable containing the filename of the XML business document. ■ If the part type is `binary`, the destination should be a `String` variable containing the filename of the attachment. To choose a destination file: 1. Click in the appropriate cell under the Destination column. 2. Click the down-arrow that is displayed along the right edge of the cell. 3. Select the variable name containing the name of the file that will hold the extracted message part.
Text	Optional descriptive text.

7. To insert a message part between two parts that are already specified:
   a. Right-click the row where you want to make your insertion.
   b. Choose Insert. A blank row is created above the one on which you clicked.
8. To delete an entry in the Parts Assignments table of the Extract Business Message Parts dialog box:
   a. Click in the ID cell of the message part you want to delete.
   b. Right-click and choose Delete.
9. Click OK.

Complex Message Manipulation

You must perform the following tasks before you can create or extract the content of business messages via a Java application that implements the `MessageManipulator` interface:

- Create the workflow variables that will be used to hold the contents of the business messages that are created or manipulated, as explained in "Defining Workflow Variables for Business Messages" on page 4-2.
- Define the Manipulate Business Message action on the appropriate node in the workflow, as explained in "Defining the Manipulate Business Message Action" on page 4-15
- Write the Java application to process the business message by implementing the `com.bea.b2b.wlpi.MessageManipulator` interface and using the `manipulate` method on that object, as explained in "Writing the Application to Manipulate Business Messages" on page 4-20.

The sections that follow explain how to define the Manipulate Business Message action and to write the Java application.

Defining the Manipulate Business Message Action

At run time, the Manipulate Business Message action is invoked to manipulate a business message. If the workflow is sending a business message (such as a request), the Manipulate Business Message action runs the associated application code to create the business message and save it in an output variable that is sent subsequently in a Send Business Message action. If the workflow is receiving a business message (such as a reply), the Manipulate Business Message action captures the incoming business message in an input variable and passes it on to the associated application code for processing.

The Manipulate Business Message action can be associated with any of the following nodes: task, decision, event, and start. You must explicitly add the Manipulate Business Message action to the workflow template definition.

Adding a Manipulate Business Message Action

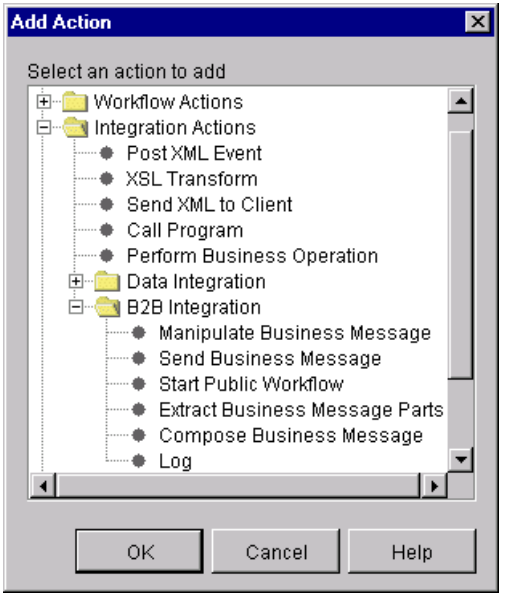
To define the Manipulate Business Message action for a workflow in the Studio:

1. In any dialog box where you can specify an action (such as the Task, Decision, Event, or Start Properties dialog box), click Add to display the Add Action dialog box.

Figure 4-8 Add Action Dialog Box

2. Click the Integration Actions folder to expand it.

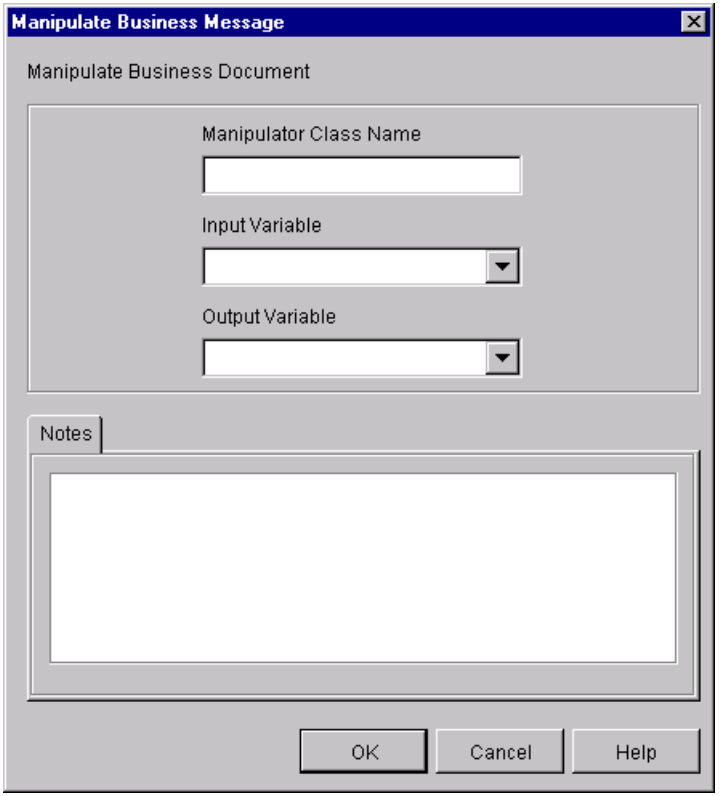
Figure 4-9 Add Action Dialog Box with Integration Actions



3. Click the B2B Integration folder to expand it.

4. Select Manipulate Business Message.
5. Click OK to display the Manipulate Business Message dialog box.

Figure 4-10 Manipulate Business Message Dialog Box



6. Select values for the following fields in the Manipulate Business Message dialog box.

Table 4-3 Fields in the Manipulate Business Message Dialog Box

Field	Description
Manipulator Class Name	Name of a Java class that implements the `com.bea.b2b.wlpi.MessageManipulator` interface. For more information, see "Writing the Application to Manipulate Business Messages" on page 4-20. (Required)
Input Variable	Name of a workflow variable that contains an existing business message, such as a message that has been received through a Receive Business Message action. The contents of this variable are passed, as the `in` parameter, to the `manipulate` operation in the specified Java class that implements the `com.bea.b2b.wlpi.MessageManipulator` interface. If no variable name is specified, the value of the `in` parameter is `null`. The specified variable must correspond to an existing workflow variable of type Java Object. For more information, see "About Using Workflow Variables" on page 2-15.
Output Variable	Name of a workflow variable that contains the business message returned by the `manipulate` operation in the specified Java class that implements the `com.bea.b2b.wlpi.MessageManipulator` interface. The specified variable must correspond to an existing workflow variable of type Java Object. For more information, see "About Using Workflow Variables" on page 2-15. If no variable name is specified, then the return value of the `manipulate` operation is ignored.
Notes	Optional descriptive text.

When specifying variables for the input or output fields, follow these guidelines:

- If the action receives a business message that you want to manipulate, then you must specify a value for the input variable field.
- If the action sends a business message, then you must specify a value for the output variable field.

- If you want to construct a business message to send, you can use a message manipulator with an output field that specifies the business message object.

7. Click OK to save your changes.

Example of a Manipulate Business Message Action

For example, suppose you specify the following settings in the Manipulate Business Message dialog box.

Table 4-4 Sample Settings in the Manipulate Business Message Dialog Box

Field	Description
Manipulator Class Name	`examples.wlpiverifier.ProcessRequest`
Input Variable	`requestMsg`
Output Variable	`replyMsg`

At run time, when the WebLogic Integration process engine executes the action with the specified settings, the following events occur:

1. An object of class `examples.wlpiverifier.ProcessRequest` is created using reflection and the default constructor.
2. The value of the `in` parameter (`requestMsg`) is retrieved.
3. The `manipulate` operation is invoked on the object.
4. The return value of the `manipulate` operation is stored in the workflow output variable (`replyMsg`).

