BEA WebLogic Integration™

Using the WebLogic Integration Studio

Release 7.0
Document Date: June 2002

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Using the WebLogic Integration Studio

Part Number	Date	Software Version
N/A	June 2002	7.0

Contents

2. Using the Studio Interface

3. Administering Data

4. Configuring Workflow Resources

5. Defining Workflow Templates

6. Defining Actions

7. Working with XML Entities

8. Using Workflow Expressions

9. Handling Workflow Exceptions

10. Monitoring Workflows

11. Importing and Exporting Workflow Packages

Index

About This Document

This document is a user guide and reference for business process management (BPM) functions that you can perform with the BEA WebLogic Integration™ Studio. For a hands-on, step-by-step introduction to the process of defining a workflow in the Studio, see *Learning to Use BPM with WebLogic Integration*.

This document is organized as follows:

- Chapter 1, "Introduction to the WebLogic Integration Studio," provides an overview of WebLogic Integration workflow components, methods of integration with internal and external resources, a description of Studio design work models and tasks, and additional tools provided by the Studio.
- Chapter 2, "Using the Studio Interface," explains how to log in to and exit from the Studio. It also describes the Studio user interface.
- Chapter 3, "Administering Data," explains how to create and maintain organizations, users, roles, business calendars, and task routings. It also describes security concepts and explains how to set up user and role permissions.
- Chapter 4, "Configuring Workflow Resources," describes how to configure external and custom components and resources to make them available for workflow designers, including custom plug-ins, Java components such as EJBs, XML-based events, and the WebLogic Integration XML repository.
- Chapter 5, "Defining Workflow Templates," explains how to define and maintain WebLogic Integration workflow templates, template definitions, and workflow components, such as nodes, connections and variables, and properties for each of the six node types. It also provides some guidelines for workflow design.
- Chapter 6, "Defining Actions," describes how to use task, workflow, integration, and miscellaneous actions to accomplish workflow activities. It also provides some guidelines for workflow design.

- Chapter 7, “Working with XML Entities,” describes the Studio’s built-in XML editor, and it explains how to retrieve, store, compose, and edit XML entities from various sources, including the WebLogic Integration XML repository.
- Chapter 8, “Using Workflow Expressions,” describes the workflow expression language, and it explains how to create workflow expressions using the Expression Builder and the XPath Wizard. The chapter also provides a reference listing of workflow functions and expression syntax.
- Chapter 9, “Handling Workflow Exceptions,” explains how to define and call exception handlers, and how to use exception handling actions.
- Chapter 10, “Monitoring Workflows,” describes the workflow monitoring facility, and it explains how to view the status and update running workflows and their variables. It also describes how to collect post run-time data by generating workload reports and statistics.
- Chapter 11, “Importing and Exporting Workflow Packages,” explains how to import and export complete workflow packages from and to Java archive files for sharing among different systems.

What You Need to Know

This document is intended for integration specialists, system administrators, business analysts, and application developers who use the WebLogic Integration Studio to design, develop, and administer workflows.

This document is based on the assumption that you have some familiarity with the Java 2 Enterprise Edition™ (J2EE) platform, Enterprise JavaBeans™ (EJBs), BEA WebLogic Server™, eXtensible Markup Language (XML), XPath language, Java Message Service (JMS), and the Java programming language.

Related Information

The following BEA WebLogic Integration documents contain information that is relevant to BPM:

- *Learning to Use BPM with WebLogic Integration*
- *Best Practices in Designing BPM Workflows*
- *Using the WebLogic Integration Worklist*
- *Programming BPM Client Applications*
- *Programming BPM Plug-Ins for WebLogic Integration*
- *BEA WebLogic Integration Javadoc*
- *Designing BEA WebLogic Integration Solutions*

For information about using the Studio for application integration, see *Using Application Integration*.

For information about using the Studio for B2B integration, see *Creating Workflows for B2B Integration*.

For information about using the Studio for data integration, see *Using the Data Integration Plug-In*.

e-docs Web Site

BEA product documentation is available from the BEA corporate Web site. On the BEA Home page, click Product Documentation or go directly to the e-docs Product Documentation page at http://e-docs.bea.com.

How to Print the 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 Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Integration documentation Home page, click the PDF files button, and select the document you want to print.

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

Contact Us!

Your feedback on the BEA 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 which release of the WebLogic Integration documentation you are using.

If you have any questions about this version of BEA WebLogic Integration, or if you have problems installing and running BEA WebLogic Integration, contact BEA Customer Support through BEA WebSupport at http://www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

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
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 italic text`	Identifies variables in code. *Example*: `String` *`expr`*

Convention	Item
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
{ }	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 Introduction to the WebLogic Integration Studio

WebLogic Integration provides a workflow management system that you use to automate business processes. A business process is a series of interconnected business activities that produce some desired result. A graphical representation of a business process is a *workflow*. You define and monitor workflows using the WebLogic Integration Studio, a WebLogic Integration client application.

The following sections provide an overview of Studio workflow concepts, modeling constructs, tasks, and work models:

- About Business Process Management in WebLogic Integration
- About the Studio
- Modeling Business Data
- Modeling Business Processes
- Integrating Users, Applications, and Data
- Workflow Design Approaches and Tasks
- Studio Tools

About Business Process Management in WebLogic Integration

In an e-business world, businesses must perform effectively at a rapid pace. To achieve this level of performance, many companies automate their business processes by using a business process management system: a system that defines, manages, and executes business processes using software. The order in which business activities are executed in the process is driven by a computer representation of the process called a workflow.

A workflow automates a business process by controlling the sequence of activities in the process, and by invoking the appropriate resources required by the various activities in the process. These resources might be software components that perform a required activity, or they might be people who respond to messages generated by the business process.

To achieve this level of workflow automation, a workflow management system provides support in three broad areas:

- Workflow definition and execution—capturing the definition of the workflow, managing the execution of workflow processes in an operational environment, and sequencing the various activities to be performed.
- Data administration—managing the people involved in a workflow, rerouting tasks when needed, and maintaining business calendars that govern the schedule of workflow activities.
- Workflow monitoring—tracking the status of workflow processes, and dynamically configuring the workflow at run time.

WebLogic Integration supports all three areas of workflow management: it supports the execution of workflows using its run-time process engine, and the definition and monitoring of workflows with the WebLogic Integration Studio.

An automated business process does not, however, entirely replace people with computers. People might still be involved in an automated process, either by making discretionary decisions or by handling exceptions or problems as they arise. Thus, WebLogic Integration also includes a Worklist application that end users use to view and execute manual tasks assigned to them. For more information about the Worklist

application, see *Using the WebLogic Integration Worklist*. Alternatively, programmers can develop their own custom worklist applications by using the WebLogic Integration API. For more information, see *Programming BPM Client Applications*.

Note: The Worklist client application is being deprecated as of this release of WebLogic Integration. For information about the features that are replacing it, see the *BEA WebLogic Integration Release Notes*.

About the Studio

The WebLogic Integration Studio is a client application with a graphical user interface that provides functionality in the three broad areas of workflow design, workflow monitoring, and data administration, as follows:

- Data administration functions
  - Modeling business units and users of the system
  - Configuring levels of permission for users
  - Rerouting tasks from one user to another for a specified period of time
  - Creating business calendars that are used to control the execution of workflows
- Workflow design functions
  - Drawing graphical process flow diagrams
  - Defining variables to store run-time data
  - Defining processes to handle run-time exceptions
  - Defining interfaces to external components
- Workflow monitoring functions
  - Displaying the status of running workflows
  - Modifying the tasks associated with running workflows, such as by reassigning tasks or by forcing work to be redone
  - Displaying the workload status of the system in graphical form

- Viewing historical data for workflows to identify bottlenecks and inefficiencies
- Viewing user or role worklists to maintain workflows dynamically

Modeling Business Data

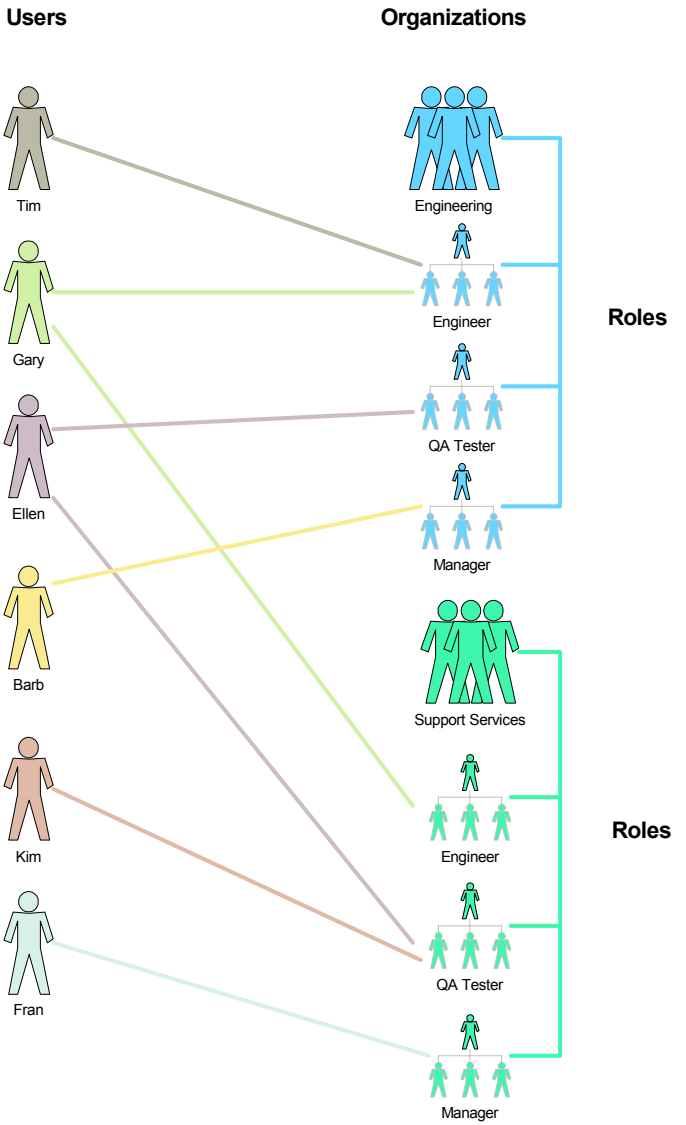
In the WebLogic Integration Studio, organizational data is subdivided into three categories:

- Organization—an entire company; a division, region, or location within a company; or any other distinction that is relevant to the particular business of a company.
- Role—a common area of responsibility, ability, or authorization level that is shared by a group of individuals who belong to a particular organization.
- User—an individual assigned to a role who has the necessary permission to perform a particular task in the workflow, such as responding to messages generated by a workflow.

Roles allow groups of users to execute manual tasks according to a generic business role within the organization. Roles are defined uniquely within organizations. This allows you to divide your organization into organizational units, and to re-use role names that can have different groups of users attached to them.

Although roles are defined uniquely within organizations, users can belong to one or more organizations, and to one or more roles with those organizations, as the following figure illustrates.

Figure 1-1 Relationship Between Organizations, Users, and Roles



In this example, there are two organizations, Engineering and Support Services, each containing their own roles of Engineer, QA Tester, and Manager. The relationships depicted are summarized in the following table:

Table 1-1 Organizations, Roles and Users

Organization	Role	Members
Engineering	Engineer	Tim, Gary
	QA Tester	Ellen
	Manager	Barbara
Support Services	Engineer	Gary
	QA Tester	Ellen, Kim
	Manager	Fran

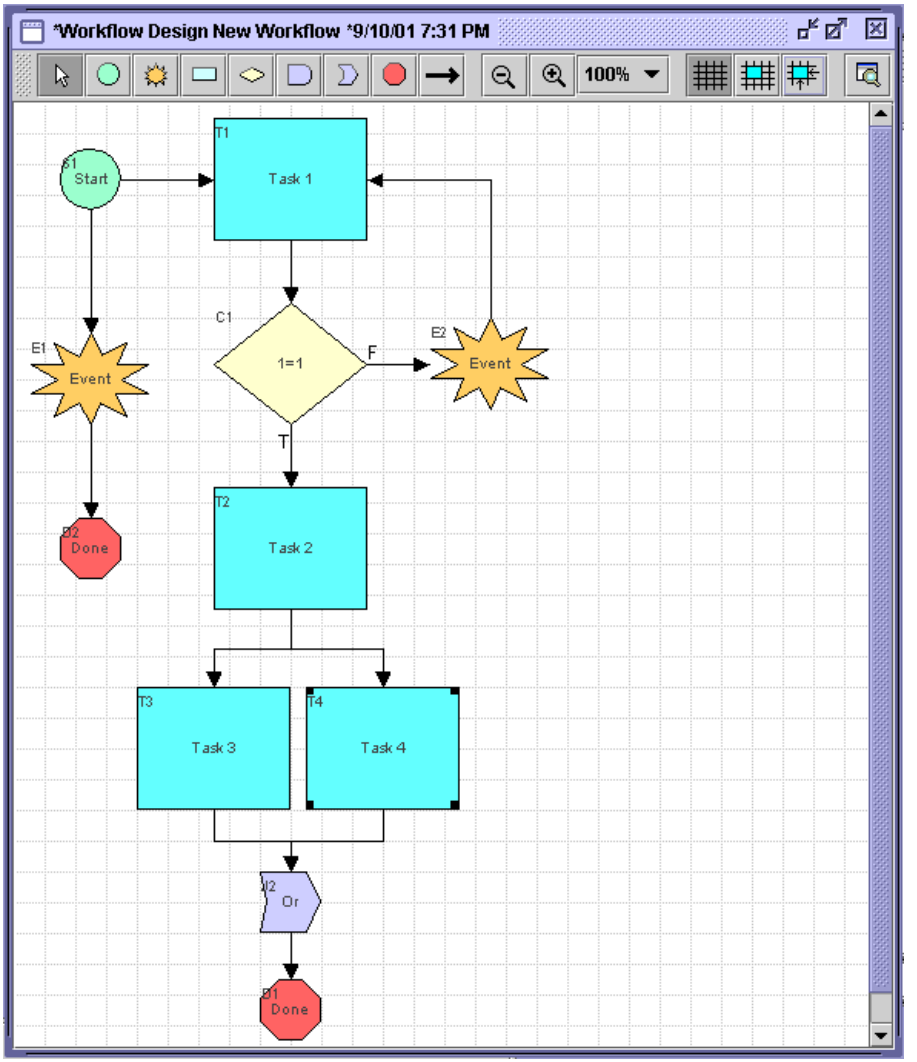
The other business data you can model are business rules, such as schedules and workflow routings. You model working hours and schedules with business *calendars* that can be associated with organizations, roles, and users. You can also create *routing* specifications that redirect activities to different users or roles for specified periods of time.

Modeling Business Processes

In the Studio, business processes are modelled, diagrammed, and saved as workflow *templates* in a database. These templates can be associated with multiple organizations, and are essentially empty containers for storing different workflow versions. Templates contain *template definitions*, which serve as different versions of the same workflow, and are distinguished by effective and expiry dates that determine a range of time in which each version may be available for *instantiation*, that is, placement in the run-time environment.

The template definition is where you represent the business processes you wish to model, by drawing and connecting shapes that make up the flow. Program control is represented visually by *nodes* and *connections*, as shown in the following figure.

Figure 1-2 Studio Workflow Template Definition: Design Area



In addition, template definitions contain *actions* and *exception handlers* that execute different activities at run time and *variables* that collect, store, and distribute run-time data. Template definition components are discussed in more detail in the following sections.

Nodes

Nodes are geometric shapes that you use to depict a workflow graphically. The Studio provides the following seven nodes:

- Start — represents the beginning of a workflow.
- Task — represents a user-assigned task or a grouping of actions (see "Actions") that form a unit of work.
- Event — represents a wait state that can be triggered by the receipt of an XML message. When the event is triggered, the flow proceeds.
- AND Join — represents a merge of multiple workflow paths into a single path such that all the workflow paths linked by the join node must have completed before the flow proceeds.
- OR join — represents a merge of multiple workflow paths into a single path such that only one of the workflow paths linked by the join node must have completed before the flow proceeds.
- Decision — represents a condition (specified in the node) that must be evaluated to either true or false. The result of the evaluation determines the path the workflow follows.
- Done — represents the end of a workflow.

Actions

Actions are, in a sense, the basic building blocks of a workflow because they define the actual behavior of the workflow. An action can be as simple as assigning a task to a user or as complicated as sending XML messages or invoking Enterprise JavaBean (EJB) methods. Actions may be added to all nodes (except Joins), to exception handlers, and even to other actions.

There are several types of actions:

- Task actions — used to assign manual tasks to users and roles, and to control the execution of Task nodes.

- Workflow actions — used to manage an entire workflow, such as stopping the current workflow or starting a sub-workflow.
- Integration actions — used to integrate the workflow with external software components or applications. This can be done by, for example, calling an executable program or sending an XML message to another application. The operations that are carried out by integration actions are discussed in more detail in "Integrating Users, Applications, and Data" on page 1-10.
- Exception handling actions — used to invoke and exit an exception handler, and to make an exception handler the active one for a workflow.
- Miscellaneous actions — used to perform additional actions, such as sending e-mail, making an audit entry, or canceling a workflow event.
- Custom actions (plug-ins) — actions that are programmed for the WebLogic Integration plug-in framework, and that you access from the Studio. For more information about the plug-in framework, see *Programming BPM Plug-Ins for WebLogic Integration*.

Variables

Variables hold run-time values and data obtained usually from outside sources, such as Java components and incoming XML documents. They can also be set to constant values by workflow actions. Variables can be used by the workflow for several purposes, such as evaluating a condition in a decision node, creating a label for a template definition, or holding workflow run-time information.

WebLogic Integration supports the following types of variables: Boolean, Date, Double, Entity EJB, Integer, Java Object, Session EJB, String, and XML. Data types can be extended through the use of plug-in types developed for the plug-in framework.

Exception Handlers

Exception handlers are used to generate, trap, and respond to exception conditions that are generated internally or externally. Exceptions can be abnormal conditions that you identify at design time in the workflow that you want to trap, or they can be server exceptions that you trap and respond to accordingly. Workflows can contain custom-defined exception handlers consisting of a series of actions that you specify.

Integrating Users, Applications, and Data

As the design tool for WebLogic Integration, the Studio goes beyond business process management, and helps to integrate all human and automated operations in the e-business cycle. The Studio provides actions for integrating Web-based and back-end applications, interacting with clients inside and outside of the firewall, and transforming data. For more information about the B2B integration, application integration, and data integration plug-in functionality for BPM, see the following documents:

- *Using Application Integration*
- *Creating Workflows for B2B Integration*
- *Using the Data Integration Plug-In*

The following sections describe some of the integration scenarios offered by basic Studio actions, and the means by which workflows interface with external components, via XML messaging, business operations, and custom-developed plug-ins.

Integrating Users and Client Applications

Workflows interact with system users via the Worklist or a custom client application using the following methods:

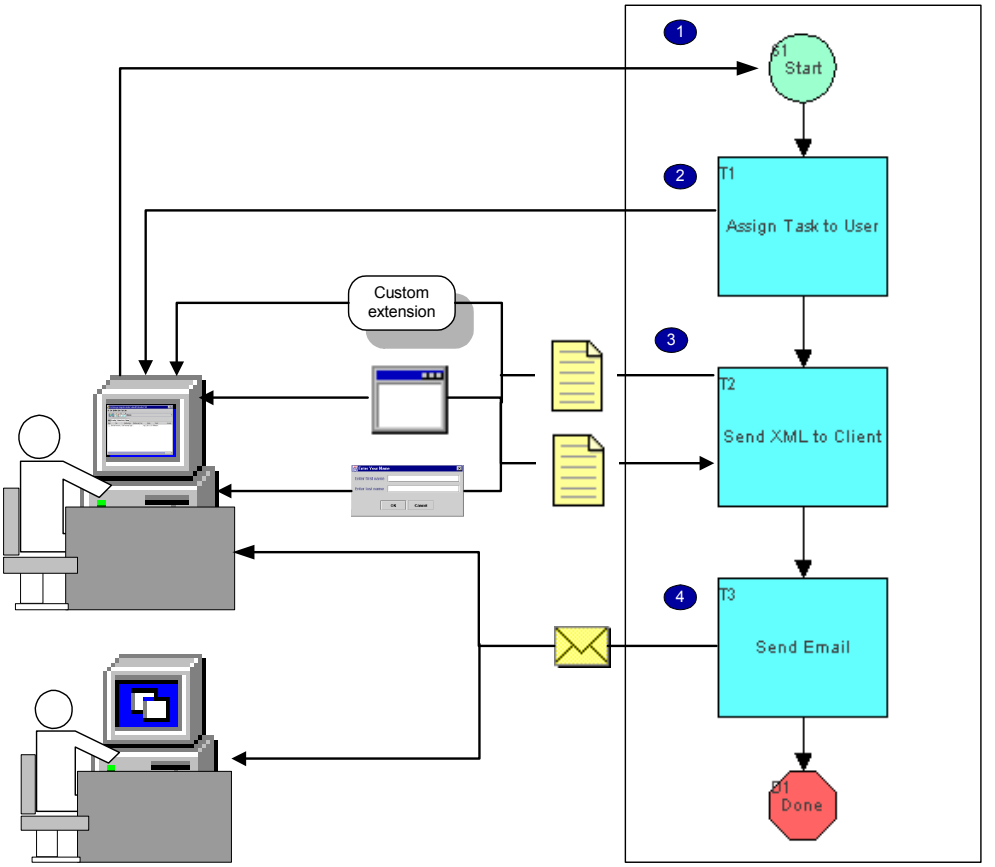
- Directly through the Worklist or custom client

- Internal XML/JMS messaging with the Worklist or a custom client application to perform additional operations, such as displaying forms
- E-mail.

Workflows can also interact with users outside the system by e-mail.

The following figure shows the various interactions that be achieved between workflows and client applications and users. Each scenario is described below the figure.

Figure 1-3 Integrating Users and Client Applications



1. The workflow is initiated by a Worklist user.
2. Assigning a task to a user sends a notification to a Worklist user to perform a task.
3. Sending XML to client sends an XML message to the Worklist application, instructing it to display a message prompt or form, or call an executable program or custom component on the client. XML messages are sent in response from the Worklist application back to the workflow.
4. Sending e-mail allows the process engine to send e-mail to Worklist users and clients outside the WebLogic Integration system.

Integrating External Components and Applications

Workflows can integrate the following kinds of external software components:

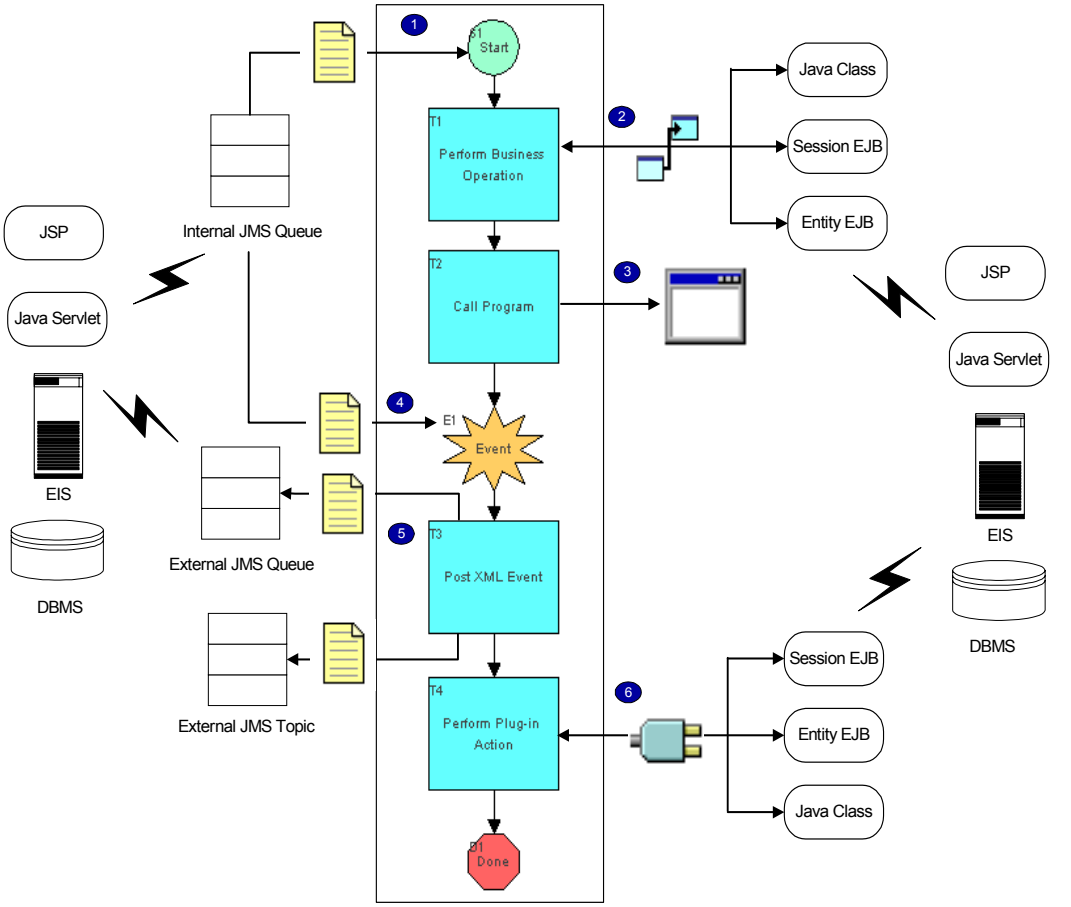
- External systems, including Web components, such as Java Server Pages (JSPs) and servlets
- Enterprise information systems, such as legacy systems, packaged enterprise applications
- Database management systems (DBMSs)

Workflows can then exchange data with these external components using the following methods:

- XML/JMS messaging
- Workflow *business operations* that represent with EJB or Java class methods.
- Plug-in components consisting of Java classes and EJBs using the plug-in framework

The following figure shows the different means by which workflows can interface with external components. Each scenario is described below the figure.

Figure 1-4 Integrating External Components and Applications



1. The workflow is triggered, and data obtained, by the receipt of an XML message on an internal JMS queue from a sending application.
2. Performing a business operation invokes pre-existing Java components or components written specifically for the workflow application, and passes parameters directly from the workflow to the component and back again.
3. Calling a program starts an executable program on the server.

4. An event is triggered, and data obtained, while the workflow is in progress, by the receipt of an XML message on an internal JMS queue from a sending application.
5. Posting an XML message to an external JMS topic or queue sends notification and data to an external application that subscribes to the topic or receives messages from the queue.
6. Performing a plug-in action invokes custom Java code written to integrate other applications or systems.

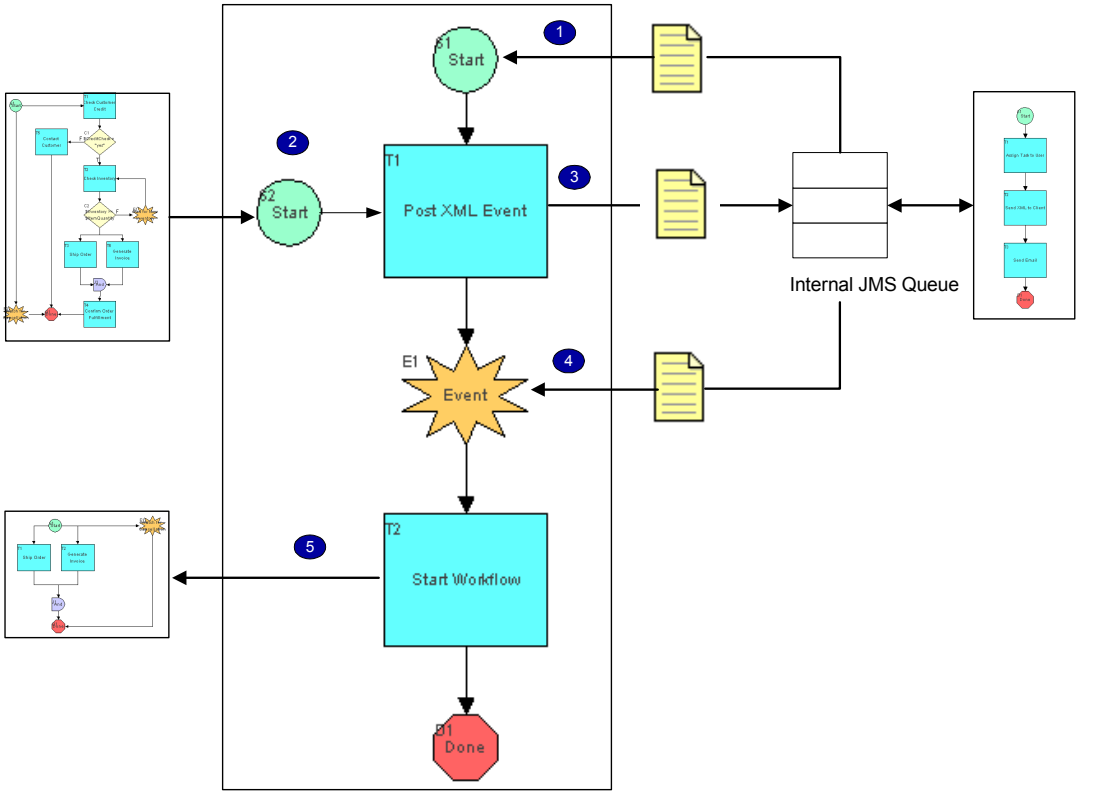
Integrating Workflows

Workflows can communicate with each other in the following ways:

- By calling each other directly
- Via XML/JMS messaging

The following figure shows the various interactions between workflows. Each scenario is described below the figure.

Figure 1-5 Integrating Workflows



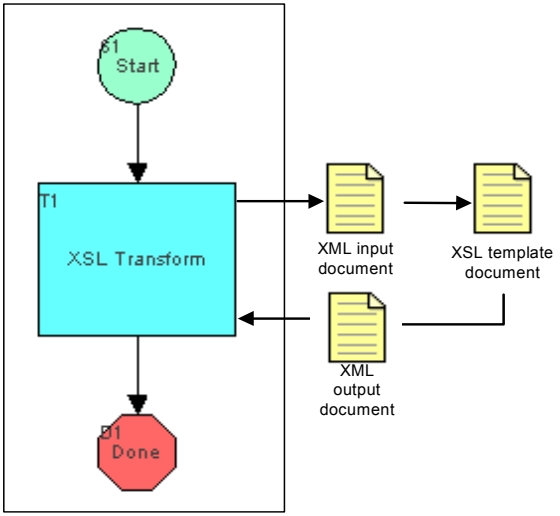
1. The workflow is triggered, and data obtained, by the receipt of an XML message on an internal JMS queue from another, sending workflow.
2. The workflow is set up with a called start, and is initiated directly by another workflow, which passes parameters to and from it directly.
3. Posting an XML message to an internal JMS queue sends notification and data to another workflow, triggering its start, or an event within it. (same as 1)
4. An event is triggered, and data obtained, while the workflow is in progress, by the receipt of an XML message on an internal JMS queue from another, sending workflow. (same as 1)

5. Starting a workflow initiates a called workflow and passes parameters to and from it directly. (same as 2)

Integrating Data

In addition to exchanging XML messages, workflows can transform XML documents from one format to another, to be passed out to external applications.

Figure 1-6 Integrating Data

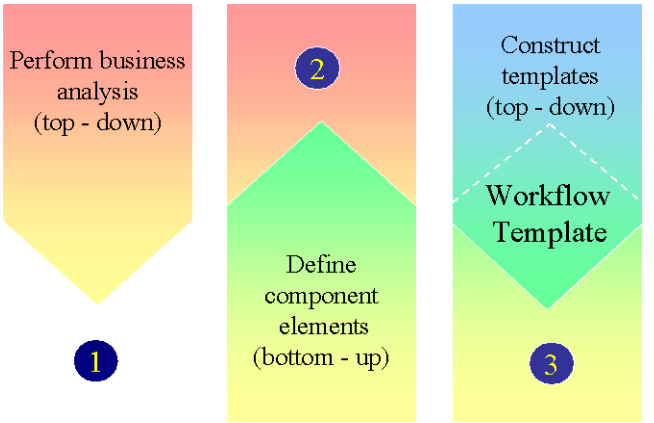


In this scenario, the workflow uses an Extensible Stylesheet Language (XSL) template to transform the structure of one XML document into another.

Workflow Design Approaches and Tasks

Theories of system design suggest that, ideally, systems should be designed using a top-down approach. In reality, systems are designed from both the top down and the bottom up. Both approaches are also used for designing WebLogic Integration workflows, as the following figure shows.

Figure 1-7 Designing Workflows



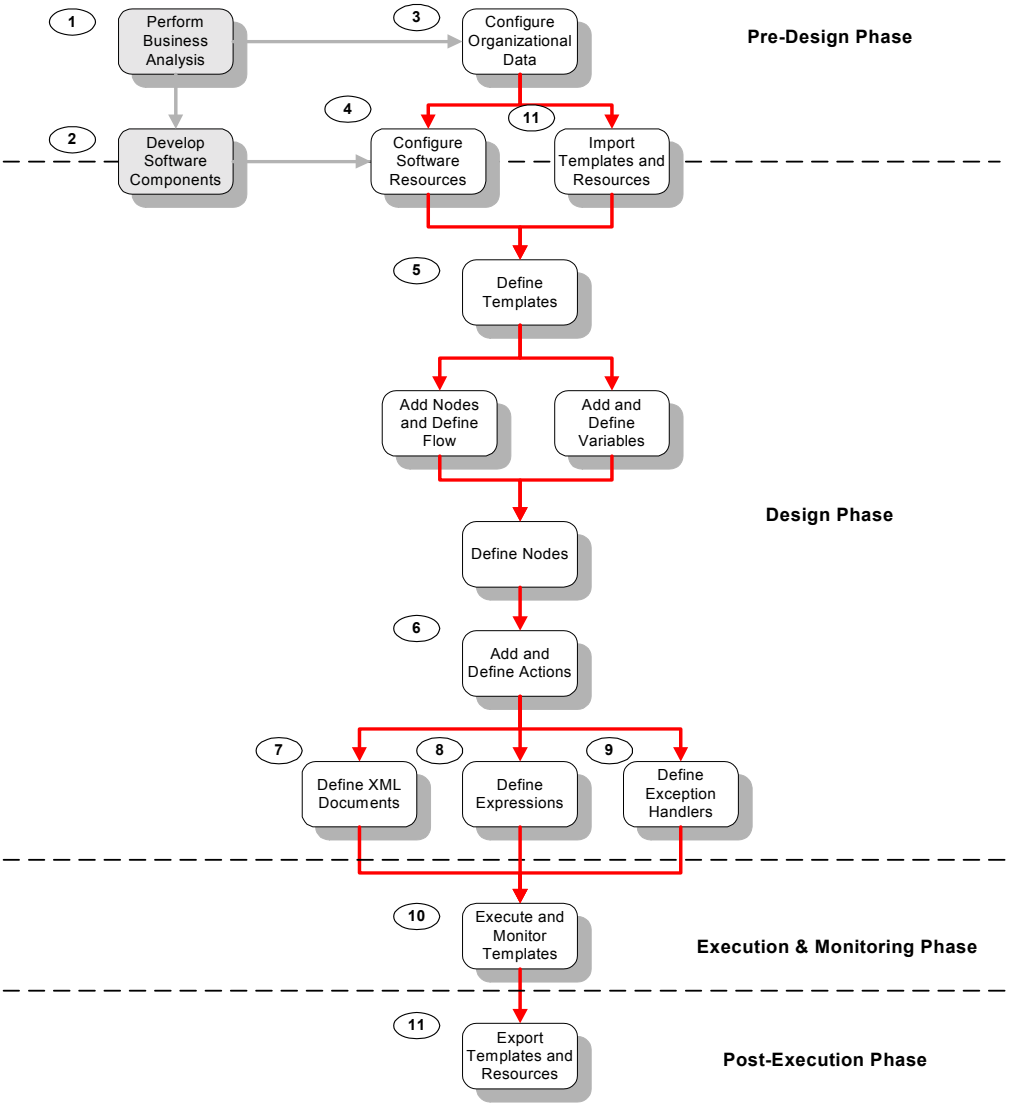
The top-down approach is the one that has been adopted as the organizing principle of this document. However, both top-down and bottom-down approaches are not only valid, but necessary, and you are likely to adopt both in practice. The application of both approaches to specific Studio design tasks and phases is discussed in the following sections.

Top-Down Approach

The Studio graphical user interface favors a top-down approach in which most external component elements have already been developed. In this approach, the workflow definition process involves moving from mapping out a high-level graphical representation of the basic activities and logic that the application fulfills, to drilling down to deeper levels of detailed specifications.

This work model is illustrated in the following flowchart, which summarizes the main tasks you perform when modeling, designing, defining, and testing WebLogic Integration workflows in the Studio. In fact, this is the model that is followed by the structure in this document, and each numbered step, starting with step 3, corresponds to a topic in this document. The figure is explained in detail below. Note that the iterative, circular nature of the design process is not actually represented by loops in the flowchart, for the sake of graphic simplicity.

Figure 1-8 Studio Design Work Model: Top-Down Approach



1. The business analysis phase includes identifying your application and data requirements, taking stock of existing components and applications you want to integrate, defining a data model that captures the business rules and structure of

your organization, and perhaps even beginning to model a workflow in a third-party design tool. For more information on this phase, see *Designing BEA WebLogic Integration Solutions*.

2. The resource development phase includes developing Java components such as EJBs and custom Java classes, XML documents and style sheets to be used in outgoing messages from workflows or incoming messages from connecting applications or components, and plug-in components. For more information on this phase, see *Designing BEA WebLogic Integration Solutions*.

3. In the pre-design phase, you use the Studio to configure business data. The tasks in this phase are discussed in Chapter 3, "Administering Data." It is assumed that you will only need to perform these tasks once initially, and occasionally when your business rules or personnel change.

4. At this point, you can begin to set up any external resources you have created in step 2, to make them globally accessible to your workflows. The tasks in this phase are discussed in Chapter 4, "Configuring Workflow Resources." You will likely perform these steps both before and during the design phase, in an iterative fashion, as the particular requirements of your workflows become clearer.

5. You begin the actual design phase by setting up workflow templates and template definitions. You then define a high-level process flow, by adding and connecting nodes to the workflow diagram, creating variables, and defining node properties. These tasks are described in "Defining Workflow Templates," and will require some use of workflow expressions, which are discussed in step 8.

6. Once you have created nodes, you begin to add and define actions, as described in "Defining Actions." At this stage, you probably need to further configure resources as described in step 4, and flows, node definitions, and variables, as in step 5. Defining actions also involves the tasks described in steps 7, 8, and 9.

7. While defining actions, you may need to import XML documents you have created and configured in steps 2 and 3, or you may need to compose the XML content from within workflow actions. These tasks are described in Chapter 7, "Working with XML Entities."

8. Throughout the workflow design process, you need to provide data formatted in the workflow expression language, which can contain literals, constants, variables, and built-in functions that supply run-time data. The semantics and syntax of workflow expressions are discussed in Chapter 8, "Using Workflow Expressions."

9. To add custom exception handlers to the template definition, you define sub-flows of actions to be performed upon the occurrence of run-time exceptions and a method of exiting the exception handler and returning to the main program flow. These tasks are described in Chapter 9, "Handling Workflow Exceptions." Once you have defined exception handlers, you once again need to add actions to nodes to reference them, as described in step 6.
10. At this point, you can run the workflow and test it. The Studio offers several monitoring features that allow you to view running workflow instances to help identify design errors, and that are described in Chapter 10, "Monitoring Workflows." Once you identify potential design bugs, you need to repeat the steps in the design process described so far; for example, to re-place actions, re-formulate expressions, and so on.
11. In the post-execution phase, when your workflow has been thoroughly tested and is running correctly, you can export workflows and any resources you have created and configured to Java archive packages for reuse. You can then re-import exported packages to begin the entire process cycle again. Import and export tasks are described in Chapter 11, "Importing and Exporting Workflow Packages."

Bottom-Up Approach

The bottom-up approach to design tasks in the Studio involves creating a catalog of reusable, exportable design patterns consisting of variables, actions, nodes, and groups of nodes that can be copied or imported into template definitions and incorporated into flows. This approach still requires that you have undertaken the preliminary tasks of analyzing your application needs and developing required resources. However, rather than beginning by outlining a high-level program flow, you begin by mapping the available Studio actions to the common operations that your workflow(s) need to accomplish, and defining the details of those actions. A suggested work model that follows this approach includes these steps:

1. Define any global entities that need to be referenced by actions. This includes business calendars (see "Administering Business Calendars" on page 3-4) and business operations (see "Configuring Business Operations" on page 4-7).
2. Create or import a workflow template and template definition. These tasks are described in "Working with Templates" on page 5-3 and "Working with Template Definitions" on page 5-7.

3. Add and define node *archetypes* that perform common activities for your workflows. A description of these types, and the actions that comprise them is provided in "Understanding Actions" on page 6-2.
4. Identify the workflow functions and other expression components that you may need to use to take advantage of features such as addressed messaging, event-triggered processing, communication between workflows, and so on. A description of functions is provided in "Using Functions" on page 8-6 in "Using Workflow Expressions."
5. Define common variables that actions need to reference, such as instance variables for business operations, XML variables to contain XML documents, input and output variables for called workflows, and so on. Variable definition tasks are described in "Working with Variables" on page 5-28.
6. Define all the low-level details for the actions contained in these node types, such as expressions, parameters for business operations, JMS messaging options for actions that post XML events, XML document content for actions that embed XML documents, and so on. For actions that reference users, roles or organizations, you can even use the default users as placeholders, until you have defined the entities that represent your business requirements.
7. Define specific node properties for Start, Event, and Task nodes. These tasks are discussed in "Defining Node Properties" on page 5-33.
8. Set up exception handlers and their actions.
9. Copy the components you have created into new workflow templates (described in "Copying Nodes" on page 5-25), or export the template definitions to Java archive files for re-import into new templates.
10. Arrange the predefined nodes into a flow and define high-level organizational data relevant for the new template definitions.

Studio Tools

In addition to the graphical drawing engine you use to design process flows, the Studio also provides the following tools to help you manage external resources and define workflow properties:

- XML Finder

  The XML Finder is a file and content management tool that allows you to access and organize different types of XML content, including Schemas, document type definitions, message formats, and so on, that may be stored in a database table or in files. The XML Finder is described in "Managing Entities in the Repository" on page 4-23 and "Using the XML Finder to Retrieve and Export XML Entities" on page 7-17.

- XML editor

  Many Studio actions allow you to compose, import and export XML documents from within workflows. These actions contain an XML editor that you can use to compose XML document templates from scratch, edit existing documents, and validate content according to XML Schemas. XML editing features are described in Chapter 7, "Working with XML Entities."

- Expression Builder

  Many Studio dialog boxes require that you enter data in workflow expression syntax. The Expression Builder is an editing tool that allows you to select from a catalogue of operators, literals, workflow functions, and variables you have created, to construct expressions component by component. When you have finished building an expression, the Expression Builder validates the syntax and informs you of any errors. The Expression Builder is described in "Using the Expression Builder" on page 8-28.

- XPath Wizard

  To extract content from incoming XML documents, you use XPath functions to locate the target data in XML elements and attributes. The XPath Wizard is a point-and-click tool that allows you to automatically generate XPath expressions from selected content in sample XML documents or Schema or DTD-generated XML, without the need to master the syntax of the XPath language. The XPath Wizard is described in "Creating XPath Expressions Using the XPath Wizard" on page 8-31.

- Import/Export wizard

  The Import/Export wizard allows you to export all workflow templates, template definitions and associated resources, such as business operations, event keys, plug-ins, and XML entities, that have been defined in the system. Workflow objects are exported to a Java archive file and can be re-imported to any system running the Studio. The Import/Export wizard is discussed in Chapter 11, "Importing and Exporting Workflow Packages."