Writing the Application to Manipulate Business Messages

You write a Java application that uses workflow variables and Java code to manipulate business messages that are exchanged between trading partners. The Manipulate Business Message action invokes this application to create a business message to either

send or process a business message that has been received. This Java application implements the `com.bea.b2b.wlpi.MessageManipulator` interface. Such an application is referred to generically as a message manipulator.

For more information about defining the message manipulator class and input and output variables for the Manipulate Business Message action, see "Defining the Manipulate Business Message Action" on page 4-15. For more information about the `com.bea.b2b.wlpi.MessageManipulator` interface, see the *BEA WebLogic Integration Javadoc*.

Message Manipulator Features

Message manipulators can implement the following operations for processing business messages:

- Create business messages before sending them. (See "Steps for Writing the Application that Implements the MessageManipulator Interface to Create a Business Message" on page 4-23.) A workflow *must* send messages to participate in conversations.
  - At run time, the Manipulate Business Message action is invoked. The Manipulate Business Message action invokes the Java application to create a business message, based on the contents of other workflow variables, and returns the business message for storage in a variable. For more information, see "Defining the Manipulate Business Message Action" on page 4-15.
  - The Send Business Message action retrieves this variable and sends the business message. For more information, see "Defining the Workflow to Send Business Messages" on page 5-1.
- Process business messages after receiving them. (See "Steps for Writing the Application that Implements the MessageManipulator Interface to Process the Contents of a Received Business Message" on page 4-26.) After a business message has been received, an invoked business message manipulator extracts the contents of the message and stores any required message parts in workflow variables for use by other actions.
- Replace specific components in a received business message and send the message.
- Perform custom message extraction operations.

MessageManipulator Interface

To process business messages that are exchanged between roles in a conversation, workflow applications use Java classes that implement the `com.bea.b2b.wlpi.MessageManipulator` interface. This interface contains a single operation, `manipulate`, with the following signature:

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

When calling the `manipulate` operation, a workflow specifies the following parameters.

Table 4-5 Parameters in the Manipulate Operation

Parameter	Description
`instance`	Current workflow instance, which can be used to get or set variables. For more information, see "Defining the Manipulate Business Message Action" on page 4-15.
`in`	XOCP message stored in the workflow variable specified as an input variable in the associated Manipulate Business Message action. If no input variable is specified in the Manipulate Business Message action or if the variable is empty, then `null` is passed.

The `manipulate` operation returns an XOCP message generated by the message manipulator. At run time, this XOCP message is stored in the output variable specified in the associated Manipulate Business Message action. If this output variable is not specified, then the return value is ignored.

Public Default Constructor

Classes that implement the message manipulator interface *must* have a public default constructor (a constructor without arguments). The process engine uses Java reflection to create objects of that class and therefore invokes the default constructor.

Steps for Writing the Application that Implements the MessageManipulator Interface to Create a Business Message

The `ChannelMasterMessageFactoryII` class shown in this section is an example of a message manipulator that constructs a business message. It is called by the Manipulate Business Message action that occurs in the workflow. It returns a reply message (`xocpmsg` variable) that is passed back to the workflow as the business message to send.

Step 1: Import the Necessary Packages

The following listing shows the packages that the `ChannelMasterMessageFactoryII` class imports, including the XOCP messaging objects that are used to create the XOCP message.

Listing 4-1 Importing the Necessary Packages

```
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.*;
```

Step 2: Implement the MessageManipulator Interface

The following listing shows the `ChannelMasterMessageFactoryII` class declaration that implements the `MessageManipulator` interface.

Listing 4-2 Implementing the MessageManipulator Interface

```
public class ChannelMasterMessageFactoryII implements MessageManipulator
```

Step 3: Include a Default Constructor

The following listing shows the default constructor used for the `ChannelMasterMessageFactoryII` class.

Listing 4-3 Implementing a Default Constructor

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

Step 4: Include an Invocation to the Manipulate Method

The code in the following listing invokes the `manipulate` method, which retrieves the current workflow instance object, as well as the incoming business message.

Listing 4-4 Invoking the manipulate Method

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

Step 5: Get Input Variables from the Workflow Instance

The code in the following listing uses the `getVariable` method to obtain values from the current workflow instance to be used for creating the business message.

Listing 4-5 Getting the Input Variables

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

Step 6: Create the Business Message

The code in the following listing does the following:

1. Creates the DOM object that represents the XML data used for composing the XML document.
2. Creates the XML document object using the DTD named `temp-xml-transporter.dtd`.
3. Creates each XML element for the XML document.
4. Creates an XOCP message, adding the XML document to it.

Listing 4-6 Creating the XML Document

```
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));
```

Step 7: Return the Request Message

The code in the following listing returns the request message in the variable `xocpmsg` (of type `XOCPMessage`). The return value is then assigned to an output variable (of type Java Object) in the workflow in preparation for sending the business message.

Listing 4-7 Returning the Request Message

```
return xocpmsg;
```

Steps for Writing the Application that Implements the MessageManipulator Interface to Process the Contents of a Received Business Message

The `ChannelMasterMessageExtractorII` class described in this section is an example of a message manipulator that receives and processes a business message. It is called by the Manipulate Business Message action associated with the Start event (defined as a Business Message start event) that is triggered when the initial business message is received from the conversation initiator workflow. It returns a reply message (`replyMsg` variable) that is passed back to the workflow as the business message to send.

Step 1: Import the Necessary Packages

The code in the following listing shows the packages that the `ChannelMasterMessageExtractorII` class imports, including the XOCP messaging objects that are used to create the XOCP business message.

Listing 4-8 Importing the Necessary Packages

```
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.*;
```

Step 2: Implement the MessageManipulator Interface

The following listing shows the `ChannelMasterMessageExtractorII` class declaration that implements the `MessageManipulator` interface.

Listing 4-9 Implementing the MessageManipulator Interface

```
public class ChannelMasterMessageExtractorII implements MessageManipulator
```

Step 3: Include a Default Constructor

The following listing shows the default constructor used for the `ChannelMasterMessageExtractorII` class.

Listing 4-10 Implementing a Default Constructor

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

Step 4: Invoke the Manipulate Method

The code in the following listing invokes the `manipulate` method, which retrieves the current workflow instance object, as well as the incoming business message (request).

Listing 4-11 Invoking the manipulate Method

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

Step 5: Process the Request Message

The code in the following listing does the following:

1. Extracts the payload components from the business message.
2. For each payload part, extracts a DOM document.
3. Uses the DOM API on the DOM object to extract the XML content.