CHAPTER

2 Using the Studio Interface

This section explains how to start the Studio, and provides an overview of the Studio graphical user interface:

- Starting and Logging On to the Studio
- Overview of the Studio Interface
- Using Interface View
- Exiting the Studio

Starting and Logging On to the Studio

To start the Studio, do one of the following:

- On a Windows system, choose Start→Programs→BEA WebLogic Platform 7.0→WebLogic Integration 7.0→Studio.
- On a UNIX system, do to the *WLI_HOME*/bin directory and run the `studio` command.

  For example, if WebLogic Integration is installed in the `/home/bea/weblogic700/integration` directory, enter the following:

  ```
  cd /home/bea/weblogic700/integration/bin
  . ./studio
  ```

The Logon to WebLogic Integration dialog box is displayed.

Figure 2-1 Logon to WebLogic Integration Dialog Box

To log on to WebLogic Integration:

1. Enter your user name and password in the appropriate fields. If you have not yet been assigned a user name and password for the Studio, enter a default user name and password. For a list of default user names and passwords, see "WebLogic Integration Users and Passwords" in "Getting Started" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

   Note: User names and passwords are case-sensitive. Be sure to enter the user names and passwords in lower case.

2. In the Server [:port] field, specify the system that is running the WebLogic Integration server as follows:

   `t3://host:port`

   - *host* is the computer name or IP address of the system that is running the WebLogic Integration server. Specify `localhost` if the server is running on the same computer as the Studio application.
   - *port* is the number you specified for the listen port when the WebLogic Integration server was installed. The default is 7001.

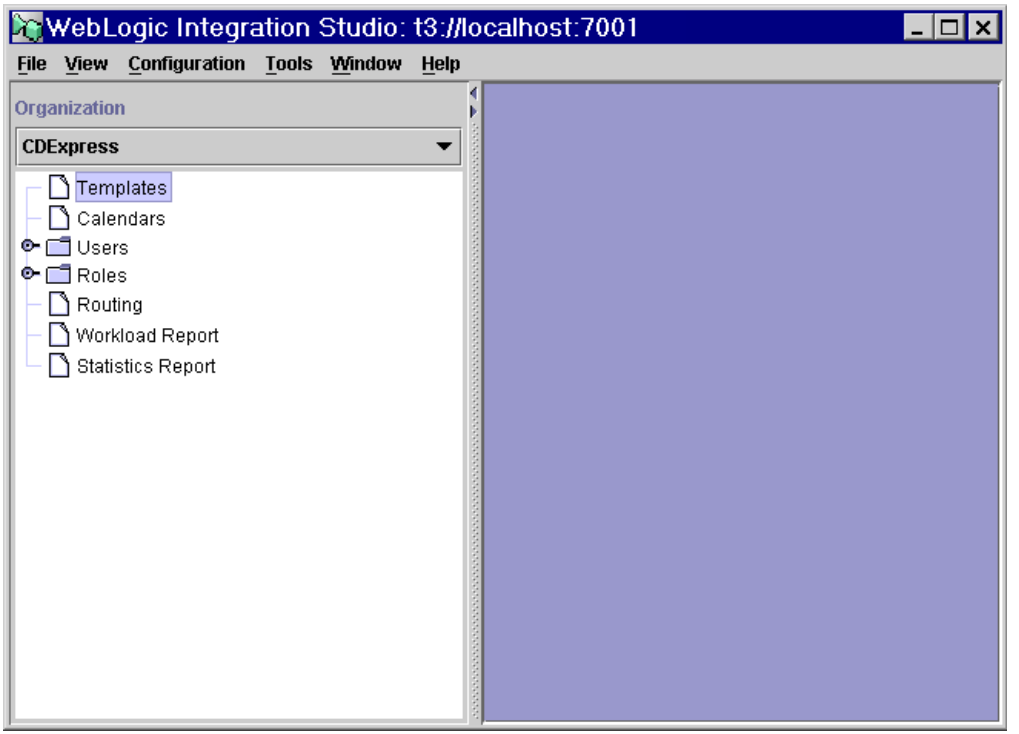
   To log in to clustered servers, enter the following in the Server[:port] field:

`t3://host1,host2,host3:port`

In this case, *host1*, *host2*, and *host3* are the computer names or IP addresses of the clustered WebLogic Integration servers.

3. Click OK to display the main window of the Studio.

Figure 2-2 WebLogic Integration Studio Main Window



Overview of the Studio Interface

This section describes the parts of the Studio user interface, and the functions they provide.

Menu Options

The following sections provide information about using the menu bar in the Studio.

File Menu

Choose File→*Option* to perform the functions described in the following table.

Menu Option	Function
Logon	Log on to the WebLogic Integration server.
Logoff	Log off from the WebLogic Integration server.
Exit	Log off from the WebLogic Integration server and exit the WebLogic Integration Studio.

View Menu

Choose View→*Option* to perform the functions described in the following table.

Menu Option	Function
Refresh	Update the information if changes are made to the database by other client applications.
Use color on flowcharts	View the workflow diagrams in color or black and white.
Sync selection with tree	Synchronize the workflow components in the folder tree hierarchy with the workflow components in the drawing area. For example, clicking the Start node in a workflow diagram opens the corresponding Start node folder in the folder tree hierarchy.
Look and feel	Change the look and feel of the Studio display. The choices are: ■ Metal ■ CDE/Motif ■ Windows

Menu Option	Function
Interface View . . .	Display additional visual objects to represent business operations, sub-workflows, XML documents, and plug-ins. Selecting this menu item displays the Interface View Preferences dialog box. For details about setting interface view preferences, see "Using Interface View" on page 2-12. If a start, event, or done node contains a customized property defined by a plug-in, a small plug-in icon is displayed in the upper-right corner of the node icon.

Configuration Menu

Choose Configuration→*Option* to perform the functions described in the following table.

Menu Option	Function
Organizations	Define organizations that represent different business entities, geographical locations, or any other distinction relevant to the particular business of the company. For details about defining organizations, see "Maintaining Organizations" on page 3-11.
Business Operations	Define business operations representing a method call on an EJB or Java Class instance. For details about defining business operations, see "Configuring Business Operations" on page 4-7.
Events	Define event key expressions. For instructions about defining event key expressions, see "Configuring Event Keys" on page 4-18.
Plugins	View and configure the available plug-ins. For details about configuring plug-ins, see "Configuring Plug-Ins" on page 4-3.
Permissions	Define the user and role permission levels. For details about defining permission levels, see "Setting Permissions for Users" on page 3-28 and "Setting Permissions for Roles" on page 3-27.
Role Mappings	Map the currently-defined roles to WebLogic Server groups. For details about mapping roles, see "Changing the Mapping for Roles" on page 3-24.

Tools Menu

Choose Tools→*Option* to perform the functions described in the following table.

Menu Option	Function
Export Package . . .	Export workflow objects as a JAR file. For details about exporting packages, see Chapter 11, "Importing and Exporting Workflow Packages."
Import Package . . .	Import workflow objects from a JAR file. For details about importing packages, see Chapter 11, "Importing and Exporting Workflow Packages."
Show XML Finder . . .	Display the XML finder dialog box to manage the XML repository. For details about managing the XML repository using the XML finder, see "Managing Entities in the Repository" on page 4-23.

Help Menu

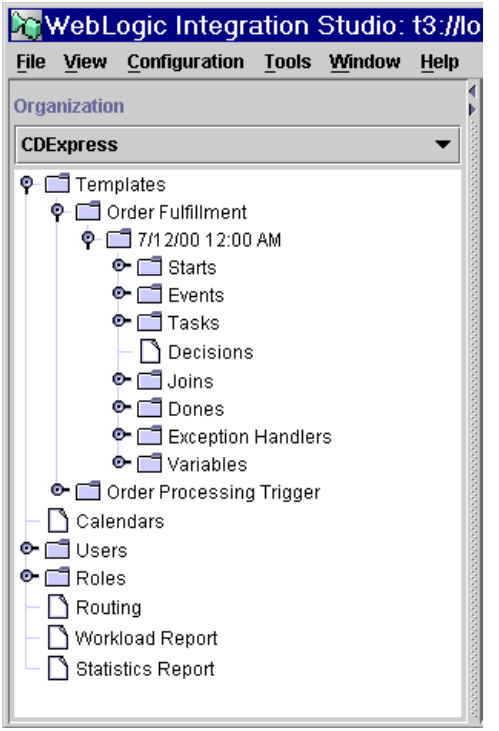
Choose Help→*Option* to perform the functions described in the following table.

Menu Option	Function
Help Topics	Access the online help for the Studio.
Plugin Help	Access the online help for the loaded plug-ins.
About WebLogic Integration Studio . . .	Provide software version information about the WebLogic Integration Studio.

Folder Tree Display

The WebLogic Integration Studio interface contains a folder tree display, which organizes workflow components in a standard tree structure.

Figure 2-3 WebLogic Integration Folder Tree Display



To select a different organization, use the Organization drop-down list at the top of the window.

To view details about any of the following items shown in the folder tree display, double-click the item to expand it:

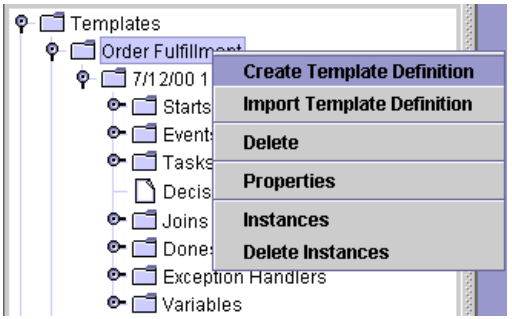
- Templates
- Calendars
- Users
- Roles
- Routing
- Workload Report

- Statistics Reports

For example, double-click the Templates folder to display a list of workflow templates. Double-click a workflow template to display a list of all workflow template definitions. Expanding a particular workflow template definition displays folders containing the Tasks, Decisions, Events, Joins, Starts, Dones, and Variables for that workflow template definition.

You can right-click most items in the folder tree to display a menu containing options relevant to the highlighted item. For example, right-click on an existing workflow template to display a menu with the options shown in the following figure.

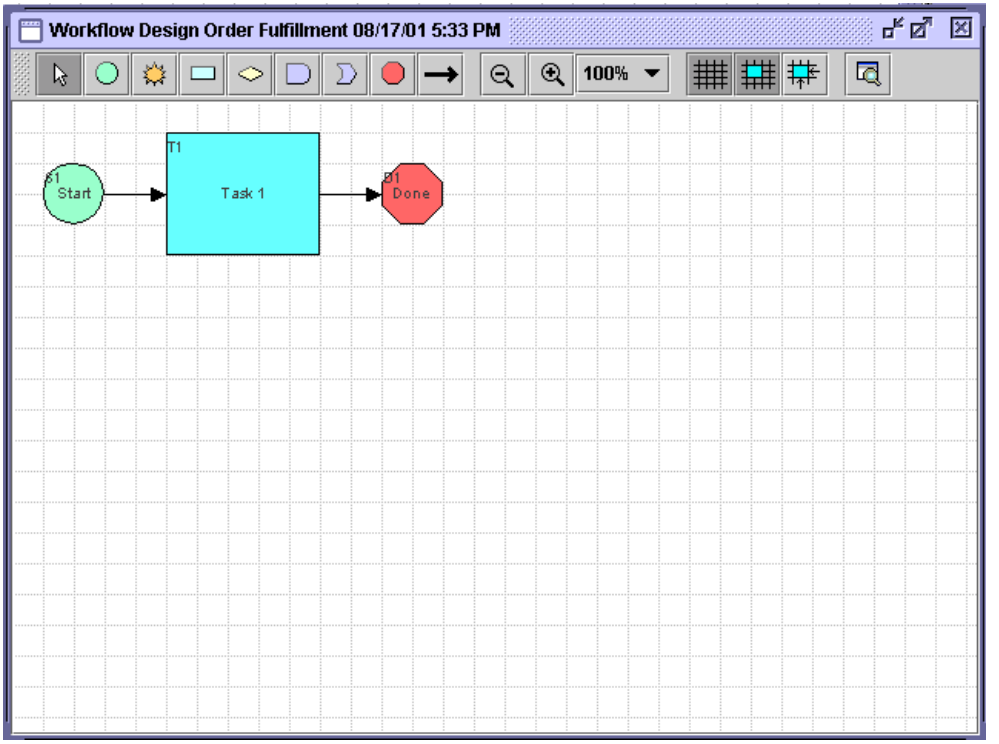
Figure 2-4 Workflow Template Menu Options



Workflow Design Area and Toolbar

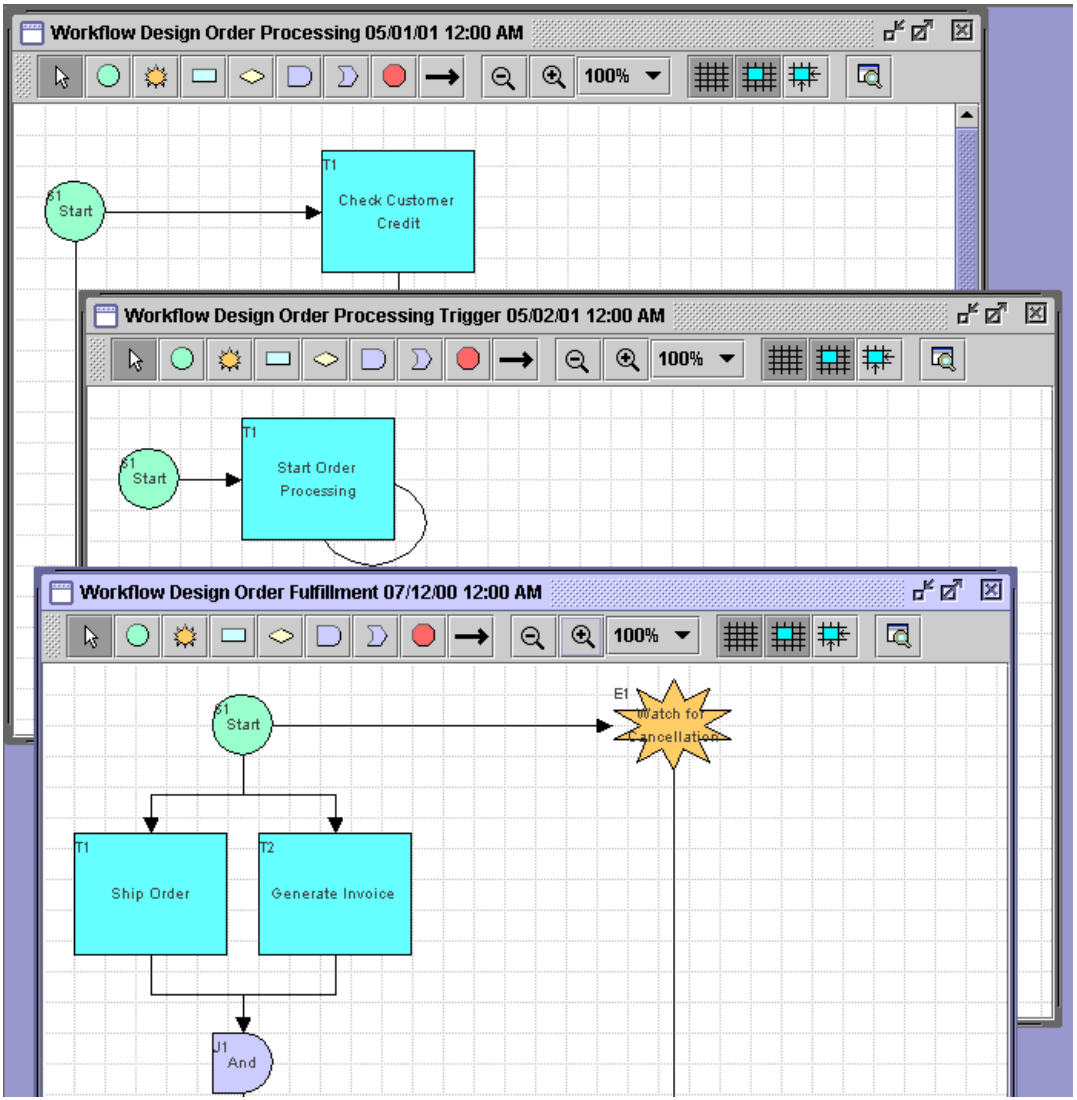
Once you have created or opened an existing workflow template definition (for procedures, see “Working with Template Definitions” on page 5-7), the workflow design area, shown in the following figure, is where you create your workflow designs. The workflow toolbar contains the shapes representing workflow nodes and the connections you use to define a workflow.

Figure 2-5 Workflow Design Window



You can have multiple workflow designs open simultaneously. You can view more than one workflow, and toggle back and forth between them.

Figure 2-6 Multiple Workflow Diagrams



Using the Toolbar

The toolbar also contains the following toggle controls:

Table 2-1 Toolbar Buttons

Toolbar button	Description
	Zoom out of the diagram. This is helpful if you are working with a complex diagram and need to view the overall flow.
	Zoom in to the diagram.
	Display or hide a grid in the diagram.
	Align shapes in the diagram.
	Set auto-alignment of shapes on or off.
	Toggle Interface View on or off. For more information about Interface View, see “Using Interface View” on page 2-12.

By default, the workflow toolbar appears at the top of the drawing area. Depending on the size of the Studio window or drawing area, you might not be able to see the entire toolbar. To see the entire toolbar, relocate the toolbar to another part of the Studio drawing area, or to another location on your desktop.

To relocate the toolbar, click the background area of the toolbar (for example, the space between the Draw Connection and Zoom Out button), and drag the toolbar to the desired location.

If you drag the toolbar to another location on your desktop outside of the Studio drawing area, the toolbar becomes a separate window on your desktop, which you can minimize, maximize, and close.

Using Interface View

Once you have created a template definition, begun to define nodes and add actions to them, it is often useful to see the objects to which a workflow interfaces, according to the nodes that reference those objects. You can use the Interface View to display the components in the workflow diagram, plus icons representing the following objects to which the workflow can interface:

- Sub-workflows — other workflows to which the current workflow connects
- Business operations — software components, such as EJBs and Java classes, that the workflow invokes to perform a function
- Inbound and outbound XML documents — XML documents that are coming into the workflow, or that the workflow is sending to another workflow or application.
- Plug-ins — custom workflow components

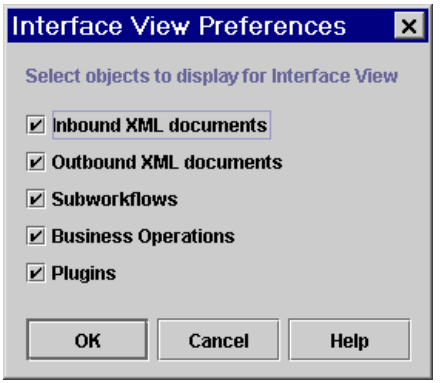
Interface View is disabled by default. To use Interface View, you must turn it on for each template definition, and for each new session in which you access the same template definition.

By contrast, Interface View preferences, which determine the objects that are visible, are set system-wide; that is, they apply to all template definitions. They are saved from session to session in the Studio; they are not saved as part of the template definition.

To specify the objects you want to view:

1. Choose View—Interface View to display the Interface View Preferences dialog box.

Figure 2-7 Interface View Preferences Dialog Box



2. Select the check boxes for the objects you want to display, or clear the check boxes for the objects you do not want to display.

To toggle between a normal and interface view for the current template definition, click the following button in the toolbar: 

Viewing Inbound XML Document Data

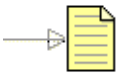
In an interface view, the following icon represents an inbound XML document in the workflow diagram.

Note: The arrow points toward a template node to indicate inbound direction.

An inbound XML document icon is displayed for Start and Event nodes that respond to incoming XML documents. Hover the mouse pointer over the XML document icon in the workflow diagram to display a text box containing the type (inbound or outbound), root, and key of the XML document.

Viewing Outbound XML Document Data

In an interface view, the following icon represents an outbound XML document in the workflow diagram.



Note: The arrow points away from a template node to indicate outbound direction.

An outbound XML document icon is displayed if either a Send XML to Client or Post XML Event action is defined for an object. You can obtain information about the XML document as follows:

- Hover the mouse pointer over the XML document icon in the workflow diagram to display a text box containing the type (inbound or outbound), root, and key of the XML document.
- Double-click the XML document icon, or right-click the icon and select Properties from the pop-up menu, to display the dialog box (read-only) for the defined action, either Send XML to Client or Post XML Event.

For details about the Send XML to Client and Post XML Event actions, see "Sending an XML Message to a Client Application" on page 6-58 and "Posting an XML Message to a JMS Topic or Queue" on page 6-82.

Viewing Sub-workflow Data

In an interface view, the following icon represents a sub-workflow in the workflow diagram.

A sub-workflow icon is displayed if a Start Workflow action is defined for an object. You can obtain information about the sub-workflow as follows:

- Hover the mouse pointer over the sub-workflow icon to display a text box containing the name of the called sub-workflow.
- Double-click the sub-workflow icon, or right-click the icon, and select Properties from the pop-up menu, to display the Start Workflow dialog box (read-only).

For details about the Start Workflow action, see "Calling a Sub-Workflow" on page 6-36.

Viewing Business Operation Data

In an interface view, the following icon represents a business operation in the workflow diagram.

A business operation icon is displayed if a Perform Business Operation action is defined for an object. You can obtain information about the business operation as follows:

- Hover the mouse pointer over the business operation icon in the workflow diagram to display a text box containing the name of the business operation to be performed.
- Double-click the business operation icon, or right-click the icon and select Properties from the pop-up menu, to display the Perform Business Operation dialog box (read-only).

For details about the Perform Business Operation action, see "Calling a Business Operation" on page 6-78.

Viewing Plug-In Data

In an interface view, the following icon, or a custom icon, represents a plug-in action in the workflow diagram.

If a node contains a plug-in action, you can obtain information about the action as follows:

- Hover the mouse pointer over the plug-in icon in the workflow diagram to display a text box containing a description of the plug-in action.
- Double-click the plug-in icon, or right-click the icon, and select Properties from the pop-up menu, to display the corresponding plug-in action dialog box (read-only).