Listing 4-12 Processing the Request Message

```
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 Sending and Receiving Business Messages

The following sections describe how to send and receive business messages in a collaborative workflow:

- Defining the Workflow to Send Business Messages
- Defining the Workflow to Receive Business Messages

Defining the Workflow to Send Business Messages

After you create a business message in the WebLogic Integration Studio by using the Compose Business Message or Manipulate Business Message action, you send the business message using the Send Business Message action. This section explains how to define a Send Business Message action in a collaborative workflow so it sends a business message to a trading partner.

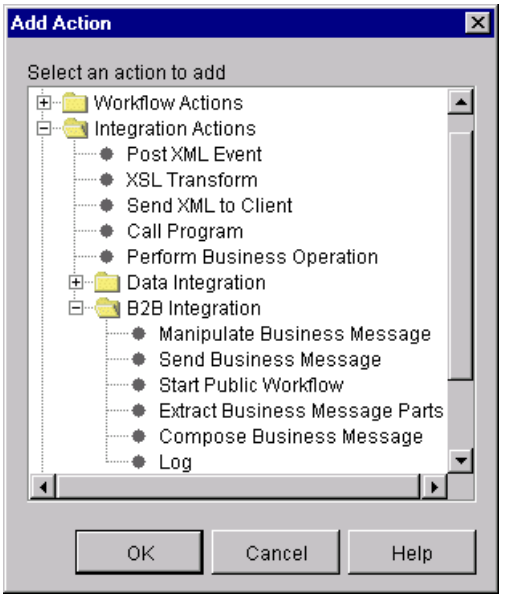
To define a Send Business Message action:

1. In any dialog box where you can specify an action (such as the Task, Decision, Event, or Start Properties dialog box), click Add to display the Add Action dialog box.

Figure 5-1 Add Action Dialog Box

2. Click the Integration Actions folder to expand it.

Figure 5-2 Add Action Dialog Box with Integration Actions



3. Click the B2B Integration folder to expand it.

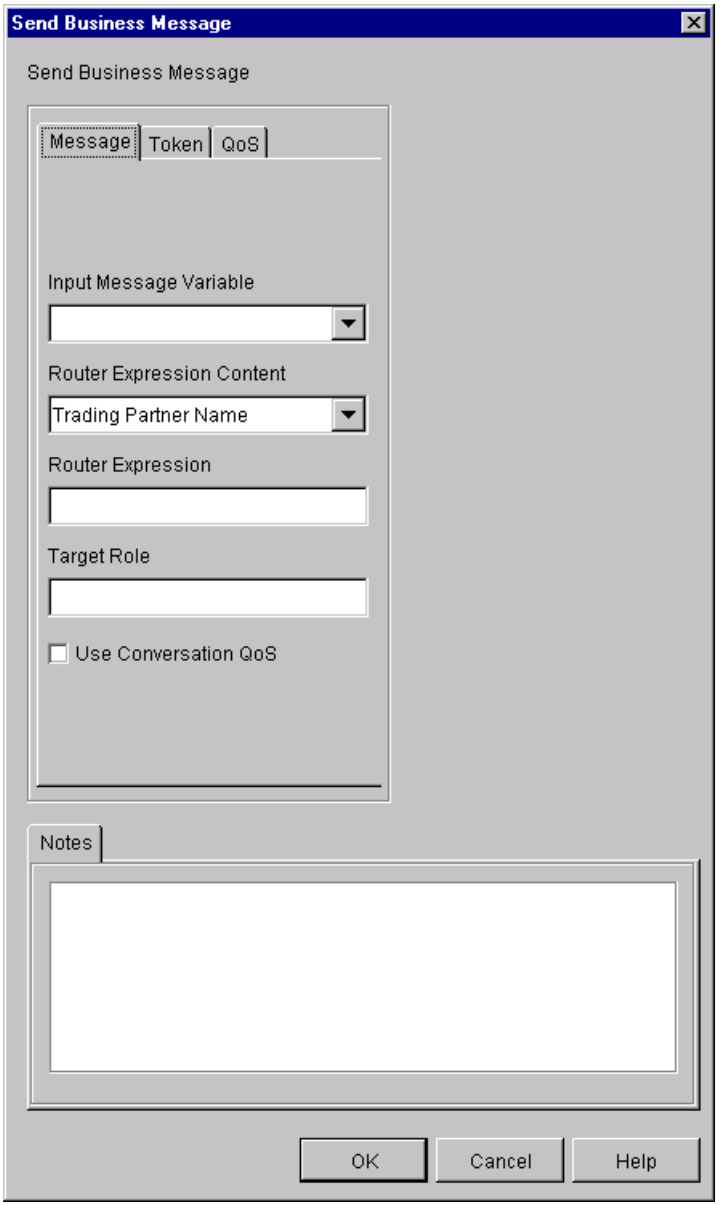
4. Select Send Business Message. Then click OK to display the Send Business Message dialog box.

Which portion of the Send Business Message dialog box is displayed depends on the protocol with which the workflow template was configured. The following sections explain how to specify the contents of this dialog box for each of the protocols supported in the B2B integration plug-in.

Sending Business Messages Using the XOCP 1.1 Protocol

When you select the Send Business Message action in the Actions dialog box for a workflow template configured with the XOCP 1.1 protocol, the following dialog box is displayed.

Figure 5-3 Send Business Message Dialog Box for XOCP Messages



The Send Business Message dialog box displays the following tabs:

- Message—Allows you to specify the input message variable, router expression contents, and the target role.
- Token—Allows you to specify message token information to workflow variables described in "Assigning Message Token Information to Workflow Variables" on page 5-6.
- QoS—Allows you to specify the quality of service at this Send Business Message action level, described in "Defining the Quality of Service for Message Delivery for a Send Business Message Action" on page 5-10.

Complete the following fields in the Send Business Message dialog box.

Table 5-1 Fields in the Send Business Message Dialog Box

Field	Description
Input Message Variable	Name of a workflow variable of type `Java Object` that contains the business message to be sent.
Router Expression Content	Contents of the Router Expression field: a trading partner name, an XPath expression, or a variable name. Router expressions might be overridden by router expressions specified in the WebLogic Integration repository. For more information about routers, see "Advanced Configuration Tasks" in *Administering B2B Integration*. This selection box has the following values from which to choose: ■ Trading Partner Name—The Router Expression field contains a single trading partner name. ■ XPath Expression—The Router Expression field contains an XPath expression. ■ Variable Name—The Router Expression field contains a workflow variable (of type `String`) with the contents of the Xpath expression. The variable is selected from a drop-down list and may have been assigned by the Receive Business Message event.

Table 5-1 Fields in the Send Business Message Dialog Box (Continued)

Field	Description
Router Expression	Router expression that is used when the message is sent. ■ If Trading Partner name is selected, the message is sent to the specified trading partner. ■ If XPath Expression is selected, the message is sent based on the specified XPath expression. If this field is blank, a null filter is used. For more information about router expressions, see "Advanced Configuration Tasks" in *Administering B2B Integration*.
Target Role	The role in the conversation to which the message is sent. Required field.