Note: If a Start, Event, or Done node contains a customized property defined by a plug-in, a small plug-in icon is displayed in the upper-right corner of the node icon.

Exiting the Studio

To log off WebLogic Integration and keep the Studio displayed:

1. If you want to save changes to a workflow, right-click the workflow template definition and choose Save from the menu that is displayed.
2. Choose File→Logoff.

To exit the Studio:

1. If you want to save changes to a workflow, right-click the workflow template definition and choose Save from the menu that is displayed.
2. Choose File→Exit, or select the System Close box. A dialog box asks you to confirm your decision. Click Yes to exit.

Note: If you try to exit the Studio without saving your changes, a dialog box prompts you to do so.

3 Administering Data

This section describes the following data administration concepts and tasks within the Studio:

- Overview of Data Configuration Tasks
- About Security Realms
- Administering Business Calendars
- Maintaining Organizations
- Maintaining Users
- Maintaining Roles
- Assigning Permissions to Users and Roles
- Administering Task Routings

Overview of Data Configuration Tasks

Data configuration tasks in the Studio include defining business calendars, creating organization, users, and roles, configuring security and permissions, and defining task routings. The following steps outline the recommended order in which you should perform these tasks when modelling organizational data and configuring the system initially:

1. Create calendars that may be associated with organizations, users, and roles. Procedures are given in “Creating a Calendar” on page 3-5. Alternatively, import previously exported calendars from existing workflow packages; see procedures in “Importing Workflow Packages” on page 11-5.
2. Add an organization or organizations. Procedures are given in “Adding an Organization” on page 3-12.
3. Add users to the system. Procedures are given in “Creating a User” on page 3-15.
4. Add users to organizations. Procedures are given in “Adding a User to an Organization” on page 3-16.
5. Optionally, use the WebLogic Server Administration Console to create a group which will correspond to a role you define in the Studio. For more information about roles, see “Maintaining Roles” on page 3-20. For information about creating WebLogic Integration groups, see "Defining Groups in the Compatibility Realm" in "Using Compatibility Security" in *Managing WebLogic Security*, in the BEA WebLogic Server documentat set at the following URL:

   `http://edocs.bea.com/wls/docs70/secmanage/security6.html`
6. Define a role or roles within an organization, map the role to a WebLogic Server group, and associate member users with the role. Procedures are given in “Creating a Role” on page 3-21.
7. If necessary, modify the levels of permission for the role. Changing the levels of permission for a role also changes the permissions in the group, and affects any other roles that are mapped to the group. Procedures are given in “Setting Permissions for Roles” on page 3-27.
8. Define additional levels of permission for users. Users inherit the permissions of the role to which they belong, but you can add other levels of permission to users that are not defined for the role to which the users belong. Procedures are given in “Setting Permissions for Users” on page 3-28.

About Security Realms

WebLogic Server provides security for BPM applications through a service called a security realm. A security realm is a logical grouping of users and groups. A user is a specific individual who performs a certain task, such as programming or sales. A group is a collection of users who perform the same task. In this scheme, Group A might represent a collection of users who are programmers, and Group B might represent a collection of users who are sales people. Within a security realm, administrators can specify the levels of access users and groups have to workflows and other resources.

Note: For important background information about WebLogic Integration and BPM security, see the following documents:

- "Using WebLogic Integration Security" in *Deploying BEA WebLogic Integration*
- "Understanding the BPM Security Model" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*
- *Introducing WebLogic Platform Security*

The third document in this list provides important information if you are planning to use BPM applications with other WebLogic Platform components.

WebLogic Integration maintains information about roles and users in WebLogic Server security realms. When you define users and roles in WebLogic Integration, you need to specify their relationship to users and groups in WebLogic Server. You do this by mapping the roles you define in the Studio to the groups in a security realm in WebLogic Server.

A security realm in WebLogic Server can be manageable or non-manageable. This is defined when the realm is implemented on WebLogic Server. A manageable realm is one in which an application can make updates to groups and users. A non-manageable realm is one in which an application can only list the groups and users.

WebLogic Integration detects automatically whether a realm is manageable or non-manageable. The type of security realm WebLogic Integration detects determines which data administration functions are available in the Studio.

You can perform the following tasks using the Studio dialog boxes only if WebLogic Integration detects a manageable realm:

- Map a role to a WebLogic Server group
- Add or delete a user
- Add a user to an organization
- Remove a user from an organization
- Add a user to a role
- Remove a user from a role

If WebLogic Integration detects a non-manageable realm, you can perform the following tasks using the Studio dialog boxes:

- See a list of users and roles. The functions provided in the Studio dialog boxes to perform add, delete, and mapping tasks are dimmed, so you cannot select them.
- Update data managed by WebLogic Integration, and not by WebLogic Server in a security realm, such as a business calendar or organization.

If you want to add users to a non-manageable realm, you must add them using the appropriate data administration functions for the realm in the WebLogic Server Administration Console. You cannot add them using the Studio dialog boxes.

Administering Business Calendars

The business calendar feature defines operating hours for entities represented in workflows. Business calendars make possible business time-related calculations, such as "Set a task's due date to three business days from today." You should define business calendars that exclude non-operating days or hours such as weekends or statutory holidays; organizations that do not use calendars use a 365-day calendar.

Calendars can be associated with the following entities, as described in the sections indicated:

- Organizations (see "Maintaining Organizations" on page 3-11)

- Users (see "Maintaining Users" on page 3-14)
- Roles (see "Maintaining Roles" on page 3-20)

You can define and assign the same business calendar to organizations, users, roles, and actions. You can also assign different business calendars to users, roles, and actions within the same organization.

Calendars are also used within workflows by timed Start nodes (see "Defining a Timed Start Node" on page 5-36), timed events (see "Embedding a Timed Sequence" on page 6-32), and due dates for user-assigned tasks (see "Setting a Task Due Date" on page 6-51).

Note: To administer business calendars, you must have Configure System permission. See "Assigning Permissions to Users and Roles" on page 3-26 for more information.

Calendar assignment is hierarchical in nature. The hierarchy places time-related actions at the lowest level, followed by roles and users, and finally, organizations at the highest level. If a time-related action is not assigned a calendar, it will, by default, be assigned the calendar of the user or role to which it is assigned. If a user or role is not assigned a calendar, it will, by default, be assigned the business calendar (organization level). In other words, calendar assignment is made at the most detailed component level.

Business calendars are rule-based. The calendar facility leads you through the definition of each rule.

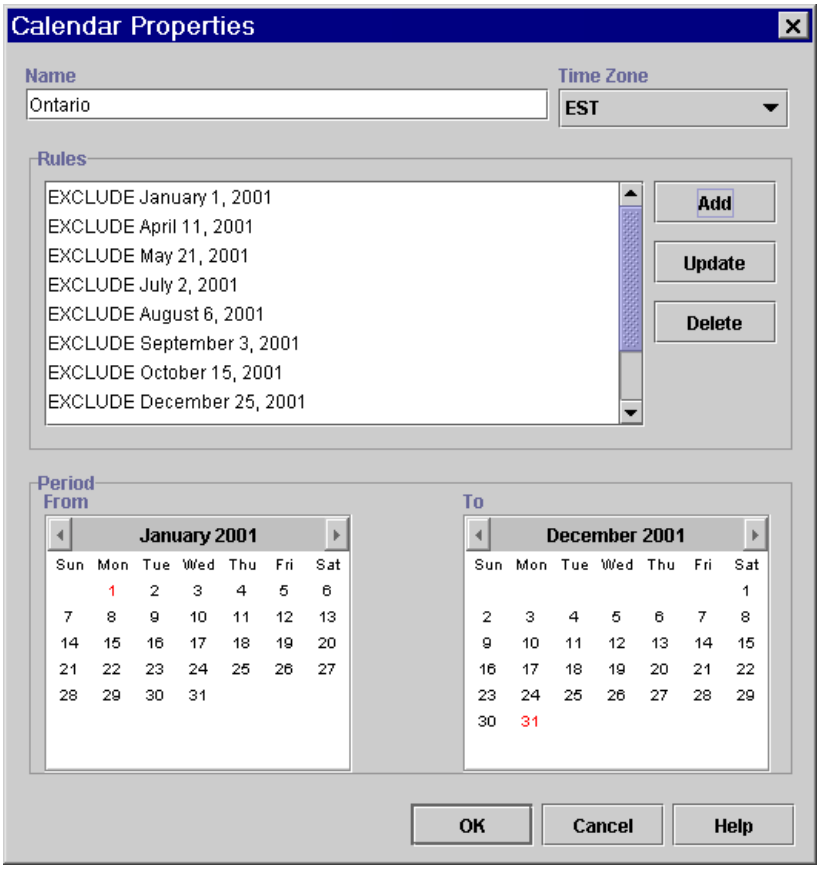
Creating a Calendar

Once you create a calendar, it is globally available for all organizations, users, and roles in the system.

To create a new calendar:

1. With any organization active, right-click Calendars in the folder tree, and choose Create Calendar to display the Calendar Properties dialog box.

Figure 3-1 Calendar Properties Dialog Box



2. Enter a meaningful name for the calendar in the Name field.
3. Select a time zone for the calendar from the Time Zone drop-down list.
4. In the Period boxes, select dates from the From and To boxes to specify the time period for the calendar. (The default is from January to December of the current year.)
5. To add a rule for the calendar, click Add. The Rule dialog box is displayed.

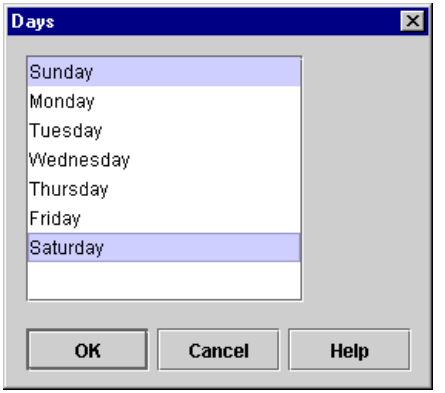
Figure 3-2 Rule Dialog Box

6. Select either Exclude or Include. These buttons determine the method by which you will define your calendar rules (by exclusion or by inclusion). It is recommended that you use one or the other method and not both throughout your rule defining process.

7. Click any of the following buttons to define a rule:

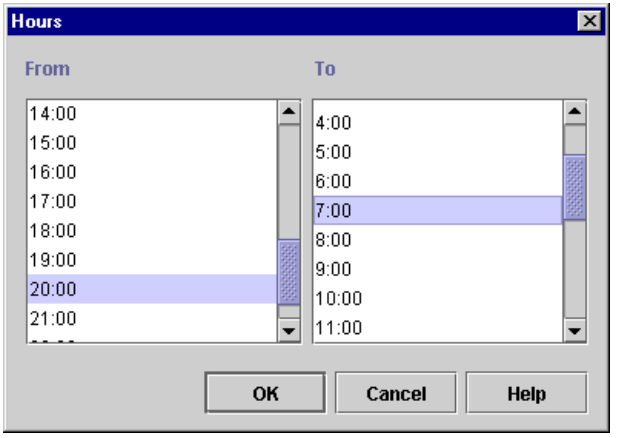
   - Days — displays the Days dialog box from which you can select a day or days to include or exclude. Hold the CTRL key down to select more than one day of the week. Click OK.

Figure 3-3 Days Dialog Box



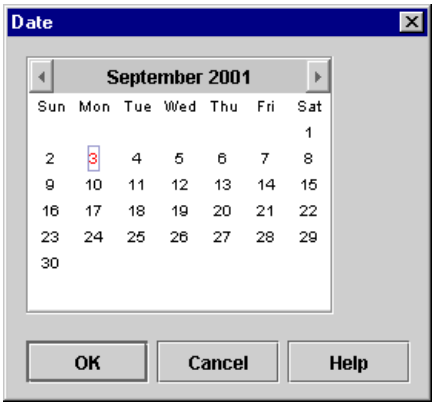
- Hours — displays the Hours dialog box, from which you can select a range of time to include or exclude. Select an hour in the From list and an hour from the To list, and click OK.

Figure 3-4 Hours Dialog Box



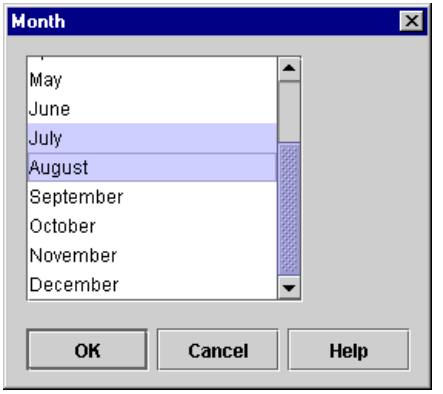
- Date — displays the Date dialog box, from which you can select a particular date. Select a month and date to exclude or include, and click OK.

Figure 3-5 Date Dialog Box



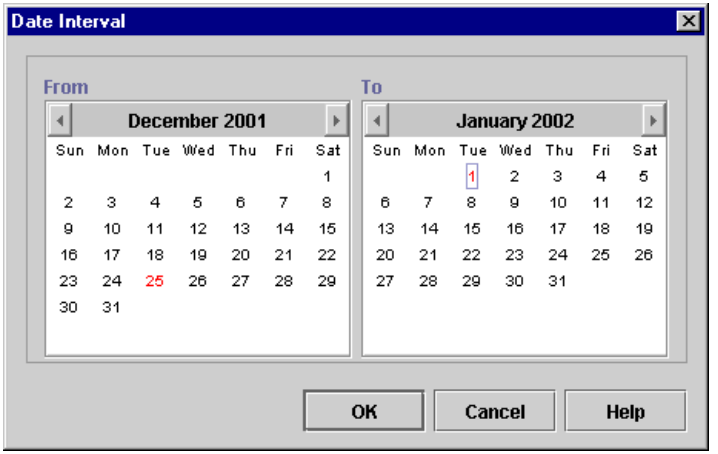
- Month — displays the Month dialog box, from which you can select a month or months to exclude or include certain month(s) of the year. (Hold the CTRL key down while clicking with your mouse to select more than one month of the year from the Month dialog box.)

Figure 3-6 Month Dialog Box



- Date Interval — displays the Date Interval dialog box, from which you can select a range of dates to be excluded or included. Select a month and date from the From box and a month and date from the To box, and click OK.

Figure 3-7 Date Interval Dialog Box



8. Click OK in the Rule dialog box to add the rule. The new rule is displayed in the Rules area of the Calendar Rules dialog box.
9. Continue to add rules to the calendar by repeating steps 7 and 8.
10. Click OK to save the calendar.

Updating a Calendar

To update an existing calendar:

1. With any organization active, in the folder tree, expand the Calendars folder, right-click the calendar you want to update, and choose Properties.
2. In the Calendar Properties dialog box, make the necessary changes to the time zone and period for the calendar.

3. To update rules for the calendar, delete or update an existing rule by selecting the appropriate rule from the Rules list and clicking the Delete or Update buttons. To add a new rule, click Add, and follow the procedures in "Creating a Calendar" on page 3-5 to define the rule.
4. Click OK to save the changes to the calendar.

Deleting a Calendar

Warning: If you delete a calendar, you will not be warned about references to the calendar by other workflow objects. Be sure to update users, roles, and organizations that have been assigned the calendar, as well as any of the following workflow components that can reference calendars:

Timed Start nodes (see "Defining a Timed Start Node" on page 5-36)

Timed events (see "Embedding a Timed Sequence" on page 6-32)

Task due dates (see "Setting a Task Due Date" on page 6-51)

To delete an existing calendar:

1. Right-click the calendar you want to delete and select Delete from the pop-up menu.
2. When prompted with the Delete Calendar warning message, click Yes. To cancel the delete, click No.

Maintaining Organizations

You use the Organization facility to define organizations, which can represent different business entities, geographical locations, or any other distinction relevant to the particular business of the company.

Modelling units within your organization as different organizations allows you to re-use the same role names but map them to different groups. Thus, for example, you could create multiple roles called Supervisor, which would actually contain different

members according to the organization (for more information, see “Maintaining Roles” on page 3-20.) Note that organizations are specific to WebLogic Integration, and do not correspond to any groups on WebLogic Server.

You also assign users to one, or more, organizations. Users can only execute workflows within the organization to which they belong.

Note: To add, update, or delete organizations, you must have Administer User permission. For more information about permission levels, see “Assigning Permissions to Users and Roles” on page 3-26.

The Organization drop-down list located above the folder tree shows the currently active organization. When an organization is selected in this list, the folder tree displays roles, users, and workflows defined for that organization.

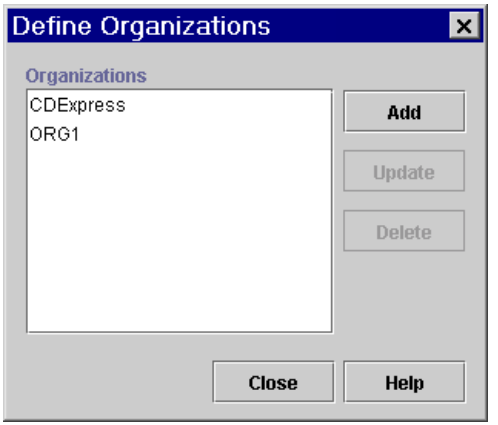
Adding an Organization

The Add Organization facility adds an organization to the WebLogic Integration database.

To add an organization:

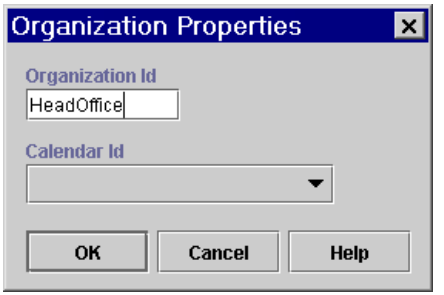
1. Choose Configuration→Organizations to display the Define Organizations dialog box.

Figure 3-8 Define Organizations Dialog Box



2. In the Define Organizations dialog box, click Add to display the Organization Properties dialog box.

Figure 3-9 Organization Properties Dialog Box



3. In the Organization ID field, enter a meaningful name for the organization.
4. If you have created a calendar, from the Calendar ID drop-down list, select a calendar to assign to the organization. For details about this feature, see "Administering Business Calendars" on page 3-4.
5. Click OK to create the organization.

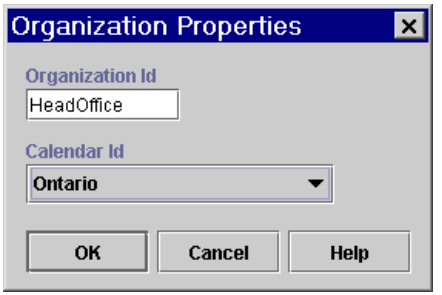
Updating an Organization

The Update Organization facility allows you to update the business calendar of an existing organization.

To update an organization:

1. Choose Configuration→Organizations to display the Define Organizations dialog box.
2. In the Define Organizations dialog box, highlight the organization to update.
3. Click Update to display the Organization Properties dialog box.

Figure 3-10 Organization Properties Dialog Box



4. Make changes as needed to the Organization Id or Calendar Id fields, and click OK.

Deleting an Organization

The Delete Organization facility allows you to remove an organization from the WebLogic Integration database, as long as that organization does not have any workflows defined for it. If the organization has workflows, you must first delete them. For procedures, see "Deleting a Template Definition" on page 5-18.

To delete an organization:

1. Choose Configuration→Organizations to display the Define Organizations dialog box.
2. In the Define Organizations dialog box, highlight the organization to delete.
3. When prompted by the Delete Organization warning message, click Yes to confirm or No to cancel.

Maintaining Users

A user is an individual who has permissions to perform certain tasks. Use the Users feature to create, update, and delete users in the current security realm in WebLogic Server. You can also add or remove users from organizations.

The User folder is located in the Studio folder tree. Expanding it displays a list of users who have already been defined for the current organization.

Note: To add, update or delete users, you must have Administer User permission. For more information about permission levels, see "Assigning Permissions to Users and Roles" on page 3-26.

Creating a User

The Create User function adds a user to the current WebLogic Server security realm, and is available only if the security realm is a manageable one.

To create a user in the security realm and WebLogic Integration database:

1. With any organization active, right-click the Users folder, and select Create Users to display the Create User dialog box.

Figure 3-11 Create User Dialog Box

2. Enter values in the following fields, and then click OK:

- User Id — uniquely defines the user. The user enters this ID to log in to the Studio and Worklist client applications.
- Password and Re-enter Password — enter a password for the user.
- E-mail Address — optionally enter the e-mail address of the user.
- Default Organization — select a default organization for the user. This organization is the one that is shown by default when the user logs on to the client applications. You can add the user to other organizations once the user has been created.
- Calendar — optionally select a business calendar for the user.

Note: The user ID and password you specify can contain alphanumeric characters from the JDK-supported character sets, including international characters.

Adding a User to an Organization

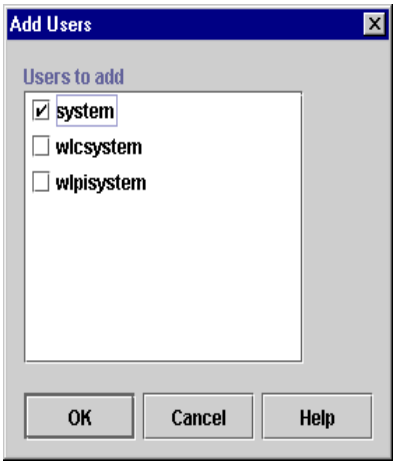
The Add User facility adds a user that is already defined in the WebLogic Server security realm to the current organization. This facility is accessible only if the WebLogic Integration Studio is operating within a manageable realm.

Adding a user to an organization allows the user to execute workflows within that organization at run time. Note, however, that it does not restrict users' access to workflow templates at design time; provided that a user has the necessary permission to open templates, he or she can access templates associated with multiple organizations.

To add a user to the current organization:

1. From the Organization field above the folder tree, select the Organization to which you would like to add the user.
2. In the folder tree, right-click the Users folder, and select Add Users to display the Add Users dialog box.