Use Collaboration QoS	If the workflow template is configured with the XOCP protocol, this check box appears, allowing you to specify whether to use the Quality of Service defined at the template level or at this Send Business Message action level: ■ If the check box is selected, WebLogic Integration uses the QoS information defined at the workflow template definition level, as described in "Defining the Quality of Service for XOCP Message Delivery in the Workflow Template" on page 2-9. ■ If the check box is not selected, WebLogic Integration uses the QoS information defined at this Send Business Message action level, as described in "Defining the Quality of Service for Message Delivery for a Send Business Message Action" on page 5-10.
Notes	Optional descriptive text.

Assigning Message Token Information to Workflow Variables

When a business message is sent by the B2B messaging service, a message token is returned as a Java object at the programming level. The message token provides information about the message, such as the message ID, conversation ID, send success/failure, the delivery status, and the number of recipient destinations after final selection (router and filter evaluations) by the B2B engine. Applications call the `getVariable` method to get access to this variable.

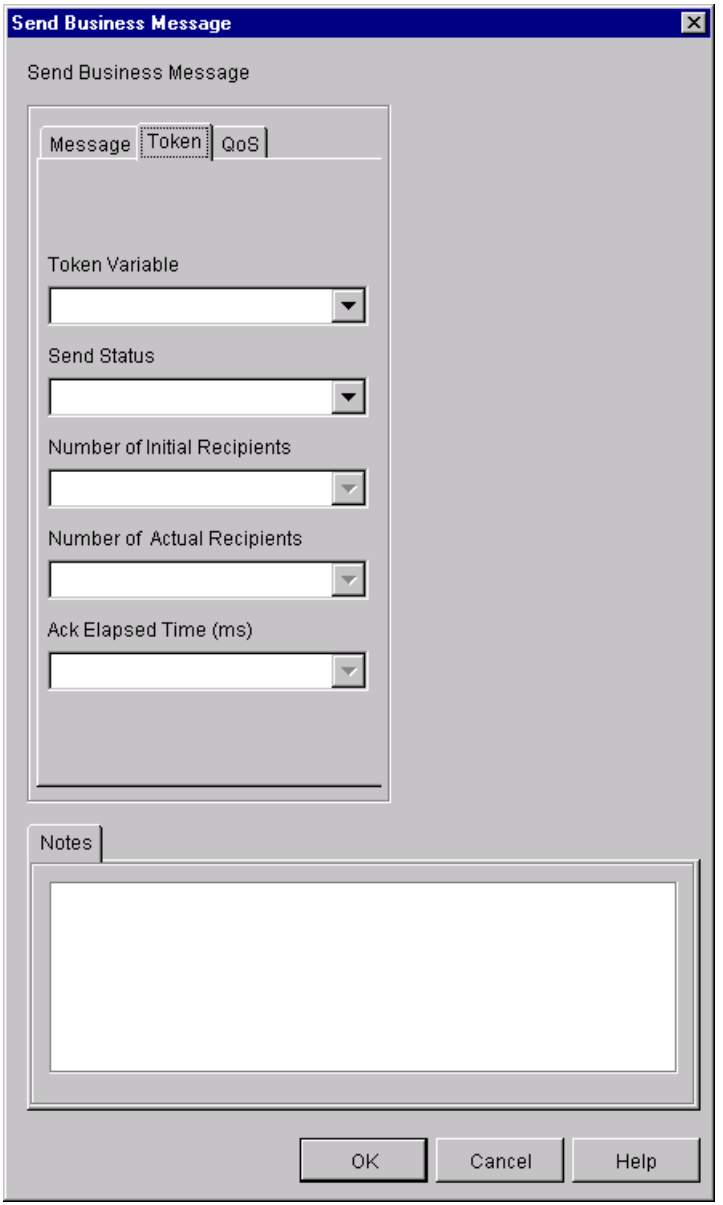
This variable can be defined as an output variable that gets processed after the workflow ends. A message token is represented by the `com.bea.b2b.protocol.messaging.MessageToken` class, which is described in the *BEA WebLogic Integration Javadoc.*

You can configure collaborative workflows to get access to the message token by assigning the token and its associated information to workflow variables. At run time, values are assigned to the workflow instance variables after the Send Business Message action has completed.

To assign a message token and related information to workflow variables:

1. Open the Send Business Message dialog box, as described in "Defining the Workflow to Send Business Messages" on page 5-1.
2. In the Send Business Message dialog box, click the Token tab.

Figure 5-4 Token Tab



Note: The available options in this dialog box depend on the selected Quality of Service settings, as described in "Defining the Quality of Service for XOCP Message Delivery in the Workflow Template" on page 2-9.

3. Complete the following fields in the Token tab.

Table 5-2 Fields in the Token Tab

Field	Description
Token Variable	Assigns the returned message token to a workflow variable of type `Java Object`. This object can then only be passed to a business operation for processing.
Send Status	Indicates whether the message was sent successfully (true for success and false for failure). Assigns the value to a workflow variable of type `Boolean` that can be accessed by the workflow or passed to a business operation.
Number of Initial Recipients	Number of recipients assigned by the B2B engine after the message has traversed the router. Assigns the value to a workflow variable of type `Integer` that can be accessed by the workflow or passed to a business operation. This field appears only if the QoS Messaging Confirmation setting is one of the following selections: ■ Up to XOCP Hub—Delivery confirmation is required when a message reaches the intermediary (default). Select this option to provide basic delivery confirmation with maximum run-time performance. ■ Up to XOCP Hub Router—Delivery confirmation is required when a message reaches the router in the intermediary. This option provides the list of trading partners selected by the router to receive the message. ■ All Destinations—Delivery confirmation is required from all destinations. Select this option to provide the maximum delivery confirmation details. May affect run-time performance.
Number of Actual Recipients	Actual number of recipients. Assigns the value to a workflow variable of type `Integer` that can be accessed by the workflow or passed to a business operation. This field is shown only if the QoS Messaging Confirmation setting is "To all destinations."

Table 5-2 Fields in the Token Tab (Continued)

Field	Description
Ack Elapsed Time (ms)	Time taken, in milliseconds, for acknowledgments to be sent by all recipients. The value in this field is assigned a workflow variable of type Long that can be accessed by the workflow or passed to a business operation. This field is shown only if the QoS Messaging Confirmation setting is "Confirm message delivery to all destinations."
Notes	Optional descriptive text.

If the business message was sent using the synchronous send delivery option, then the message token cannot be used to wait for acknowledgments. If it is used for this purpose, the method returns immediately.

Defining the Quality of Service for Message Delivery for a Send Business Message Action