Figure 3-12 Add Users Dialog Box



3. Click the check box to the left of the user to add, and click OK.
4. The user is added to the folder tree.

Updating a User

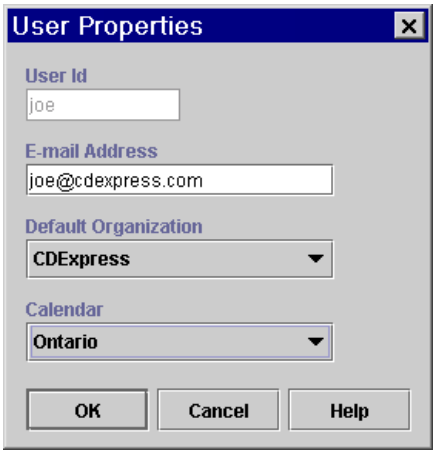
You can update a user by changing the user's ID, e-mail address, default organization, and business calendar.

Note: To change a password for a user once it has been created, you must use the WebLogic Server Administration Console. For more information, see "Updating Passwords" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

To update a user:

1. From the Organization field above the folder tree, select an organization with which the user is defined.
2. In the folder tree, expand the Users folder, right-click the user name, and choose Properties from the pop-up menu to display the User Properties dialog box.

Figure 3-13 User Properties Dialog Box



3. Edit the following fields as desired:
   - User Id
   - E-mail Address
   - Default Organization
   - Calendar — select a business calendar to assign to the user. For details about this feature, see "Administering Business Calendars" on page 3-4.
4. Click OK or click Cancel to cancel the operation.

Removing a User from an Organization

The Remove User facility removes a user that is already defined in the WebLogic Server security realm from the current organization. It does not remove the user from the security realm.

To remove a user from the current organization:

1. From the Organization field above the folder tree, select the organization from which you want to remove the user.
2. In the folder tree, expand the Users folder, right-click the user name and select Remove from the pop-up menu.

3. When prompted with the Remove User warning message, click Yes. To cancel the delete, click No.

Deleting a User

Deleting a user removes it from the WebLogic Server security realm and WebLogic Integration database.

Warning: If you delete a user, you will not be warned about any workflow components that may reference the user. Be sure that you update the following:

Task routing specifications (see "Administering Task Routings" on page 3-29)

Assign Task to User action (see "Assigning a Task to a User" on page 6-46)

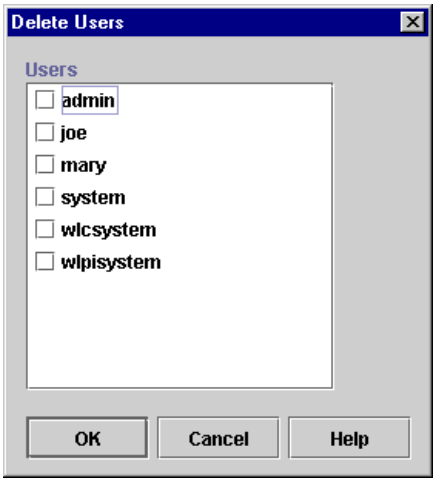
Assign Task Using Routing Table action (see "Assigning a Task Using a Routing Table" on page 6-49)

Send E-mail Message action (see "Sending E-Mail Messages" on page 6-72)

To delete a user:

1. With any organization active, right-click the Users folder, and select Delete Users to display the Delete Users dialog box, which displays all users defined in the system.

Figure 3-14 Delete Users Dialog Box



2. Select the check box to the left of each user that you want to delete, and click OK.
3. When prompted, click Yes to confirm the deletion.

Maintaining Roles

A role is a common area of responsibility, ability, or authorization level that is shared by a group of individuals. A role can only be a member of one organization, but you can use the same name in multiple organizations. For example, you can have a role named *Supervisor* defined in Org1 and Org2. The name of the role is the same, but, in actual fact, the roles are different. *Supervisor* in Org1 is not the same as *Supervisor* in Org2, even though the names are the same.

Roles are mapped to groups in WebLogic Server. Using the previous roles and organizations as an example, you can map *Supervisor* in Org1 to a group called *SupervisorOrg1*, and *Supervisor* in Org2 to a group called *SupervisorOrg2*.

Roles are defined and displayed within each organization. To display the roles that belong to a particular organization, select an organization from the Organizations drop-down list in the WebLogic Integration main window and expand the Roles folder. To display the properties for that role, double-click the role.

Note: To add, update or delete roles, you must have Administer User permission. For more information about permission levels, see “Assigning Permissions to Users and Roles” on page 3-26.

Creating a Role

The Create Role facility allows you to create a new role in the WebLogic Server security realm, and to map the role to a group in WebLogic Server. This facility is accessible only if the WebLogic Integration Studio is operating in a manageable realm.

When you create a role, you must map it to a group on WebLogic Server. You can choose to create the group automatically when you create the role, or you can create the group in the WebLogic Server Administration Console in advance. For information about creating WebLogic Integration groups, see "Defining Groups in the Compatibility Realm" in "Using Compatibility Security" in *Managing WebLogic Security*, in the BEA WebLogic Server documentation set at the following URL:

`http://edocs.bea.com/wls/docs70/secmanage/security6.html`

To add a new role:

1. From the Organization field above the folder tree, select the organization in which you want to create the role.
2. In the folder tree, right-click the Roles folder and choose Create Role from the menu to display the Create Role dialog box.

Figure 3-15 Create Role Dialog Box

3. In the ID field, enter a meaningful name for the role.
4. Optionally, use the Calendar drop-down list to assign a calendar to the role. For details about this feature, see "Administering Business Calendars" on page 3-4.
5. Do one of the following:
   - Use the WLS Group drop-down list to assign the role to an existing group defined in WebLogic Server.
   - Create a new group in WebLogic Server with the same name as the role by clicking the Map to a group with the same name as the role check box.
6. In the Members section of the dialog box, select the check box(es) to the left of the users that you will make members of this role.

7. Click OK to save the new role. Click Cancel to cancel the operation.

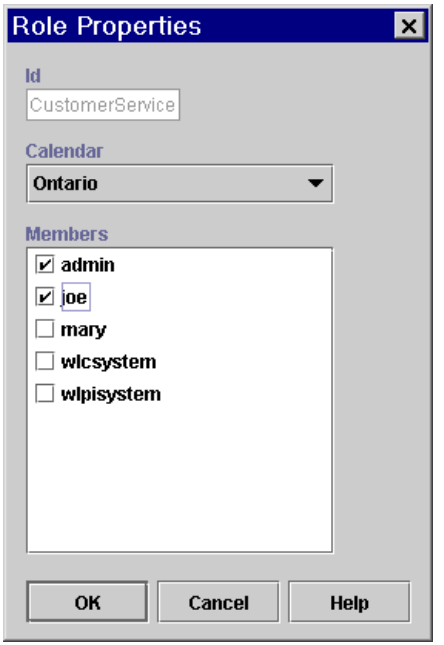
Updating a Role

You can update a role by changing its calendar and members.

To update an existing role:

1. From the Organization field above the folder tree, select the organization in which the role is defined.
2. In the folder tree, expand the Roles folder, right-click the role name and choose Properties from the menu to display the Role Properties dialog box.

Figure 3-16 Role Properties Dialog Box



3. Make changes as needed to the calendar and role members.
4. Click OK to save the update. Click Cancel to cancel the operation.

Deleting a Role

Deleting a role removes it from the WebLogic Server security realm and the WebLogic Integration database.

Warning: If you delete a role, you will not be warned about any workflow actions that may reference the role. Be sure that you update the following actions, that may reference the role, to avoid server exceptions at run time:

Task routing specifications (see "Administering Task Routings" on page 3-29)

Assign Task to User action (see "Assigning a Task to a User" on page 6-46)

Assign Task to Role (see "Assigning a Task to a Role" on page 6-47)

Assign Task Using Routing Table (see "Assigning a Task Using a Routing Table" on page 6-49)

Send E-mail Message (see "Sending E-Mail Messages" on page 6-72)

To delete a role:

1. From the Organization field above the folder tree, select the organization in which the role is defined.
2. Expand the Roles folder, right-click the role name and choose Delete from the pop-up menu.
3. When prompted by the Delete Role warning message, click Yes. To cancel the delete, click No.

Changing the Mapping for Roles

To reroute tasks assigned to a role, you need to modify role mappings. You can reroute the tasks assigned to a role by selecting a WebLogic Server group to which the target role is already mapped.

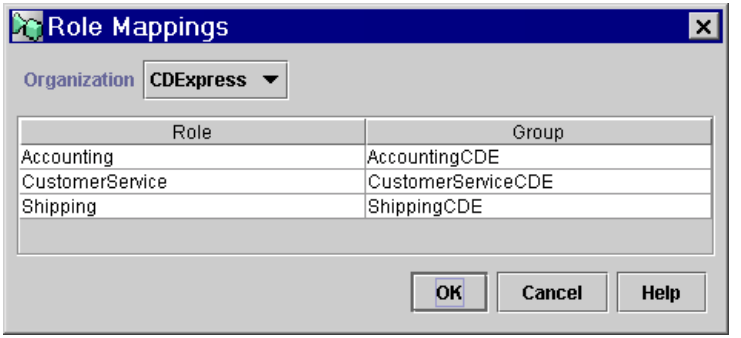
For more information on creating groups in WebLogic Server, see "Defining Groups" in "Configuring WebLogic Security" in the *Managing WebLogic Security* in the BEA WebLogic Server documentation set, at the following URL:

```
http://edocs.bea.com/wls/docs70/secmanage/security7.html
```

To change the mapping for a role:

1. From the main Studio window, choose Configuration—Role Mappings.The Role Mappings dialog box is displayed.

Figure 3-17 Role Mappings Dialog Box



2. From the Organization drop-down list, select the organization containing the roles you want to assign to WebLogic Server groups. The current mappings are displayed in a table.
3. In the table, select the role or group you want to remap. A drop-down arrow appears to the right of the group name to be changed.
4. In the Group field, click the drop-down arrow and, from the drop-down list, select the new WebLogic Server group to which you want to map the role, from among all WebLogic Server groups already defined.
5. Click OK to complete the procedure, or Cancel to cancel the operation.

Assigning Permissions to Users and Roles

Levels of permission enable you to protect and control access to Studio functionality. Roles and users can perform the tasks shown in the following table only if they have the corresponding level of permission.

Table 3-1 Levels of Permission

Permission Level	Allows User or Role to . . .
Configure System	■ Add, update, and delete business calendars ■ Add, update, and delete event key interfaces
Configure Component	■ Define, update, and delete business operations ■ Load and configure plug-ins
Administer User	■ Add, update, and delete users, roles, and organizations ■ Map roles to groups ■ Specify levels of permission for users and roles ■ Work with task routing specifications
Monitor Instance	■ Monitor workflow instances, including updating workflow variables and task properties ■ Work with workload and statistics reports
Create Template	■ Create templates and template definitions ■ Open template definitions ■ Set template and template definition properties ■ Set template and template definition variables
Delete Template	Delete templates and template definitions
Execute Template	■ Start a workflow within a within a particular organization form the Worklist or custom client ■ Set workflow instance variables and properties
Monitor Instance	■ Assign and unassign instance tasks ■ Get and set instance properties

Since users inherit permissions from the role to which it belongs, you will want to define permissions for roles first, and then users. Procedures are provided in the following sections.

Note: To assign permissions to users and roles, you must have Administer User permission. For more information about permission levels, see "Assigning Permissions to Users and Roles" on page 3-26.

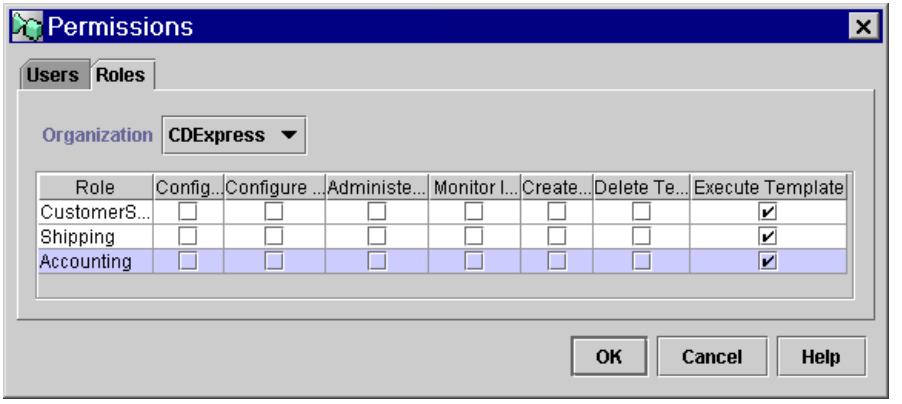
Setting Permissions for Roles

A role inherits the levels of permission for the WebLogic Server group to which it is mapped. You can add and remove levels of permission for a role. Any changes you make to a role are also reflected in the group to which the role is mapped.

To set the levels of permission for a role:

1. From the main Studio window, choose Configuration—Permissions. The Permissions dialog box is displayed
2. Select the Roles tab.

Figure 3-18 Permissions Dialog Box: Roles Tab



3. From the Organization drop-down list, select the organization for which you want to set the role permissions. The Permissions dialog box displays all the roles in the organization and the permissions currently assigned.

4. Select or clear the check boxes as desired.
5. Click OK to accept the changes, or Cancel to cancel the operation.

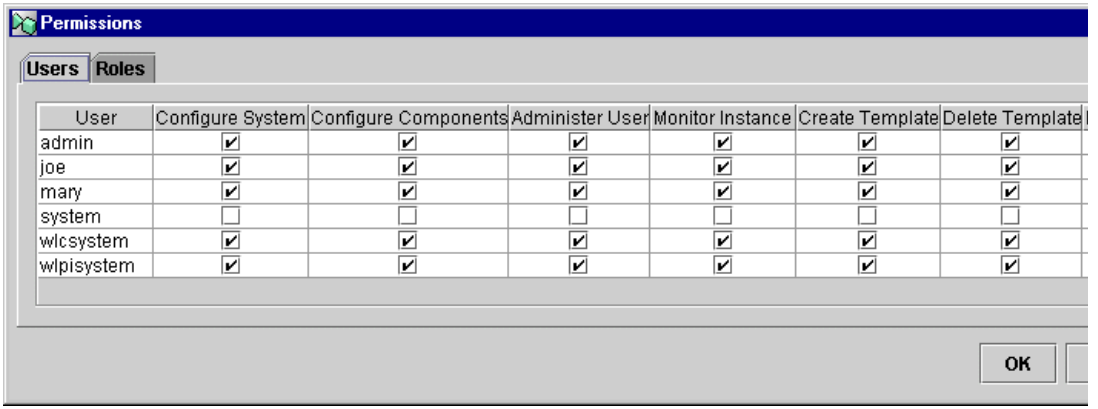
Setting Permissions for Users

A user inherits the levels of permission for the role to which it belongs. You can add other levels of permission to a user that are not defined for the role to which it belongs, but you cannot remove the permissions a user inherits from the role to which it belongs. Any permissions you add to a user are specific to the user, and are not reflected in the role to which the user belongs.

To set levels of permission for a user:

1. From the main Studio window, choose Configuration→Permissions. The Permissions dialog box is displayed.
2. Select the Users tab (if it is not already selected).

Figure 3-19 Permissions Dialog Box: Users Tab



The Permissions dialog box displays all the currently defined users, and the permissions currently assigned. If a permission is checked, but grayed out, you cannot remove that permission because it was inherited from the role to which the user belongs.

3. Select or clear the check boxes as desired.

4. Click OK to accept the changes, or Cancel to cancel the operation.

Administering Task Routings

Task routings can be defined on a per-organization basis to reroute currently assigned tasks from one user to another user or role for a specified temporary period of time. Task routings reroute *all* tasks assigned to the user you specify. You can route tasks from users, roles, and users in roles. For more information about these distinctions, and about task assignment, see "Setting Up Manual Tasks" on page 6-44.

Note: The Routing feature reroutes only tasks assigned to a user; it does not reroute tasks assigned to a role. Rerouted tasks can be directed, however, to another user, user in role, or role. To reroute tasks assigned to a role, see "Changing the Mapping for Roles" on page 3-24.

You can also reroute individual tasks according to particular conditions. This is done with an action specified within a task node. For more information, see "Assigning a Task Using a Routing Table" on page 6-49.

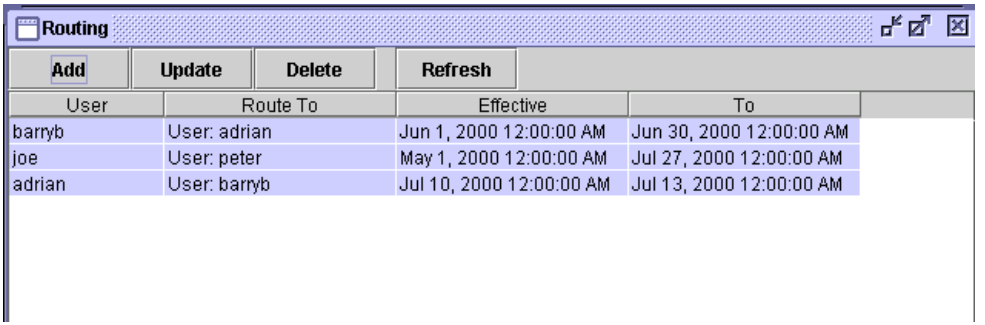
Notes: To administer task routings, you must have Administer User permission. For more information about permission levels, see "Assigning Permissions to Users and Roles" on page 3-26.

Viewing Task Routing Specifications

To view task routing specifications for an organization:

1. From the Organization field above the folder tree, select the organization whose routings you want to view.
2. Right-click the Routing folder, and select Open from the pop-up menu, to display the Routing window.

Figure 3-20 Routing Dialog Box



The Routing window displays the following information for each routing specification:

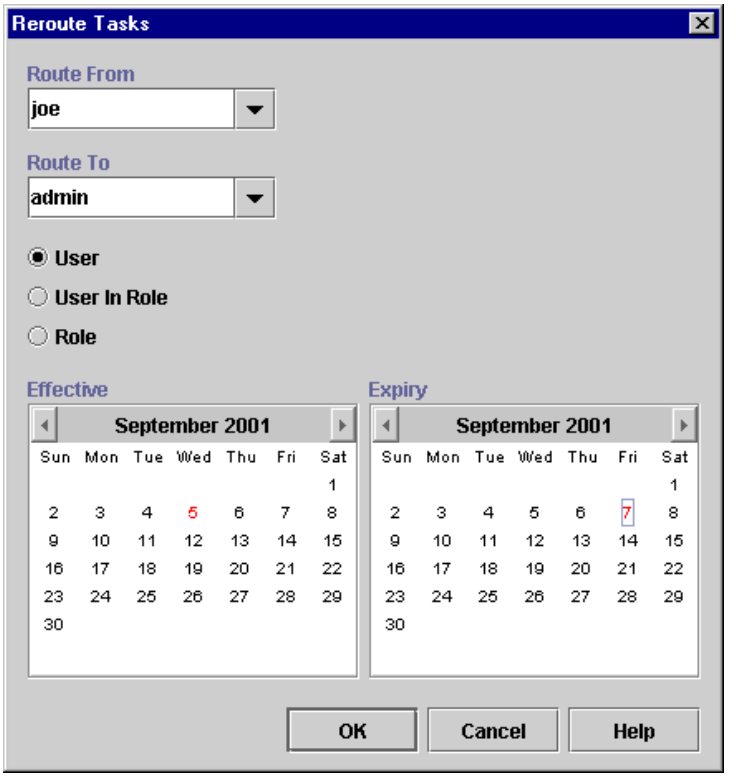
User	The user ID for the user from whom all tasks are to be routed.
Route To	The ID of the user, role or user in role to whom all tasks are to be routed.
Effective	The date and time when task routing is to begin.
To	The date and time when task routing is to end.

Adding a Routing Specification

To add a task routing specification:

1. From the Organization field above the folder tree, select the organization to which you would like to add a routing specification.
2. In the folder tree, right-click the Routing folder, and from the pop-up menu, select Open to display the Routing window.
3. In the Routing window, click Add to display the Reroute Tasks dialog box.

Figure 3-21 Reroute Tasks Dialog Box



4. From the Route From drop-down list, select the user from which all tasks should be rerouted.

5. From the Route To drop-down list, select the User, User In Role, or Role to which all tasks should be rerouted, and select the corresponding radio button.

   Note: When you assign to a User in Role, the system performs workload balancing by first reviewing the number of tasks assigned to all users in the role, selecting the user with the least number of assigned tasks, and assigning all of the rerouted tasks to this user.

6. Specify an Effective date by selecting a month and year, and clicking on a day in the month display.

7. In a similar fashion, specify the Expiry date for the task rerouting.

8. Click OK to save the rerouting specification. Click Cancel to cancel the operation.

Updating a Task Routing Specification

To update a task rerouting specification:

1. From the Organization field above the folder tree, select the organization to which you would like to add a routing specification.
2. In the folder tree, right-click the Routing folder, and from the pop-up menu, select Open to display the Routing window.
3. In the list of displayed task routings, select the task rerouting you want to update.
4. Click Update to display the Reroute Tasks dialog box.
5. Make changes as needed to the Route To, Effective, and Expiry values.
6. Click OK to save the changes. Click Cancel to cancel the operation.

Deleting a Task Routing Specification

To delete a task rerouting specification:

1. From the Organization field above the folder tree, select the organization to which you would like to add a routing specification.
2. In the folder tree, right-click the Routing folder, and from the pop-up menu, select Open to display the Routing window.
3. In the list of displayed task routings, select the task rerouting you want to update.
4. When prompted by the Delete Reroute warning message, click Yes. To cancel the delete, click No.

Refreshing the Rerouting Task List

Click Refresh in the Routing dialog box to refresh the rerouting task list and display any changes that have been made since you first invoked the Routing dialog box.