The Quality of Service (QoS) is a set of attributes that are defined for reliable business message publishing that uses the XOCP protocol. In the Studio, you can define the QoS at the following levels:

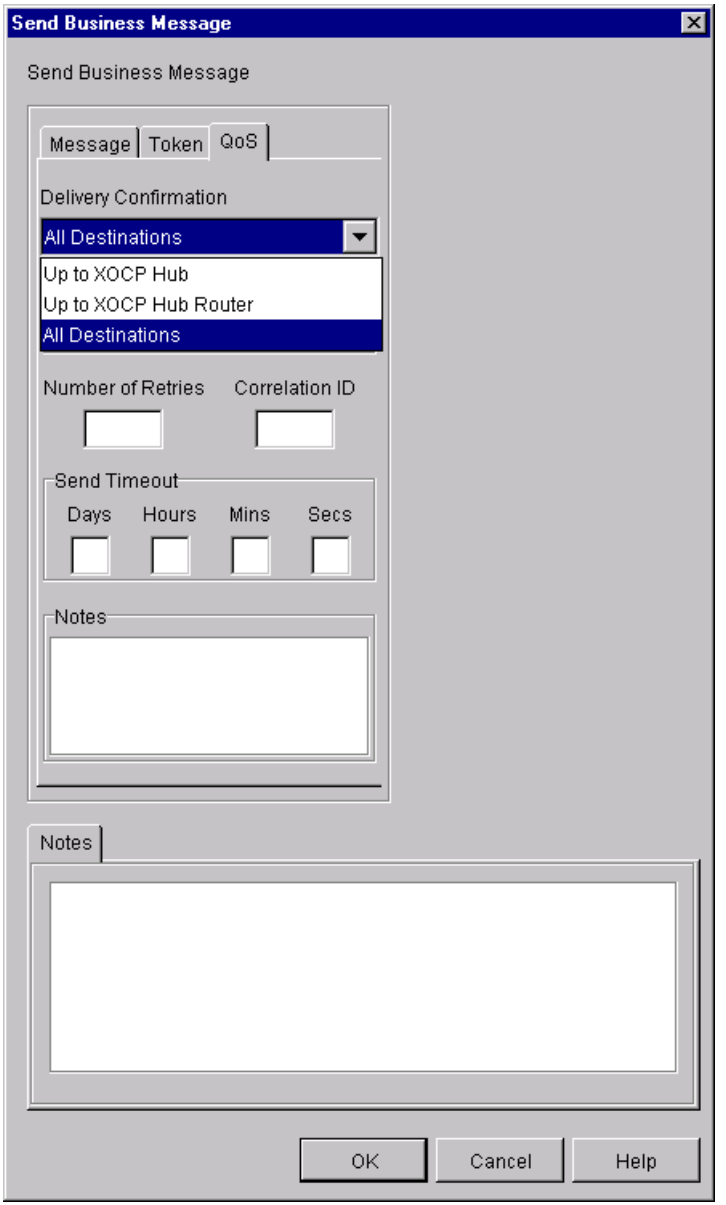
- At the template level, where the settings apply to all Send Business Message actions, unless the QoS is specifically overridden by the definition of the action. For more information, see "Defining the Quality of Service for XOCP Message Delivery in the Workflow Template" on page 2-9.
- At the Send Business Message action level, where the settings apply to the specific action only, but override the settings specified at the template level.

To define QoS at the Send Business Message level:

1. Make sure the Use Conversation QoS check box is unselected in the Send Business Message action dialog box.
2. In the Send Business Message dialog box, select the QoS tab.

   The QoS tab appears.

Figure 5-5 QoS Tab in the Send Business Message Action Dialog Box



3. Complete the fields in the QoS tab as described in Table 2-3.

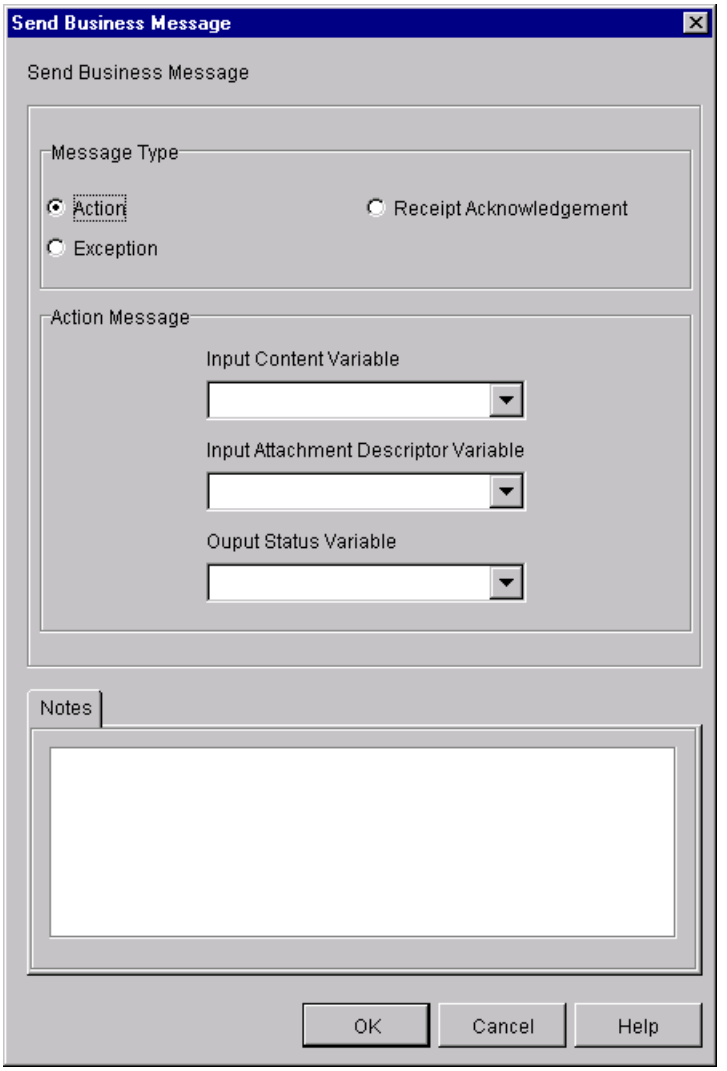
4. Click OK.

Note: The definitions specified here apply to this Send Message action only, not to all send actions within this conversation.

Sending Business Messages Using the RosettaNet 2.0 Protocol

When you select the Send Business Message action in the Actions dialog box for a workflow template configured with the RosettaNet 2.0 protocol, the following dialog box is displayed.

Figure 5-6 Send Business Message Dialog Box for RosettaNet 2.0 Messages



Complete the following fields in the Send Business Message dialog box.

Table 5-3 Buttons and Fields in the Send Business Message Dialog Box

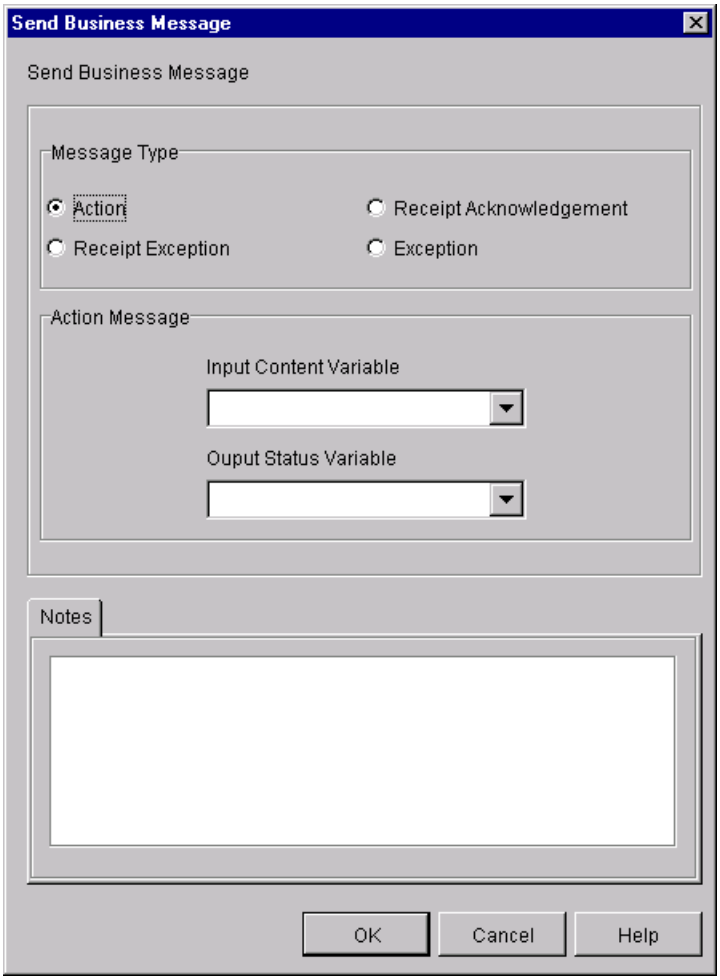
Field	Description
Message Type	Specifies the type of RosettaNet 2.0 message to send. To send a business message, click Action.
Input Content Variable	Specifies the name of the workflow variable of type `XML` that contains the RosettaNet 2.0 business document content.
Input Attachment Descriptor Variable	Specifies the name of the workflow variable of type `XML` that contains the attachment file description to be included in the business message.
Output Status Variable	Specifies the name of the workflow variable of type `Integer` that will contain the message send status.
Notes	Optional descriptive text.

For more information about sending RosettaNet 2.0 messages, see *Implementing RosettaNet for B2B Integration*.

Sending Business Messages Using the RosettaNet 1.1 Protocol

When you select the Send Business Message action in the Actions dialog box for a workflow template configured with the RosettaNet 1.1 protocol, the following dialog box is displayed.

Figure 5-7 Send Business Message Dialog Box for RosettaNet 1.1 Messages



Complete the following fields in the Send Business Message dialog box.

Table 5-4 Buttons and Fields in the Send Business Message Dialog Box

Field	Description
Message Type	Specifies the type of RosettaNet 1.1 message to send. To send a business message, click Action.
Input Content Variable	Specifies the name of the workflow variable of type `XML` that contains the RosettaNet 1.1 business message content.
Output Status Variable	Specifies the name of the workflow variable of type `Integer` that will contain the message send status.
Notes	Optional descriptive text.

For more information about RosettaNet 1.1 messages, including a few examples, see *Implementing RosettaNet for B2B Integration*.

Defining the Workflow to Receive Business Messages

A workflow can receive a business message in the following circumstances:

- When a conversation participant workflow is waiting for the initial business message sent by the conversation initiator workflow. The first business message triggers the start node of the conversation, which is defined as a Business Message start. For more information, see “Defining the Start Node for a Conversation Participant Workflow” on page 3-5.
- When a conversation initiator workflow or conversation participant workflow is waiting in an event node for another message, such as a reply to a request, as described in “Defining Business Message Receive Events” on page 5-17.

The following sections describe procedures for setting up your workflow to receive business messages for each of the protocols supported in the B2B integration plug-in.

Defining Business Message Receive Events

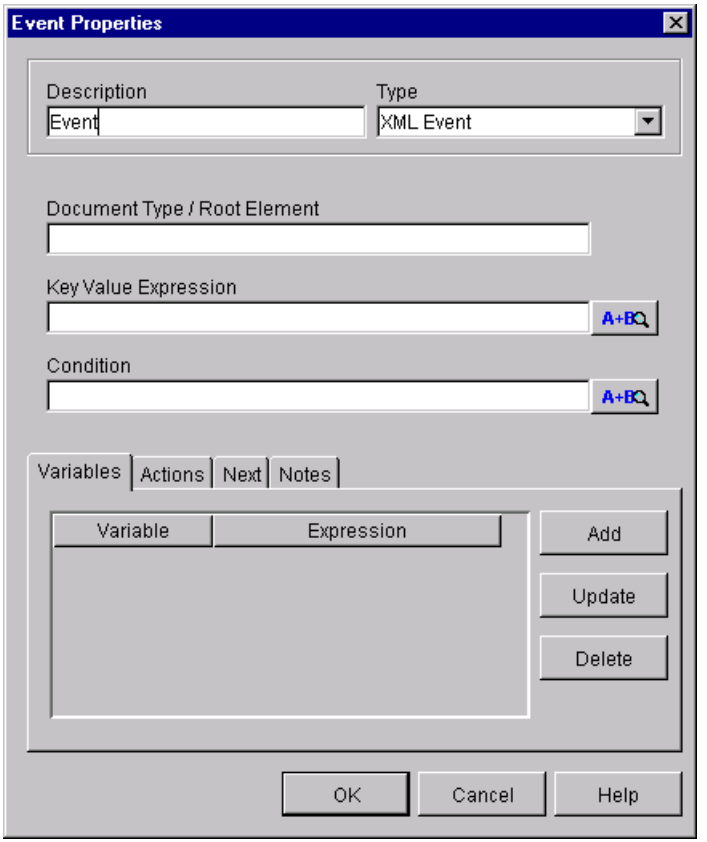