4 Configuring Workflow Resources

This section describes how to configure system and application components, and includes:

- Overview of Resource Configuration Tasks
- Configuring Plug-Ins
- Configuring Business Operations
- Configuring Event Keys
- Managing Entities in the Repository

Overview of Resource Configuration Tasks

All of the tasks described in this section can be performed before or during workflow design, but the resources described can be configured without accessing workflow templates. In some cases, workflow design activities are actually dependent on these resources having been set up already, such as business operations, which must be defined before a workflow can call them. Once the resources are defined, they are available globally for access by all workflows, users, and organizations in the system. These tasks do not need to be performed in any particular order.

Note: You may also want to familiarize yourself with the workflow expression language and the Studio's Expression Builder and XPath Wizard tools if you will be configuring event keys. Complete information on workflow expressions is available in Chapter 8, "Using Workflow Expressions."

- Load and configure plug-ins. If you have developed custom client or server components that add functionality to be accessed through the Studio, you will want to configure these beforehand. For information on developing plug-ins, see *Programming BPM Plug-Ins for WebLogic Integration*. Information on loading and configuring plug-ins is provided in "Configuring Plug-Ins" on page 4-3.

- Define business operations. If you have developed Java components, such as custom classes or Enterprise JavaBeans (EJBs), you need to set up interfaces to them, in the form of *business operations* that are then called by workflows to invoke the functionality provided in the Java component's methods. Business operations must be defined before workflows can invoke them. Procedures are given in "Configuring Business Operations" on page 4-7. Alternatively, import previously exported business operations from existing workflow packages; see procedures in "Importing Workflow Packages" on page 11-5.

- Configure event keys. If your workflows or nodes within them are to be triggered by *events*, such as incoming XML messages on a Java Message Service (JMS) queue, you can set up event keys that retrieve the relevant data from the incoming documents to trigger those events. You can also configure the event keys while designing workflows, so setting them up ahead of time is optional. Details are provided in "Configuring Event Keys" on page 4-18. Alternatively, import previously exported event keys from existing workflow packages; see procedures in "Importing Workflow Packages" on page 11-5.

- Set up the repository. If you will be using any XML data transformation operations, or sending outgoing XML messages to an internal queue or an external topic or queue, you may want to import XML documents, style sheets, and other entities beforehand into the repository, which provides a centralized database for convenient access by any connecting Studio client. Information is provided in "Managing Entities in the Repository" on page 4-23. Alternatively, import previously exported repository entities from existing workflow packages; see procedures in "Importing Workflow Packages" on page 11-5.

Configuring Plug-Ins

A plug-in is a group of Java classes implemented as EJBs that extend the functionality provided in selected workflow components. Plug-ins provide a way to customize existing WebLogic Integration features so they are more appropriate for your environment, and to add functionality that is specific to your environment.

A plug-in can extend the functionality of the following workflow components:

- Workflow templates
- Workflow template definitions
- Start nodes
- Event nodes
- Done nodes
- Variables
- Actions
- Functions (that are part of expressions)

If a plug-in is developed for any of these workflow components, the Studio dialog box for these components is also modified to include a way for you to access the plug-in's functionality. For example, the Studio provides several default methods in the Properties dialog box of a Start node to trigger the start of a workflow: timed, manual, called, and event. To extend the default methods, a developer can create a plug-in that specifies a custom event to trigger a workflow, such as receiving an e–mail message, which might be the preferred method for starting workflows in your environment. This plug-in method will appear as an option in the Start Properties dialog box.

Once a plug-in is developed and is deployed on WebLogic Server, it is available for use by WebLogic Integration. When WebLogic Integration starts, it checks WebLogic Server to see if any plug-ins are available on the server.

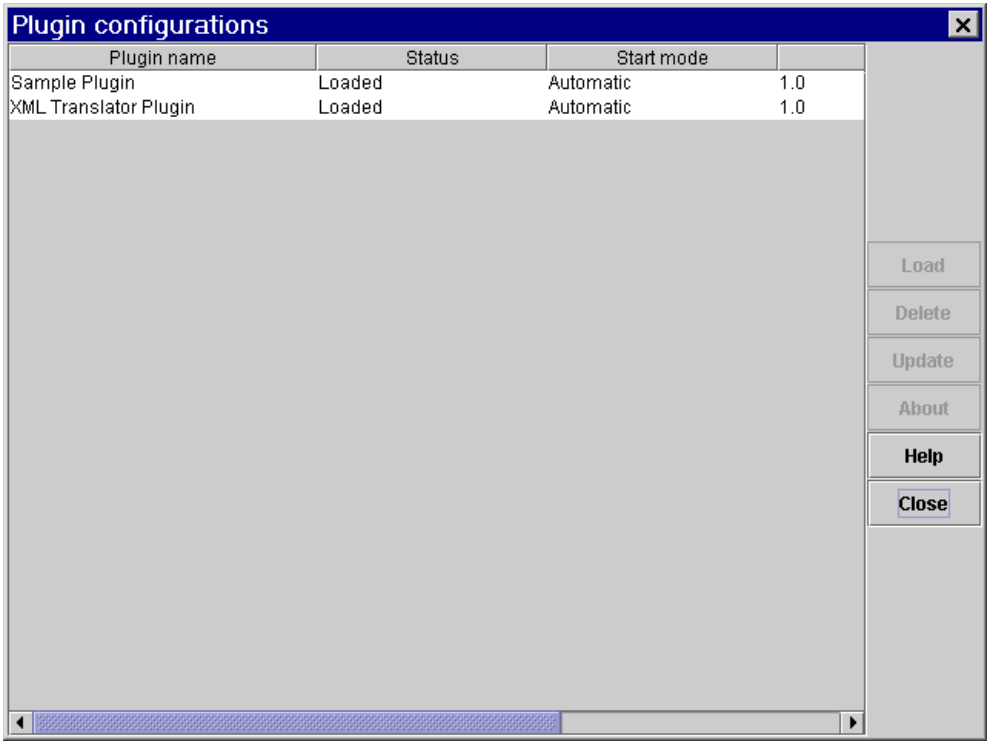
Before you can use an available plug-in, you must load it using the Studio to activate it. You might also need to specify some configuration settings for the plug-in before you can use it.

Note: To load or configure a plug-in, you must have Configure Components permission. For details about permission levels, see "Assigning Permissions to Users and Roles" on page 3-26.

Viewing Plug-ins

To view a plug-in, choose Configuration—Plugins to display the Plugin configurations dialog box.

Figure 4-1 Plugin Configurations Dialog Box



The information displayed for each plug-in is explained in the following table.

Plugin Name	The name for the plug-in as specified by the plug-in software.
Status	Loaded—The plug-in is loaded Initialized—The plug-in is available but has not been loaded Missing—The plug-in has a registered configuration, but is not deployed or available. Error—The plug-in threw an exception when called, or requires a later version of the plug-in framework.
Start mode	Automatic—The plug-in is loaded each time the server is started Manual—The plug-in must be loaded manually each time you start the server Disabled—The plug-in cannot be loaded
Version	The software version for the plug-in
Vendor	The name of the company that supplied the plug-in

You can also obtain information about a plug-in by selecting the plug-in from the list and clicking About.

Loading Plug-Ins

If a plug-in's start mode is manual, you can load it each time a WebLogic Integration server session is started. If a plug-in's start mode is disabled, you must first change it to manual or automatic before you can load it.

To load an initialized plug-in:

1. In the Plugin configurations dialog box, select the desired plug-in.
2. Click Load. The status of the plug-in changes to Loaded in the list.

To load a disabled plug-in:

1. In the Plugin configurations dialog box, select the desired plug-in, and click Update to display the Configuration dialog box.

2. In the Configuration dialog box, change the Start Mode to Automatic or Manual, and click OK.
3. With the plug-in selected in the Plugin configurations dialog box, click Load.

The status of the plug-in changes to Loaded in the list.

Note: Changes to the plug-in's start mode do not take effect until the WebLogic Integration server is restarted.

Updating a Plug-In Configuration

To configure a plug-in:

1. In the Plugin configurations dialog box, select the plug-in you want to configure.
2. Click Update. A configuration dialog box is displayed.
3. Optionally, select the start mode for the plug-in by clicking one of the following buttons:
   - Automatic — Select this option if you want to load the plug-in each time the server is started
   - Manual — Select this option if you want to load the plug-in manually as required
   - Disabled — Select this option to disable the plug-in functionality

   Note: The start mode does not become effective until the next time the WebLogic Integration server is started.
4. Specify configuration settings for the plug-in as appropriate.

   For details about defining configuration settings for a plug-in, see the online help for that plug-in. To access plug-in help, press the F1 key (for context-sensitive plug-in help), or choose Help—Plugin Help from the main menu in the Studio and select the appropriate plug-in help from the menu.
5. Click OK to complete the configuration or Cancel to cancel the operation.

Deleting a Plug-In Configuration

You can delete a configuration for a plug-in if you no longer need the configuration. When you delete a configuration, you do not delete the plug-in itself. You just delete its registered configuration.

Note: You cannot delete the configuration of a plug-in unless the status of that plug-in is Missing.

To delete a configuration:

1. From the Plugin configurations dialog box, select the plug-in whose configuration you want to delete.
2. Click Delete. The registered configuration for the selected plug-in is deleted, but the plug-in remains visible in the Plugin configurations dialog box.

   The following table describes the actions that occur when you restart the WebLogic Integration server after a plug-in configuration has been deleted.

If the WebLogic Integration server is restarted and the plug-in is . . .	Then . . .
Not deployed	The plug-in manager does not search for the plug-in and the plug-in is not listed in the Plugin configurations dialog box.
Deployed	The plug-in is automatically loaded using the default configuration values defined by the plug-in.

Configuring Business Operations

To enable workflows to invoke software components that perform business logic, such as Java classes and Enterprise JavaBeans (EJBs), you define business operations. A business operation represents a method call on a Java class or EJB, including any variables that are passed to it as parameters, and result values that are returned to the

workflow. You can use the business operations facility to create customized functions that invoke existing applications or applications that are built specifically for the workflow. The business operations facility displays all Java classes and EJBs registered in WebLogic Server, as well as their methods and parameters.

Note: For information about deploying Java classes and EJBs so that they are visible in the Studio, see "Deploying EJBs and Java Classes for Business Operations" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

Once the business operation is defined, it is then globally available for all workflows in the system to invoke. You can also export and import business operations as part of Java archive package files, with or without the workflows that reference them (for more information, see Chapter 11, "Importing and Exporting Workflow Packages.")

Within an individual workflow, you use the Perform Business Operation action to invoke the business operation and, optionally, assign the results of the method call to a workflow variable. For more information, see "Calling a Business Operation" on page 6-78.

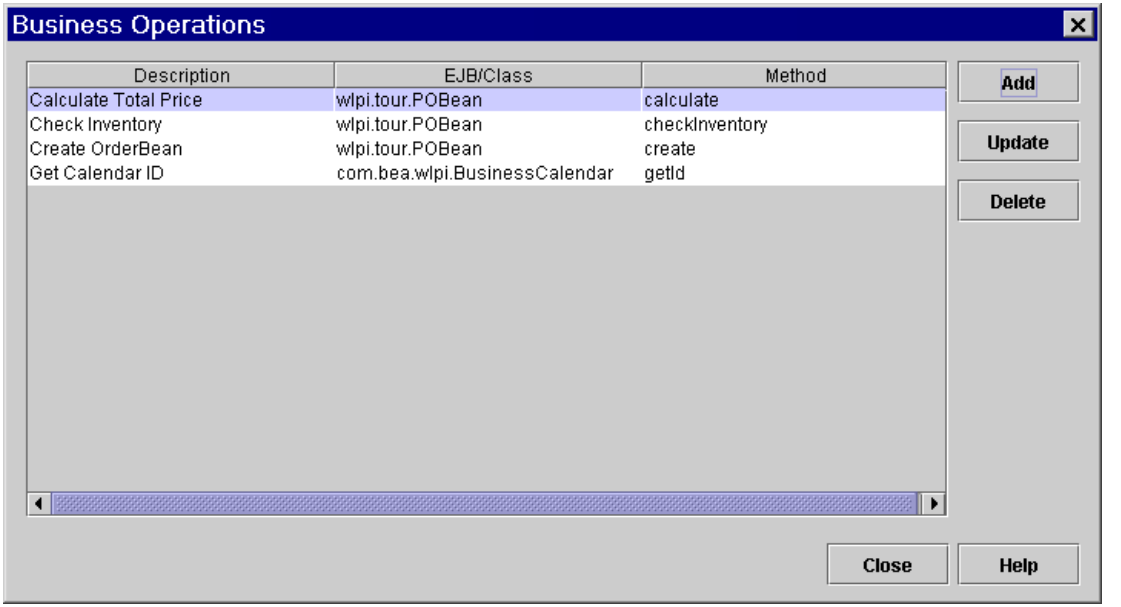
Note that before a workflow can call a method on an EJB or a non-static method on a Java class, the workflow must call its constructor method to create an instance of the EJB or Java class on the server. Therefore, you must be sure to create a business operation for the `create()` method of EJBs and a constructor method of Java classes, and a variable to store a reference to the instance. More information is provided in the following sections, which describe how to add and define business operations for each Java component type, and in "Calling a Business Operation" on page 6-78, which describes the steps for calling the necessary methods and assigning variables. An example of defining business operations is also provided in "Creating and Performing a Business Operation: Defining the Check Inventory Task" in *Learning to Use BPM with WebLogic Integration*.

Note: To add, define, or delete a business operation, you must have Configure Components permission. For details about permission levels, see "Assigning Permissions to Users and Roles" on page 3-26.

Viewing Business Operations

To view a business operation, choose Configuration—Business Operations to display the Business Operations dialog box.

Figure 4-2 Business Operations Dialog Box



The information displayed for each business operation is explained in the following table.

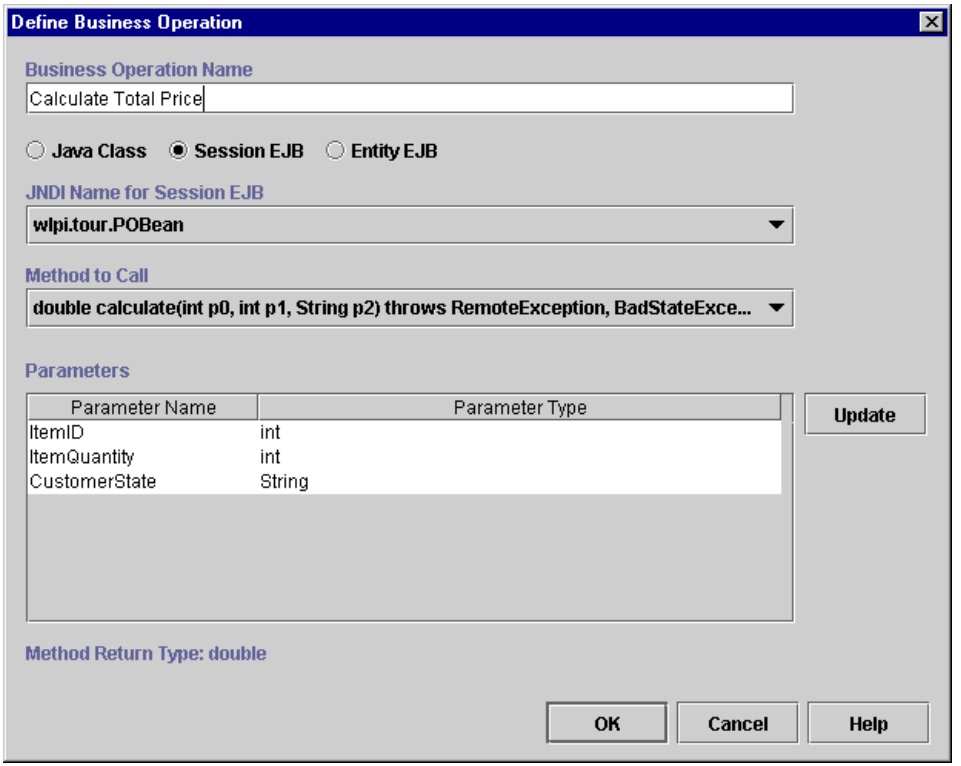
Description	The name that you define for the business operation.
EJB/Class	The Enterprise JavaBean (EJB) or Java class to be invoked.
Method	The method in the EJB or Java class to be called.

Adding a Business Operation

To create and define a business operation:

1. From the Business Operations dialog box, click Add to display the Define Business Operation dialog box.

Figure 4-3 Define Business Operation Dialog Box



2. In the Name field, enter a meaningful description of the business operation. A business operation that returns the number of items available in stock might be called, for example, `Check Inventory`. For methods that create an instance of the Java class or EJB at run time, give the business operation a name that indicates the purpose of the method and the name of the class or Bean that it creates, such as `Create Order Processing EJB Instance`.

   Note: You will also need to create a corresponding variable to reference the instance.

3. Specify the software component the business operation is invoking. The choices are:

   - Java Class — the business operation calls a method in a Java class on WebLogic Server. The Java class can be serializable or non-serializable.

Non-serializable Java class references persist for the duration of a transaction only, and will need to be recreated each time the workflow instance reaches a quiescent state. Follow the steps given in "Adding a Business Operation for a Java Class" on page 4-11.

- Session EJB — the business operation calls a method in a session EJB on WebLogic Server. Stateful session EJB references persist for the duration of a transaction only, and will need to be recreated each time the workflow instance reaches a quiescent state. Stateless session EJB references persist for the duration of a workflow instance. Follow the steps given in "Adding a Business Operation for a Session EJB" on page 4-13.
- Entity EJB — the business operation calls a method in an entity EJB on WebLogic Server. Entity EJBs represent persistent data, and persist for at least the duration of the workflow instance and, often, beyond the life of a workflow instance. Follow the steps given in "Adding a Business Operation for an Entity EJB" on page 4-15.

4. In the Define Business Operations dialog box, click OK to save the business operation. The business operation is added to the list of valid business operations in the Business Operation dialog box.

Adding a Business Operation for a Java Class

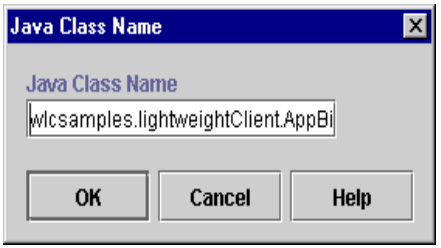
If you are creating business operations for non-static methods in a Java class, you must also create a business operation for a constructor method for the class. (Details are given in "Calling the Business Operation to Create an EJB or Java Class Instance" on page 6-79.) Be sure to give this business operation a meaningful name that identifies its function, as it will need to be called from the workflow before any non-static methods in the class may be called. You will also need to create a variable of type Java Object to hold a reference to the Java class instance when the instance is created at run time. For information on variables, see "Working with Variables" on page 5-28.

Finally, when you name your business operation, it is a good idea to indicate whether the business operation calls a static or non-static method, so the workflow designer will know whether or not a constructor method needs to be called first.

To define a business operation for a Java class, proceed as follows:

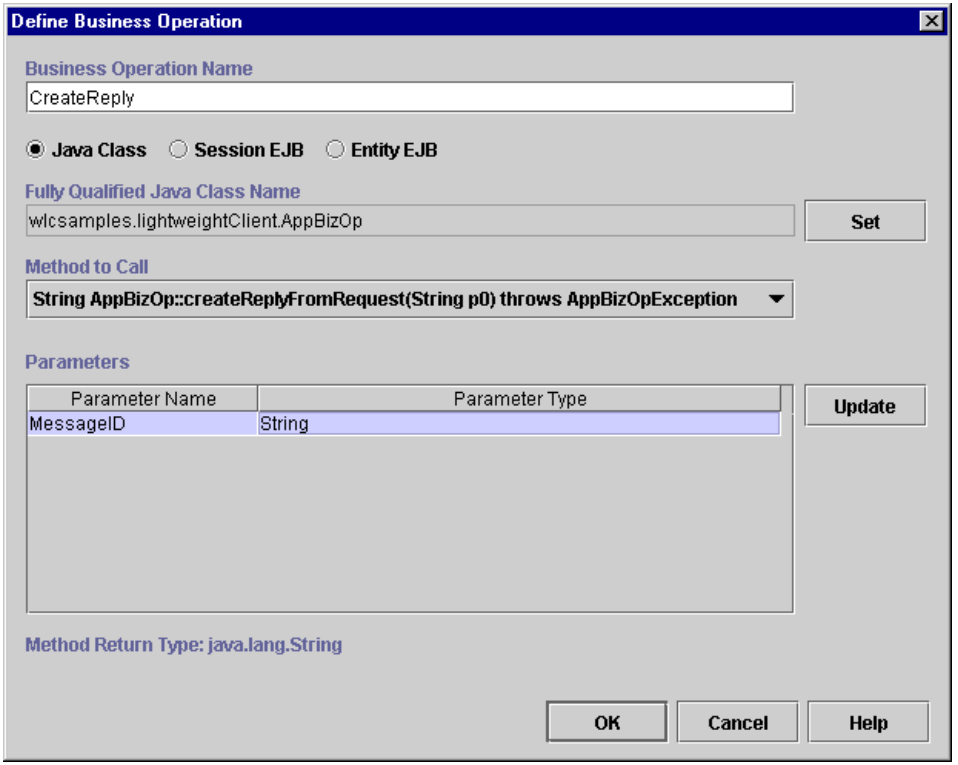
1. Select the Java Class radio button.
2. Click Set to display the Java Class Name dialog box.

Figure 4-4 Java Class Name Dialog Box



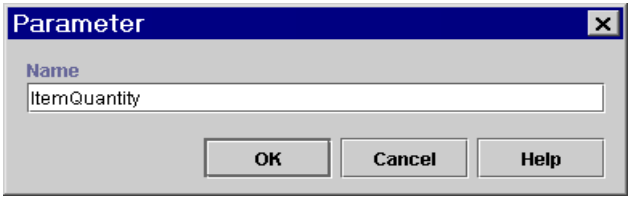
3. Enter a fully qualified Java class name (for example, `java.lang.String`), and click OK. The Java class name is set in the Fully Qualified Java Class Name field of the Define Business Operation dialog box.

Figure 4-5 Define Business Operation Dialog Box: Java Class Option



4. In the Method to Call drop-down list, select a method to call upon invoking the business operation in the workflow. There are three types of Java class methods in the list:
   - Constructor Type — these method types are first in the list. Constructor methods return an object of the type you entered as the Java Class Name.
   - Method Type — these are second in the list. They are invoked on Java objects and can return any type.
   - Static Type — these are last in the list and begin with the word `Static`. Static methods require no object to be created and can return any type.
5. Optionally, give the method's parameters meaningful names by highlighting a parameter in the Parameters list and clicking Update to display the Parameter dialog box.

Figure 4-6 Parameter Dialog Box



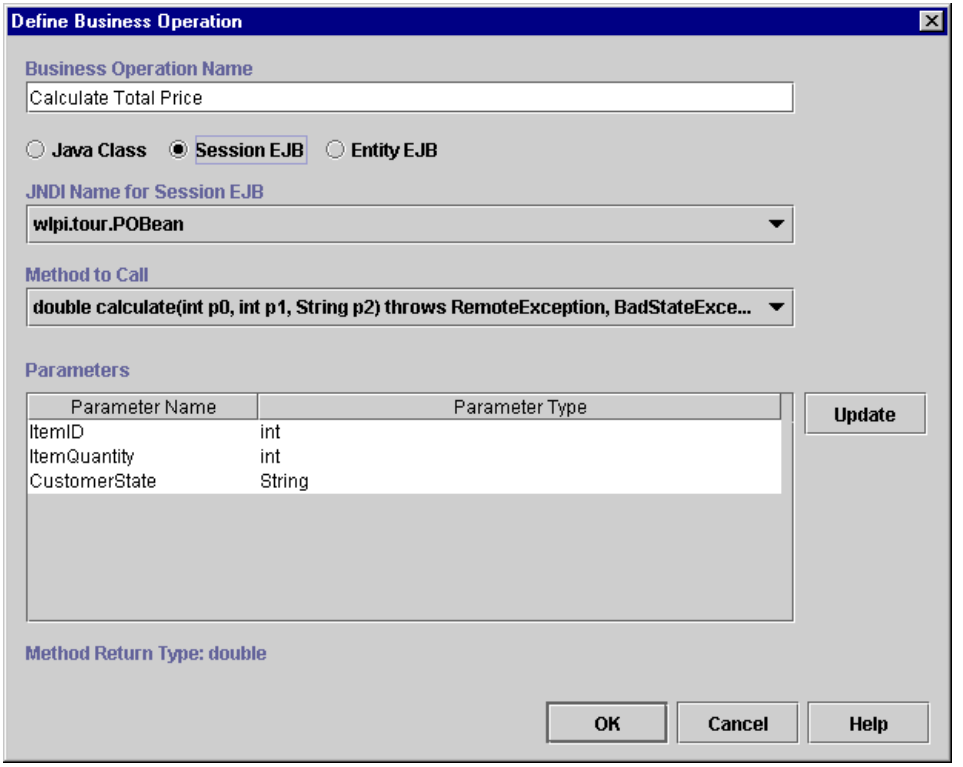
6. In the Name field, enter a descriptive name for the parameter and click OK.
7. Repeat steps 5 and 6 for all parameters in the Parameters in the list.

Adding a Business Operation for a Session EJB

In addition to creating business operations that call methods that provide business logic for your workflow, you must create a business operation for the `create()` method of the Session EJB whose methods you are calling via a business operation. (Details are given in "Calling the Business Operation to Create an EJB or Java Class Instance" on page 6-79.) Be sure to give this business operation a meaningful name that identifies its function, as it will need to be called from the workflow in each transaction that calls other methods. You will also need to create a variable of type Session EJB to hold a reference to the EJB instance when the instance is created at run time. For information on variables, see "Working with Variables" on page 5-28.

All Session EJBs deployed on WebLogic Server are displayed in a list in the Define Business Operations dialog box according to their Java Naming and Directory Interface (JNDI) name.

Figure 4-7 Define Business Operation: Session EJB Option



To define a business operation for a Session EJB:

1. Select the Session EJB radio button to display fields relevant to Session EJBs in the Define Business Operation dialog box.
2. From the JNDI Name for Session EJB drop-down list, select the JNDI name for the Session EJB.
3. In the Method to Call drop-down list, select a method to call upon invoking the business operation in the workflow.

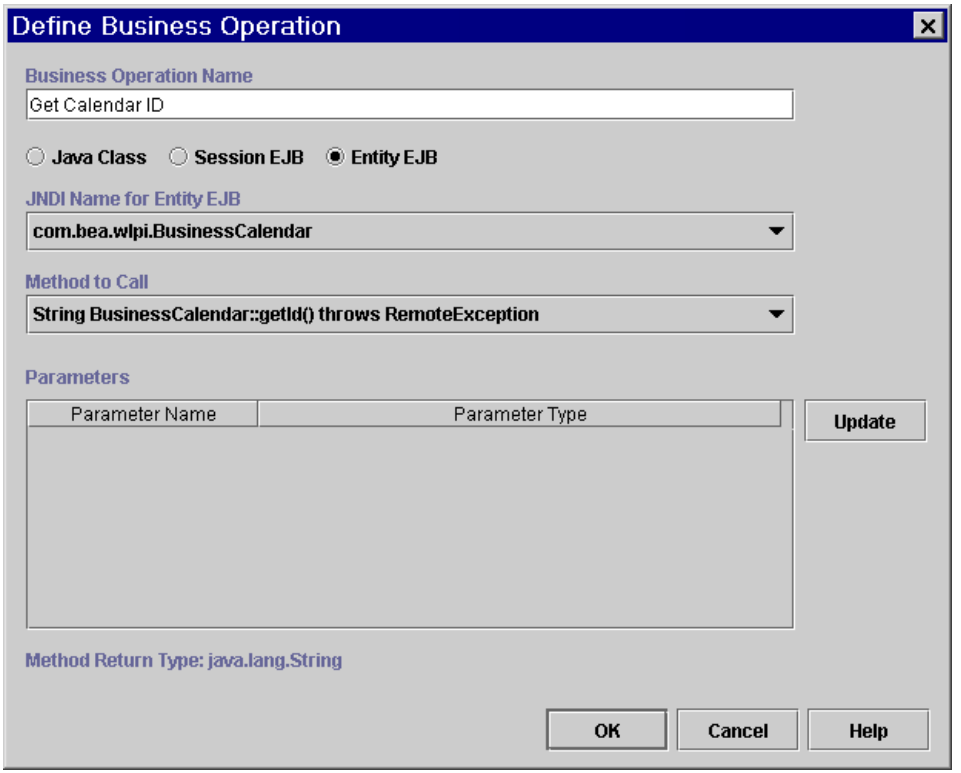
4. Optionally, give the method's parameters meaningful names by highlighting a parameter in the Parameters list and clicking Update to display the Parameter dialog box.
5. In the Name field, enter a descriptive name for the parameter and click OK.
6. Repeat steps 4 and 5 for all parameters in the Parameters in the list.

Adding a Business Operation for an Entity EJB

In addition to creating business operations that call methods that provide business logic for your workflow, you must create a business operation for the `create()` method of the Entity EJB whose methods you are calling via a business operation. (Details are given in "Calling the Business Operation to Create an EJB or Java Class Instance" on page 6-79.) Be sure to give this business operation a meaningful name that identifies its function, as it will need to be called from the workflow before other methods in the EJB may be called. You will also need to create a variable of type Entity EJB to hold a reference to the EJB instance when the instance is created at run time. For information on variables, see "Working with Variables" on page 5-28.

All Entity EJBs deployed on WebLogic Server are displayed in a list in the Define Business Operations dialog box according to their JNDI name.

Figure 4-8 Define Business Operation: Entity EJB Option



To define a business operation for an Entity EJB:

1. Select the Entity EJB radio button to display fields relevant to Entity EJBs in the Define Business Operation dialog box.
2. From the JNDI Name for Entity EJB drop-down list, select the JNDI name for the Entity EJB.
3. In the Method to Call drop-down list, select a method to call upon invoking the business operation in the workflow.
4. Optionally, give the method's parameters meaningful names by highlighting a parameter in the Parameters list and clicking Update to display the Parameter dialog box.

5. In the Name field, enter a descriptive name for the parameter and click OK.
6. Repeat steps 4 and 5 for all parameters in the Parameters in the list.

Updating a Business Operation

When updating a business operation, you should be sure to update any Perform Business Operation actions that reference the business operation from workflows. For more information on this action, "Calling a Business Operation" on page 6-78.

To update a business operation:

1. In the Business Operations dialog box, select the business operation you want to update, and click Update. The Define Business Operation dialog box is displayed.
2. Make changes as needed, and click OK when done.

Deleting a Business Operation

Warning: Before deleting a business operation, make sure that the business operation is not referenced by any workflows using the Perform Business Operation action, or you will not be able to activate the workflow. When making a deletion, you will not be warned if any references exist, so be sure to update the Perform Business Operation action accordingly (for information, see "Calling a Business Operation" on page 6-78).

To delete a business operation:

1. In the Business Operations dialog box, select the business operation you want to delete, and click Delete.
2. When prompted by a warning message, click OK to confirm the delete, or Cancel to cancel.

Configuring Event Keys

A workflow can be started, or nodes within a workflow triggered, by an *event*. An event is an asynchronous notification from another workflow or from an external source, such as another application. Start nodes can be defined as event-triggered, and Event nodes are always asynchronous nodes that can only be triggered by an external event.

An event notification most typically takes the form of an XML document contained in a Java Message Service (JMS) message and received on a JMS queue, although it may also be plug-in defined, which means that the event notification can be a custom trigger rather than an XML document. (For more information, see *Programming BPM Plug-Ins for WebLogic Integration*).

In an XML event type, the actual trigger is either the document type declaration (DOCTYPE) specified in the prolog of the XML message, or it is the root element of the XML message. You specify the DOCTYPE or root element with which you want to trigger the event or start the workflow in a Start or Event node's properties dialog box. The event is not triggered unless the DOCTYPE or root element specified in the node's properties dialog box matches that in the incoming XML message.

In addition to using the DOCTYPE or root element, you can further qualify an event with an event *key*. An event key allows you to specify the contents of incoming XML messages or of JMS header or property fields that will trigger a Start or Event node. That is, rather than allowing all incoming XML documents with a particular DOCTYPE or root element to trigger the node, you can filter the instances of incoming XML messages according to specific values contained in the document or header, so that only a message, or messages, containing those values can trigger the node in the running workflow.

An event key consists of two parts:

- *Key value expression*

  You specify the key value expression in the Properties dialog box for Start or Event nodes. The key value expression is a workflow expression that is evaluated at run time to specify the exact data that the incoming message must contain for the node to be triggered. In a Start node, the expression typically contains a constant that refers to particular, recurring data contained in the incoming XML document, or it could be the value of a JMS header. In an Event

node, the expression typically contains variables or functions to obtain a unique value at run time. Examples of key value expressions are given in "Understanding Event Keys" on page 5-39 and steps for defining key value expressions are given in "Defining Event-Triggered Start Properties" on page 5-46 and "Defining Event Properties" on page 5-49.

- *Event key expression*

  This is an expression that returns the key value from the header or body of the incoming message at run time and converts it to the data type required by the corresponding key value expression in a Start or Event node. You specify the event key expression in an event key expression dialog box that you access from the Configuration menu. The expression typically contains an XPath language expression to parse the XML document, or an EventAttribute() function expression to extract a value from a JMS message header.

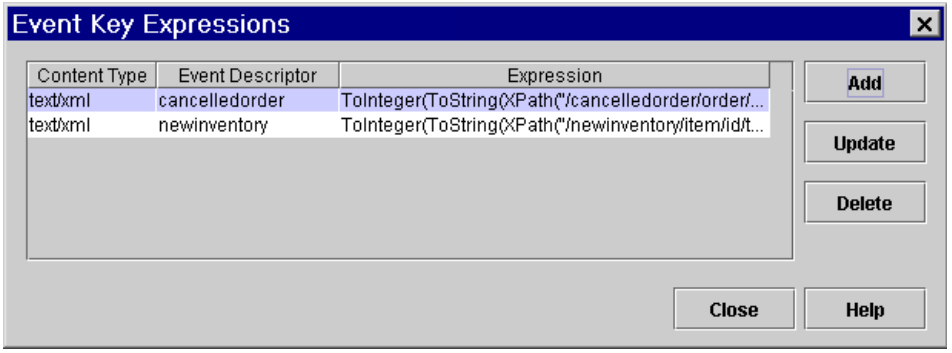
In the event key configuration, you specify an *event descriptor*, which corresponds to the DOCTYPE or root element specified in a Start or Event node properties dialog box. You then specify an event key expression, which corresponds to a key value expression defined in the Start or Event properties dialog box, so that the process engine can compare the two values at run time and determine if there is a match. The relationship between event descriptors and DOCTYPE/Root elements, and between event key expressions and key value expressions is discussed more fully, with examples, in "Understanding Event Keys" on page 5-39.

However, since you can configure event keys independently of workflows, the procedures for doing so are given below. Once you have configured an event key expression, it is available for all workflows in all organizations. If you know the contents of your incoming messages, you can configure your event key expressions in advance to make them available to the workflow designer who will set up corresponding key value expressions in Start or Event nodes. You can also export and import event keys to and from a Java archive package file, with or without the workflows that reference them; for more information, see Chapter 11, "Importing and Exporting Workflow Packages."

Viewing Event Key Configurations

To view an event key configuration, choose Configuration—Events to display the Event Key Expressions dialog box.

Figure 4-9 Event Key Expressions Dialog Box



The information displayed for each event key is explained in the following table.

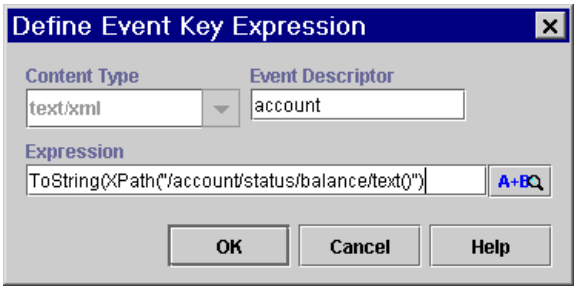
Content Type	By default, this is preset to text/xml and cannot be edited, unless a plug-in event type is available with a loaded plug-in.
Event Descriptor	For an XML/JMS event, this is the DOCTYPE or root element of the incoming XML document contained in the JMS message.
Expression	The expression that returns the key value from the header or body of the incoming message at run time and converts it to the data type required by the corresponding key value expression in a Start or Event node.

Adding an Event Key Configuration

To add an event key configuration:

1. From the Event Key Expressions dialog box, click Add to display the Define Event Key Expression dialog box.

Figure 4-10 Define Event Key Expression Dialog Box



2. In the Content Type field, select text/xml for an XML/JMS message.
3. In the Event Descriptor field, enter the DOCTYPE or root element of the incoming XML document.
4. In the Expression field, enter one of the following:
   - To extract a value from a JMS header or property field, construct an expression using the `EventAttribute()` function, with the name of the field inside the parentheses.
   - To extract a value from the XML body, use an expression using the `XPath()` function (see "XPath()" on page 8-10), or the dot notation for XML elements to be returned as strings (for information, see "XML Element Dot Notation" on page 8-12). You can also use the Expression button A+B to invoke the XPath Wizard, from which you can generate XPath expressions automatically from a sample incoming document. For information, see "Creating XPath Expressions Using the XPath Wizard" on page 8-31.

   For more information on the syntax for event key expressions, see "Extracting Run-Time Event Data" on page 8-7. Note also that an XPath() or EventAttribute() function must be wrapped in a typecasting function to return a data type that matches the type returned by the key value expression defined in the corresponding Event or Start node. For information on typecasting functions, see "Converting Data Types" on page 8-17.

   Note also that the event key expression that you enter here should return a value that matches a value specified by a key value expression in a Start or Event node. For more information, see for defining key value expressions in Start or Event nodes, see "Defining Event And Event-Triggered Start Properties" on page 5-38.

5. Click OK. The event key is stored in an event key table in the WebLogic Integration database, and is displayed in the Event Key Expressions dialog box.
6. Click Close.

Updating an Event Key Configuration

When updating an event key configuration, you can only update the expression, but not the event descriptor.

To update an event key:

1. From the Event Key Expressions dialog box, select the event key you wish to update, and click Update to display the Define Event Key Expression dialog box.
2. Edit the Expression as necessary.
3. Click OK when done.

Deleting an Event Key Configuration

When deleting an event key, take care that the key value is not referenced by key value expressions used in Start or Event nodes in workflows, or these events will not be triggered.

To delete an event key:

1. From the Event Key Expressions dialog box, select the event key you wish to delete, and click Delete.
2. When prompted by a warning message, click OK to confirm the delete, or click Cancel.

Managing Entities in the Repository

The WebLogic Integration repository contains a database table that is used to store XML entities, such as XML documents, DTD files, and XSL template documents. You can use the Studio to view, organize, and populate the repository, to make existing XML entities available globally for use and reuse by all workflows in the system. If you have previously-stored XML documents that you would like to reference in the workflows you create, you can set up your repository so that these documents are available to any client logged on to the system.

For example, if you will be using the Send XML to Client action to interact with a Worklist user (for information, see "Sending an XML Message to the Worklist Application" on page 6-61), you can populate the repository with the document type definition (DTD) files that are used to send messages to the Worklist, so that any Studio client user can easily access them in a central place. Or if you will be using the XSL Transform action to translate XML documents at run time (for information, see "Transforming XML Documents" on page 6-96), you may want to store your XSL stylesheet transform documents in the repository for easy access when defining that action.

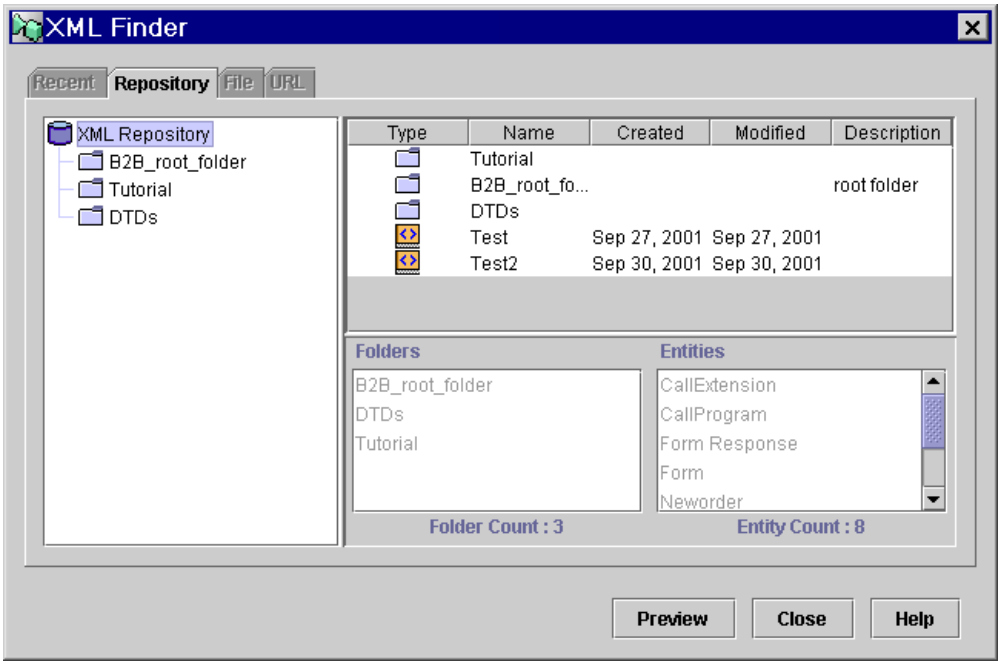
In this section, we discuss how to set up the repository initially. However, the repository and all of the functions described in this section can be also be accessed from within workflow dialog boxes, as described in Chapter 7, "Working with XML Entities." You can also export entities contained in the repository to a file on disk (this option is discussed in "Exporting an Entity to the File System" on page 4-34) and to a Java archive package file for re-import into another system (for more information on import/export, see Chapter 11, "Importing and Exporting Workflow Packages.")

Viewing the XML Entities in the Repository

To view the XML entities in the repository:

1. Choose Tools—Show XML Finder. The XML Finder dialog box opens with the Repository tab selected.
2. In the left pane, select XML Repository. This displays the XML entities in the repository.

Figure 4-11 XML Entities in the Repository



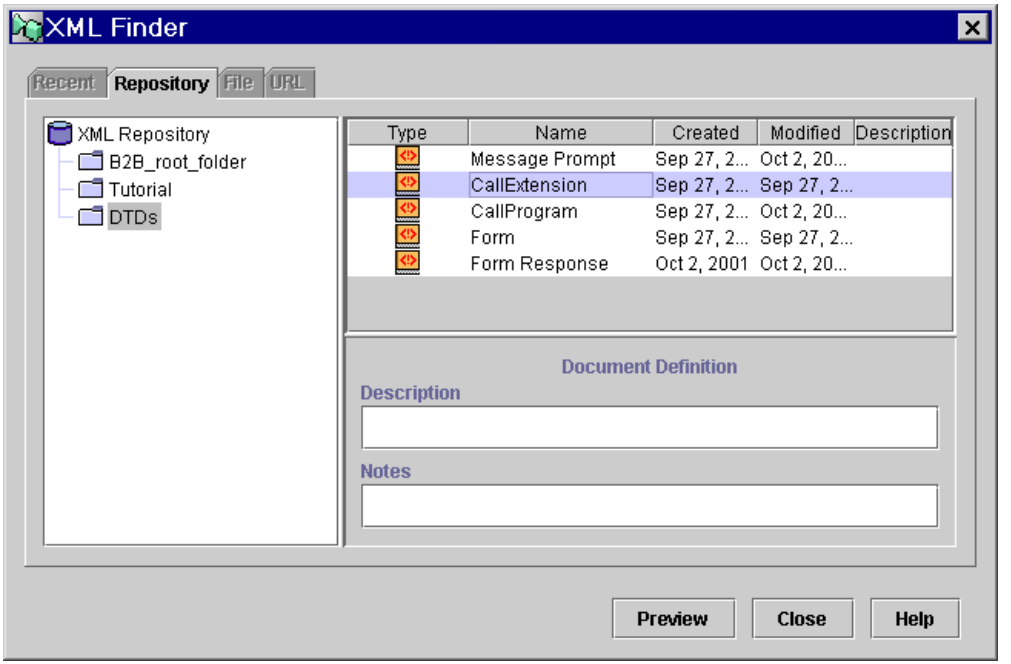
The following table describes information displayed in the Repository window.