If a workflow waits to receive a business message, such as a reply to a request or a subsequent (not an initial) request, you must define a Business Message Receive event. This event is triggered at run time when the appropriate business message is received in the conversation.

To define a Business Message Receive event:

1. Display or add an event node as described in "Defining Workflow Templates" in *Using the WebLogic Integration Studio*.

2. Double-click the event shape or right-click it in the folder tree and select the Properties command to display the Event Properties dialog box.

Figure 5-8 Event Properties Dialog Box



3. In the Description field, enter a meaningful name for the Receive Business Message event.

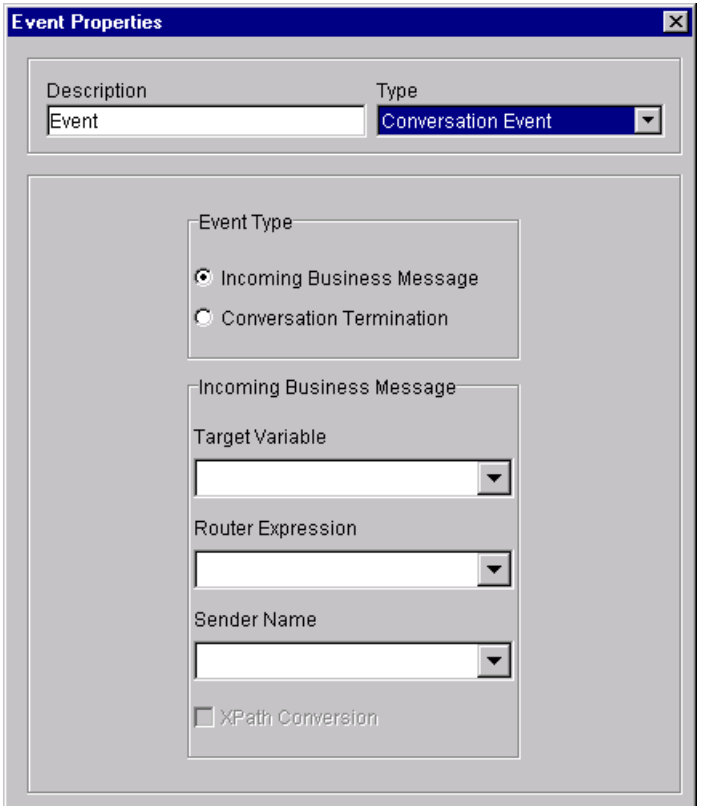
4. In the drop-down list labeled Type, select Collaboration event.

The display of the Event Properties dialog box is refreshed. Which contents are displayed depends upon the protocol with which the workflow template has been configured. The following sections explain how to define the Receive Business Event for each of the protocols supported by the B2B integration plug-in.

Defining a Business Message Receive Event for the XOCP 1.1 Protocol

If the workflow template is configured with the XOCP 1.1 protocol, when you select the Conversation event type, the following Event Properties dialog box is refreshed with the following contents.

Figure 5-9 Event Properties Dialog Box for Defining a Receive XOCP Message Event



To configure a Business Message Receive event for an XOCP message, select values for the following fields in the Event Properties dialog box.

Table 5-5 Fields in the Event Properties Dialog Box

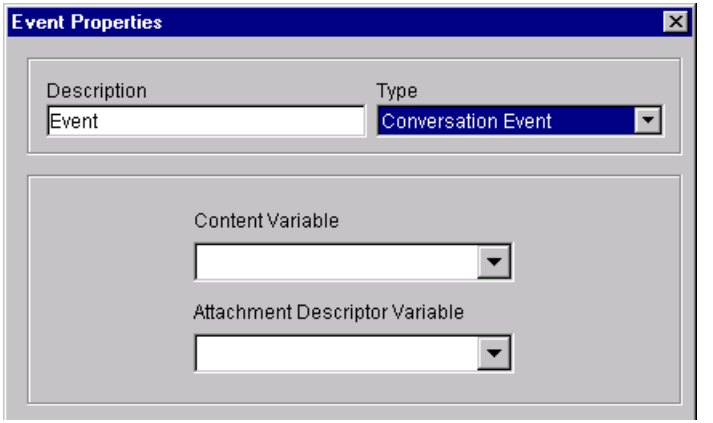
Variable	Description
Event Type	Select Incoming Business Message to define the event to wait for an incoming business message. (Selecting a method of conversation termination is explained in "Defining the End of Conversation Participant Workflows" on page 6-4.)
Target Variable	Name of a target workflow variable (of type `Java Object`) in which to store the business message. Required field.
Router Expression	Name of a workflow variable (of type `String`) in which to store an XPath expression. The value represents the XPath expression used by the sender to send the message. This XPath expression can be used later, in a router expression, to publish a reply to the current message. Optional field.
Sender Name	Name of a workflow variable (of type `String`) in which to store the name of the Trading Partner that sent the message. If the Convert Sender's Name to XPath check box is selected, then this name is converted to an XPath expression.
XPath Conversion	If this check box is selected, the contents of the variable specified in the Sender's Name field are converted to an XPath expression suitable for use in the Send Business Message action. If it is not selected, the Sender's Name variable is used as the actual name of the sending Trading Partner.

At run time, when the business message is received, the event is triggered and the target variable is set to the business message that was just received. If a router variable is specified, it contains an XPath expression that can be used to reply to the sender. For more information, see "Defining the Workflow to Send Business Messages" on page 5-1.

Defining a Business Message Receive Event for the RosettaNet 2.0 Protocol

If the workflow template is configured with the RosettaNet 2.0 protocol, the display of the Event Properties dialog box is refreshed with the following contents when you choose the Conversation event type.

Figure 5-10 Event Properties Dialog Box for Defining a Receive RosettaNet 2.0 Message Event



To configure a Business Message Receive event for a RosettaNet 2.0 message, select values for the following fields in the Event Properties dialog box.

Table 5-6 Fields in the Event Properties Dialog Box

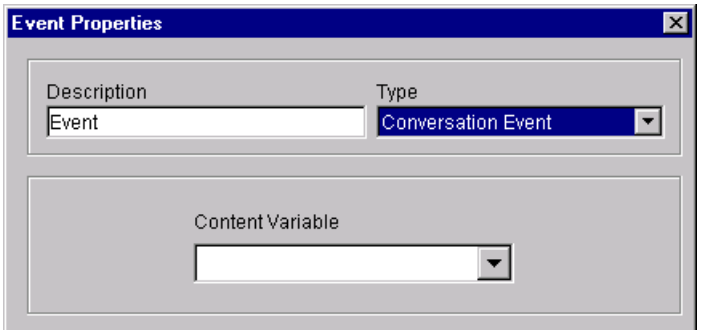
Variable	Description
Content Variable	Name of a target workflow variable of type `XML` in which to store the RosettaNet 2.0 XML business document content.
Attachment Descriptor Variable	Name of a target workflow variable of type `XML` in which to store the RosettaNet 2.0 message attachment content.

At run time, when the business message is received, the event is triggered and the target variables are set to the business message that was just received. For more information about defining a business message receive event for workflow templates configured with the RosettaNet 2.0 protocol, including examples, see *Implementing RosettaNet for B2B Integration*.

Defining a Business Message Receive Event for the RosettaNet 1.1 Protocol

If the workflow template is configured with the RosettaNet 1.1 protocol, when you select the Conversation event type, the display of the Event Properties dialog box is refreshed with the following contents.

Figure 5-11 Event Properties Dialog Box for Defining a Receive RosettaNet 1.1 Message Event



To configure a Business Message Receive event for a RosettaNet 1.1 message, select the name of a target workflow variable of type `XML` in which to store the RosettaNet 1.1 XML business document content.

At run time, when the business message is received, the event is triggered and the target variables are set to the business message that was just received. For more information about defining a business message receive event for workflow templates configured with the RosettaNet 1.1 protocol, including examples, see *Implementing RosettaNet for B2B Integration.*

6 Ending Collaborative Workflows

The following sections describe key tasks associated with ending collaborative workflows:

- Defining Conversation Termination
- Defining Workflow End

Defining Conversation Termination

A conversation is terminated when the conversation initiator workflow reaches a done state. Conversation participant workflows can end their participation in a conversation before the conversation is terminated.

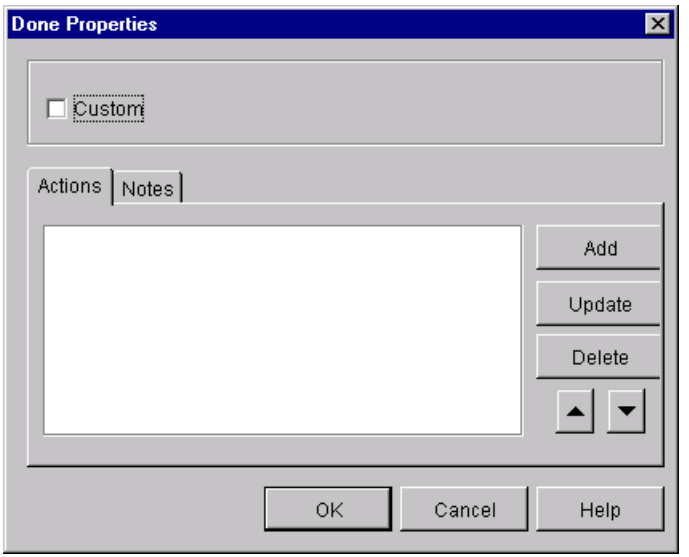
Defining the Termination of Conversation Initiator Workflows

For a conversation initiator workflow, you can define the conversation termination property (terminate with success or failure) for any done node in the workflow. Once a done node is reached in the workflow, the running instance of the workflow is marked done, regardless of whether the active workflow has reached all the done nodes. A conversation initiator workflow can terminate a conversation, but other participants in the conversation cannot.

To define the termination for a conversation initiator workflow:

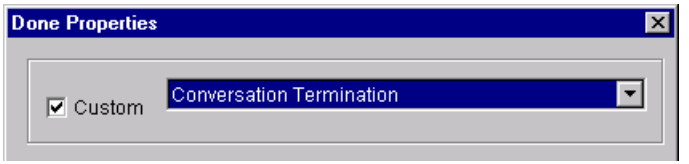
1. Add or view a done shape, as described in "Defining Workflow Templates" in *Using the WebLogic Integration Studio*.
2. Double-click the done shape or right-click it in the folder tree. Select Properties to display the Done Properties dialog box.

Figure 6-1 Done Properties Dialog Box



3. Click Custom. A drop-down list for each of the plug-ins installed in the Studio is displayed.

Figure 6-2 Custom Done Properties Drop-Down List



4. Select Collaboration Termination.

What you see in the Done Properties dialog box depends on the protocol defined for the workflow template.

Done Properties for Each Supported Protocol

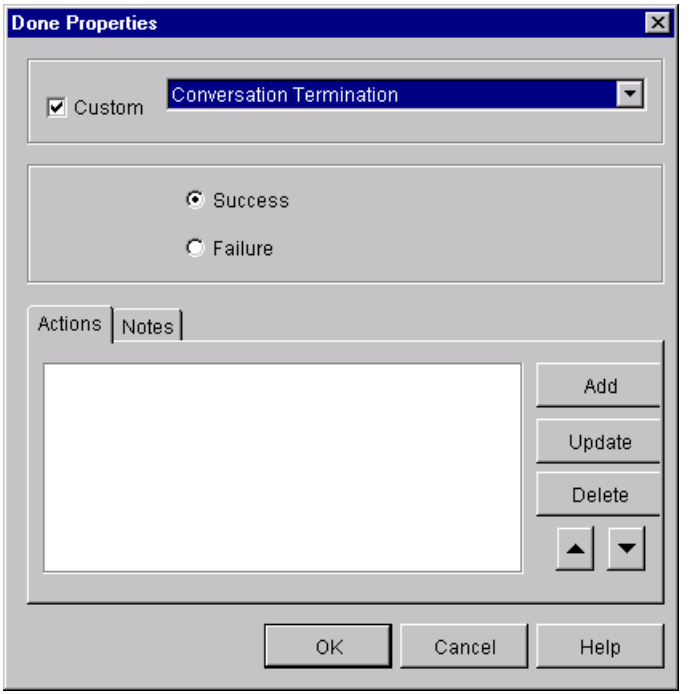
The following sections describe the Done Properties dialog box for each supported protocol:

- Done Properties Dialog Box for the XOCP 1.1 Protocol
- Done Properties Dialog Box for the RosettaNet 1.1 or 2.0 Protocols

Done Properties Dialog Box for the XOCP 1.1 Protocol

For workflow templates defined for the XOCP 1.1 protocol, the Done Properties dialog box appears as shown in the following figure after you have chosen a custom Done property using the Conversation Termination property.

Figure 6-3 Done Properties Dialog Box for the XOCP 1.1 Protocol



Select a conversation termination option in the Done Properties dialog box.

Table 6-1 Exchange Termination Options in the Done Properties Dialog Box

Field	Description
Success	Select this option if the conversation should be terminated with a SUCCESS result (default). The conversation is terminated after the actions for this state are done. The SUCCESS result indicates that the workflow instance completed successfully. The participants of the conversation are notified (if possible) that the conversation is being terminated.
Failure	Select this option if the conversation should be terminated with a FAILURE result. The conversation is terminated after the actions for this state are done. The FAILURE result indicates that the workflow instance encountered conversation-specific or application-specific errors. The participants of the conversation are notified (if possible) that the conversation is being terminated.

Done Properties Dialog Box for the RosettaNet 1.1 or 2.0 Protocols

If your workflow is configured with the RosettaNet 1.1 or 2.0 protocol, conversation termination is not handled in the Done node of the workflow. If you select the Conversation Termination property, the following message is displayed:

```
This feature is not available for the current conversation settings
```

For information about terminating RosettaNet-based conversations, see *Implementing RosettaNet for B2B Integration.*

Defining the End of Conversation Participant Workflows

A conversation participant workflow has defined conversation properties, a Business Message start property, and (optionally for XOCP-based workflows) a Conversation Terminate event. The Conversation Terminate event may be used in a participant workflow to wait for a conversation termination signal from the conversation initiator. It allows a participant workflow to perform additional processing (such as housekeeping operations) based on the status of the conversation termination.

Note: The use of this event is optional, and it is relevant only to the XOCP protocol. A workflow that does not wait for this event can leave the conversation by simply ending the workflow (a Done node).

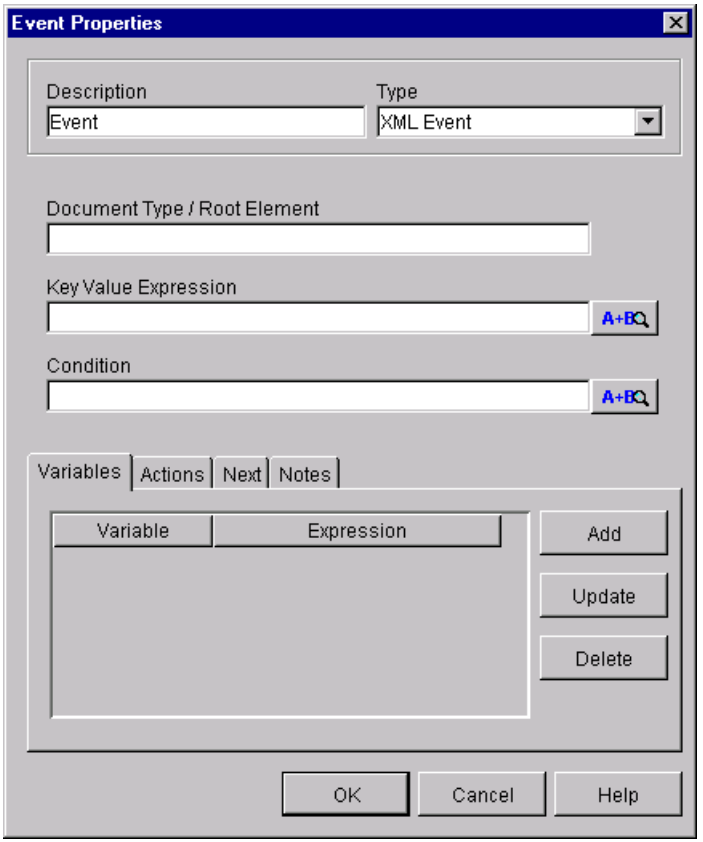
A workflow event shape represents a notification node. The workflow waits for a conversation termination message to trigger the event. Upon that trigger, actions defined within the event can be executed and/or workflow variables can be set.

To add a Conversation Terminate event to a conversation participant workflow:

1. Display or add an event shape as described in "Defining Workflow Templates" in *Using the WebLogic Integration Studio*.

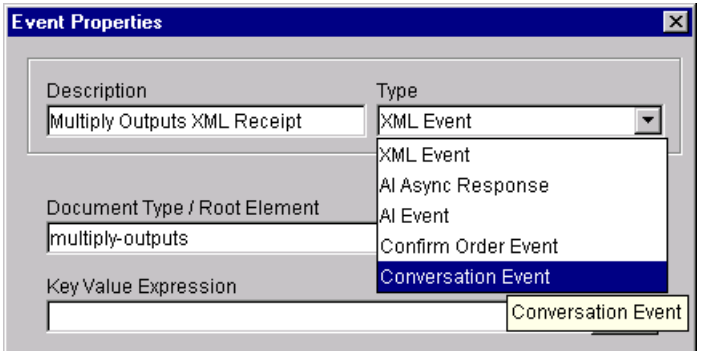
2. Double-click the event shape or right-click it in the folder tree. Select the Properties command to display the Event Properties dialog box.

Figure 6-4 Event Properties Dialog Box



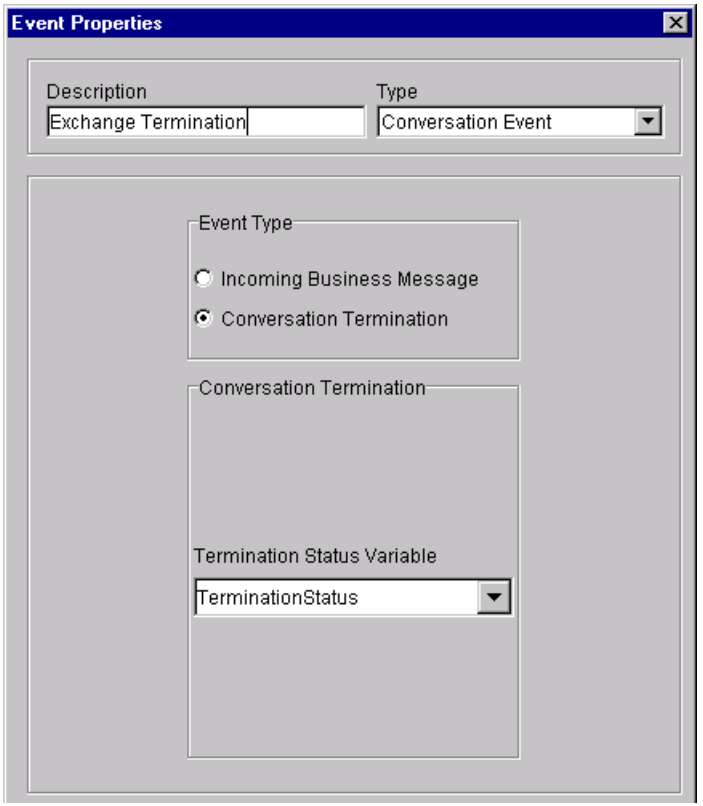
3. Enter text in the Description field, and select Conversation Event from the Type drop-down list.

Figure 6-5 Choosing a Collaboration Event



After you have selected Conversation Event as the event type in the Event Properties dialog box, the display of the Event Properties dialog box is refreshed as shown in the following figure.

Figure 6-6 End of Conversation Collaboration Event for the XOCP 1.1 Protocol



4. In the Event Type selection box, select Conversation Termination.
5. Select a workflow variable of type `Boolean` in which the termination status variable can be stored. This variable will be set to one of the values listed in the following table.

Table 6-2 Terminate Status Options in the Event Properties Dialog Box

Option	Description
True	Indicates that the conversation was terminated with a `SUCCESS` value.
False	Indicates that the initiator terminated the conversation with a `FAILURE` value.

Note: You must explicitly create this Boolean variable before selecting it in this dialog box. For more information, see "About Using Workflow Variables" on page 2-15.

6. Click OK to save your changes.

The Studio assigns the value of the conversation termination status to a workflow variable of type `Boolean`, which can be accessed by the workflow or passed to a business operation. The developer of the workflow should take appropriate actions based on this value.

Defining Workflow End

All collaborative workflows require at least one done node. For conversation initiator workflows, defining the done node is explained in "Defining the Termination of Conversation Initiator Workflows" on page 6-1. For conversation participant workflows, receiving a conversation termination event is entirely separate from defining the end of the workflow. Typically, conversation participant workflows have at least one done node that simply ends the workflow. There are no B2B-specific actions or events that can be configured in the done node of conversation participant workflows, regardless of protocol.

As mentioned in "Defining the End of Conversation Participant Workflows" on page 6-4, conversation participant workflows do not need to handle a conversation termination event. These workflows can simply end, and when they do, they are automatically separated from the conversation in which they were participating.

For more information about configuring Done nodes, see "Defining Workflow Templates" in *Using the WebLogic Integration Studio*.

Index

C

D

E

F

Q

R

S

T

U

V

W

X