Type	The type of entity, including folders, documents, and other XML entities (for information on XML entity types, see "Working with XML Entities" on page 4-28).
Name	The name of the folder or entity.
Created	The date the entity was first created in the repository.
Modified	The date the entity was last modified.
Description	The description of a folder or entity that was entered when the entity was created.
Folders	A list of all folders contained in the repository.
Entities	A list of all entities contained in the repository.

To view the contents of a folder:

1. In the left pane, expand all folders and select the folder whose contents you wish to view. A list of the entities contained in the folder appear in the right pane, along with any description or notes that were entered for the folder when it was created.

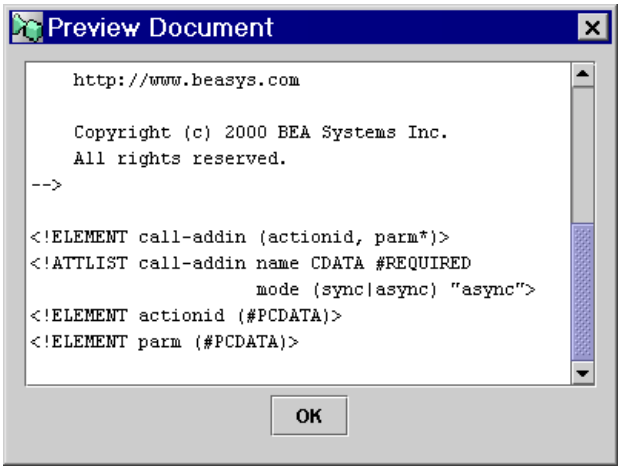
Figure 4-12 XML Entities in the Selected Folder



2. Select an entity in the list to display any description or notes that were entered for the entity when it was created.

3. Optionally, with an entity selected, click Preview to display the Preview Document window and view the document's content.

Figure 4-13 Preview Document Window



4. Click OK to close the Preview Document dialog box.

For more information about entities, see "Working with XML Entities" on page 4-28.

The left panel shows you a tree view of the repository with folders and sub-folders arranged hierarchically. The top-most panel on the right shows you the contents of the selected folder. The Description field contains a description of the selected folder, and the Notes field contains any notes about the selected folder.

Working with Folders

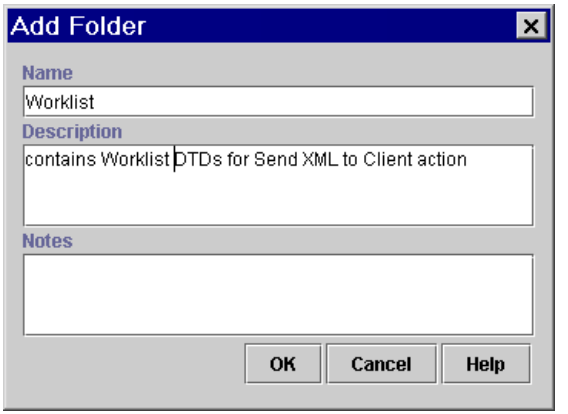
You can perform several different actions on a folder, including adding, updating, and deleting. To perform actions on a folder:

Adding a Folder

To add a folder:

1. In the left pane of the Repository window, right-click the XML Repository icon or any sub-folder, and from the pop-up menu, select Add Folder to display the Add Folder dialog box.

Figure 4-14 Add Folder Dialog Box



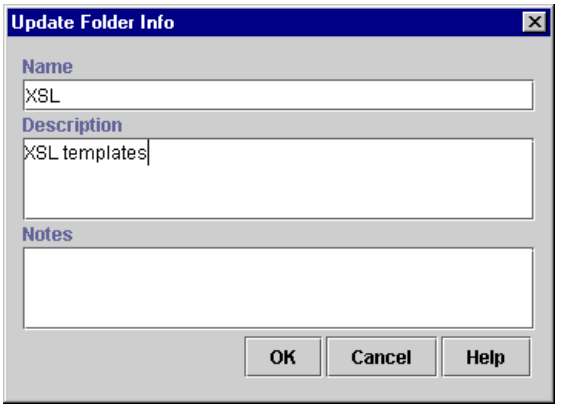
2. In the Name field, enter the name of the folder.
3. Optionally, enter a description and notes about the folder in the Description and Notes fields, respectively.
4. Click OK. The new folder appears in the XML Finder dialog box.

Updating Folder Information

To update a folder:

1. In the left pane of the Repository window, right-click the folder you want to update, and from the pop-up menu, select Update Folder Info to display the Update Folder Info dialog box.

Figure 4-15 Update Folder Info Dialog Box



2. Change the contents of the Name, Description, and Notes fields as necessary.
3. Click OK.

Deleting a Folder

A folder may only be deleted when it has no sub-folders.

To delete a folder:

1. In the left pane of the Repository window, right-click the folder you want to delete, and from the pop-up menu, select Delete Folder.
2. When prompted, confirm the deletion.

Working with XML Entities

The repository stores different types of XML entities, each one represented by a symbol, as shown in the following table.

Symbol	XML Entity Type
	Document Type Definition (DTD) file

Symbol	XML Entity Type
	Message Format Language (MFL) file
	Schema (XSD) file
	Text file
	XML document
	Extensible Stylesheet Language (XSL) template document

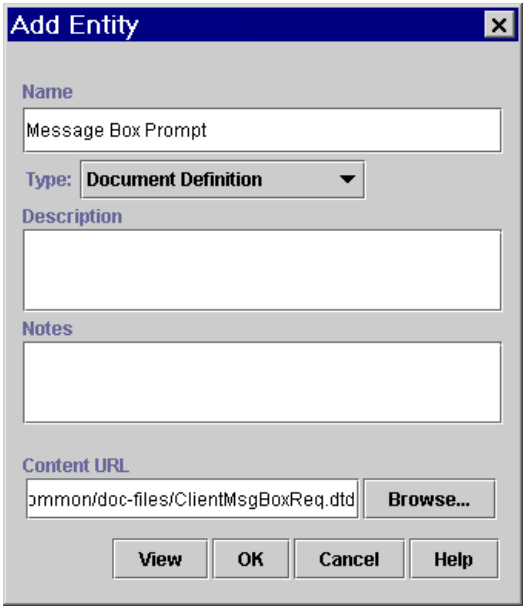
You can perform several different actions on XML entities in folders in the repository, including adding, updating, moving, and deleting entities.

Importing an XML Entity into the Repository

To add an XML entity:

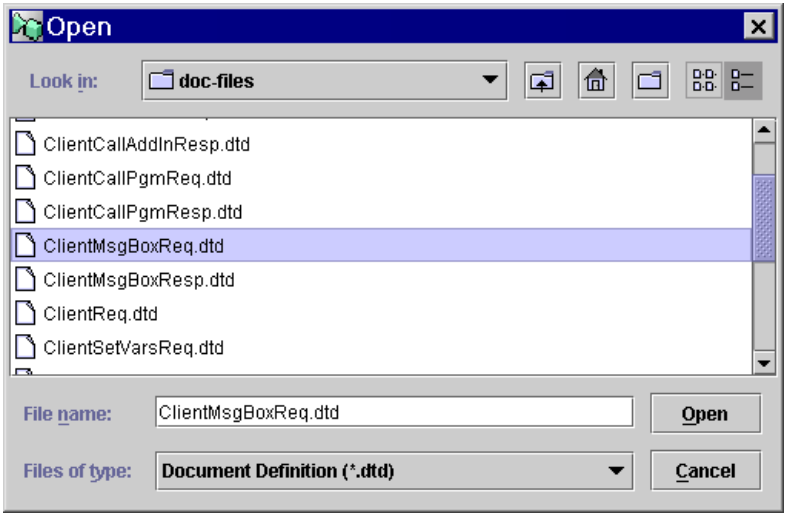
1. In the left pane of the Repository window, expand folders, right-click the folder to which you want to add the entity, and from the pop-up menu, select Add Entity to display the Add Entity dialog box.

Figure 4-16 Add Entity Dialog Box

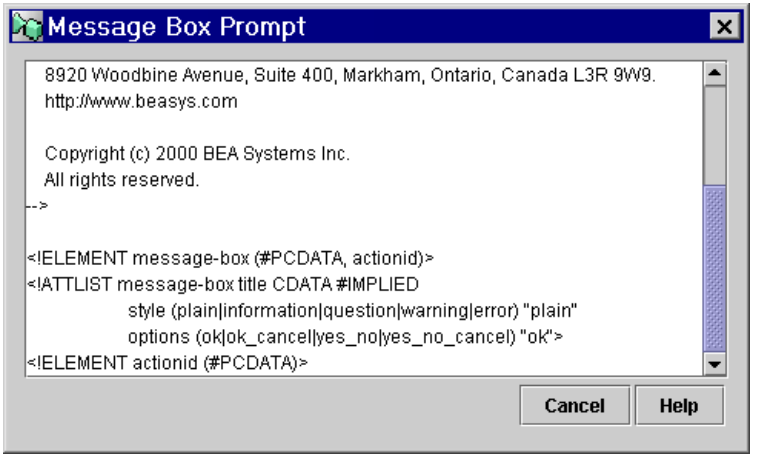


2. In the Name field, enter a unique name for the entity you are adding.
3. From the Type drop-down list, select the type of entity you are adding.
4. Optionally, enter a description and notes about the entity in the Description and Notes fields, respectively.
5. In the Content URL field, enter a URL for the entity you are adding, or use Browse to locate the entity on a local or mapped network drive. The Open dialog box appears.

Figure 4-17 Open Dialog Box



6. From the Look in drop-down list, select the folder containing the file whose contents you want to import.
7. In the File name field, enter the filename and extension, or select the file, and click Open.
8. The URL for the file is returned to the Add Entity dialog box.
9. Optionally, click View to display the contents of the entity you are adding.



10. Click Cancel to close the window and return the content to the Add Entity dialog box.
11. Click OK. The new entity appears in the XML Finder dialog box in the selected folder.

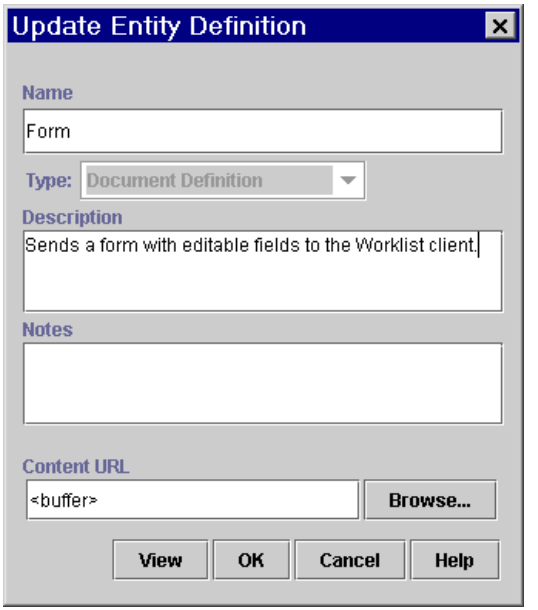
Updating an Entity

You can use the Update Entity feature to change the content of an entity you have defined. You cannot, however, change the entity type. To change the type of an entity, you must create a new entity with the desired content type. For details, see “Importing an XML Entity into the Repository” on page 4-29.

To update an entity:

1. In the left pane of the Repository window, expand folders, select the folder containing the entity you want to update.
2. In the right pane of the window, right-click the entity you want to update, and from the pop-up menu, select Update Entity Definition to display the Update Entity Definition dialog box.

Figure 4-18 Update Entity Definition Dialog Box



3. Change the contents of the Name, Description, and Notes fields as necessary.
4. In the Content URL field, enter the URL for the source of the new content you want to add, or use Browse to locate the entity on a local or mapped network drive.
5. Optionally, click View to display the new content for the entity.
6. Click OK to close the Update Entity Definition dialog box. The selected entity is updated.

Moving an Entity

You can move an entity from one folder to another by cutting it from the source folder and pasting it to the target folder.

To move an entity:

1. In the left pane of the Repository window, expand folders, and select the folder containing the entity you want to cut.

2. In the right pane of the window, right-click the entity you want to cut, and from the pop-up menu, select Cut.
3. In the left pane of the Repository window, expand folders, right-click the folder into which you want to paste the entity, and from the pop-up menu, select Paste. The entity is pasted into the target folder.

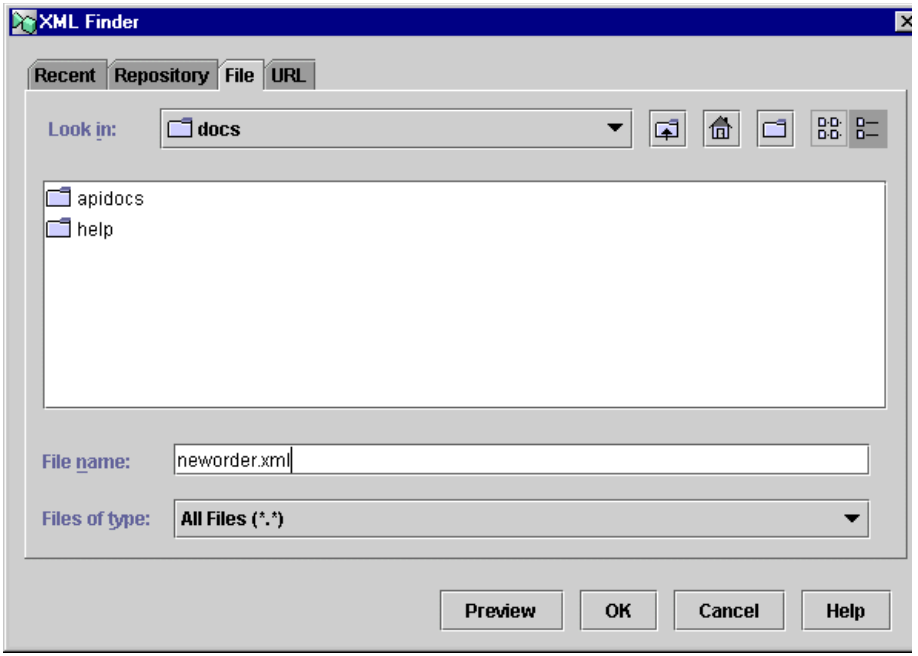
Exporting an Entity to the File System

You can save an XML entity from the repository database table to the local file system or a network drive mapped to the local machine.

To export an entity to a file:

1. In the left pane of the Repository window, expand the folders and select the folder containing the entity you want to export.
2. In the right pane of the window, right-click the entity you want to export, and from the pop-up menu, select Export Entity to display the Save dialog box.

Figure 4-19 Save Dialog Box



3. From the Look in drop-down list, select the drive and folder to which you want to export the entity.
4. In the File name field, specify a name for the file to which you want to export the entity, or select an existing file. If you do not specify a name, the system assigns the name of the entity by default. If you select an existing file, you are prompted to overwrite it.
5. Click Save. The file is saved to disk with the appropriate extension for the file type.

Deleting an Entity

If you delete an entity that is referenced by the XSL Transform action in a workflow, be sure to update this action in order to avoid WebLogic Integration server exceptions at run time (for more information on this action, see "Transforming XML Documents" on page 6-96).

To delete an entity:

1. In the left pane of the Repository window, expand the folders and select the folder containing the entity you want to delete.
2. In the right pane of the window, right-click the entity you want to delete, and from the pop-up menu, select Delete Entity.
3. When prompted, confirm the deletion.

CHAPTER

5 Defining Workflow Templates

This section discusses concepts and tasks pertaining to defining and maintaining workflow templates, template definitions, workflow variables and nodes. It includes:

- Overview of Template Definition Tasks
- Working with Templates
- Working with Template Definitions
- Working with Nodes
- Working with Variables
- Defining Node Properties
- Working with Exception Handlers

Overview of Template Definition Tasks

Defining a complete workflow template definition includes creating a template, creating a template definition, designing the flow, defining variables, specifying node properties, and, optionally, defining exception handlers. While defining workflows is an iterative process, which requires revision and refinement at each level, the following steps outline the recommended order in which you should perform these tasks when defining a new template definition:

Note: You may also want to familiarize yourself with the workflow expression language and the Studio's Expression Builder and XPath Wizard tools before beginning to define workflow templates. Many of the tasks described in this section, such as creating a label for a template definition, defining a condition for a Decision node, and defining events, require entering expressions into dialog box fields. Complete information on workflow expressions is available in Chapter 8, "Using Workflow Expressions."

1. Create a workflow template. Procedures are given in "Creating a Workflow Template" on page 5-4. Alternatively, import a template from a previously exported workflow package. Procedures are given in "Importing Workflow Packages" on page 11-5.

2. Within the template, create a template definition. Procedures are given in "Creating a Workflow Template" on page 5-4. Alternatively, import a template definition from a previously exported workflow package or XML file. Procedures are given in "Importing Workflow Packages" on page 11-5 and "Importing Workflow Template Definitions from XML" on page 11-11.

3. Create a high-level workflow by adding shapes and connections to the design area. Procedures are given in "Working with Nodes" on page 5-19.

4. Rename Task, Event, and Start node shapes so that their intended functions are easily recognizable. Procedures are given in "Renaming Nodes" on page 5-23.

5. Begin to create variables. Procedures are given in "Creating a Variable" on page 5-30.

6. Rename Decision nodes by defining conditions for them. Procedures for defining Decision properties are given in "Defining Decision Properties" on page 5-52.

7. Specify properties for Start nodes by defining the trigger type and properties, variable initializations, and, optionally, by configuring event keys for event-triggered starts and calendars for timed starts. For details about event keys, see "Configuring Event Keys" on page 4-18. For details about calendars, see "Administering Business Calendars" on page 3-4.

   Alternatively, you can import previously exported event keys and calendars from existing workflow packages. For details, see "Importing Workflow Packages" on page 11-5. For procedures for defining Start properties, see "Defining Event-Triggered Start Properties" on page 5-46.

8. Specify properties for Event nodes, and optionally configure event keys for the events. For details, see "Configuring Event Keys" on page 4-18.

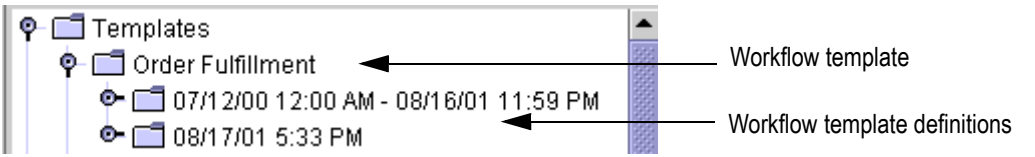
Alternatively, you can import previously exported event keys from existing workflow packages. For details, see “Importing Workflow Packages” on page 11-5. For procedures for defining Event node properties, see “Defining Event Properties” on page 5-49.

9. Specify Task node properties for manually assigned tasks. Explanations and procedures are given for task properties in “Defining Task Properties” on page 5-53.
10. Begin to add actions to Task nodes and other nodes, if necessary. Procedures for adding and defining actions are given in Chapter 6, “Defining Actions.”
11. Optionally, define exception handlers for the template definition and add actions to invoke them. Exception handlers are discussed in Chapter 9, “Handling Workflow Exceptions.”
12. Save the template definition. Procedures are given in “Saving and Closing a Template Definition” on page 5-12.
13. When you are ready to run the workflow, activate the template definition, as described in “Updating, Labeling, and Activating a Template Definition” on page 5-12, and save it.

Working with Templates

A workflow template is, in essence, a folder or a container for WebLogic Integration workflow template definitions. Each workflow template can hold one or more workflow template definitions. Workflow template definitions are identified in the folder tree by an Effective and Expiry date and time, as described in “Working with Template Definitions” on page 5-7.

Figure 5-1 Workflow Templates and Workflow Template Definitions



A template has a one-to-many relationship with organizations, which means that a template is unique within the system, but can be defined for multiple organizations. A template is visible in the folder tree for each organization for which it is defined, along with all its template definitions. Any changes that are made to a template definition in the folder tree of one organization, including deletions, automatically appear in the folder tree in all other organizations with which the template is associated.

When you import templates and template definitions using the Import/Export function, templates may be overwritten, but template definitions may not. If you import a template definition with the same dates as an existing one, the existing template definition will not be overwritten, but another one will be created. (For more information on importing templates, see "Importing Workflow Packages" on page 11-5).

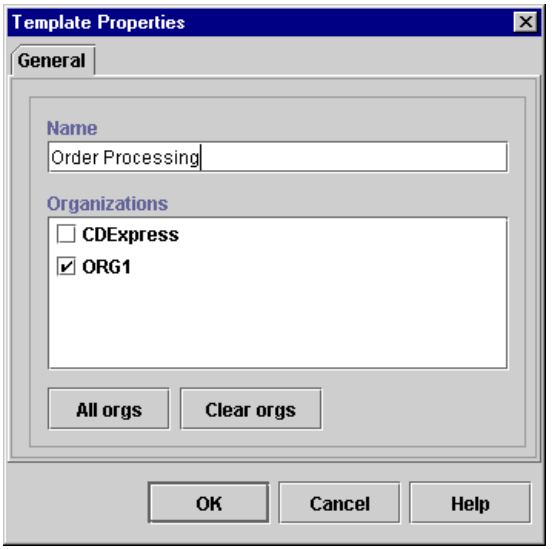
Creating a Workflow Template

Note: To create a template, you must have Create Template permission. For details about permission levels, see "Assigning Permissions to Users and Roles" on page 3-26.

To create a workflow template:

1. With any organization active, right-click on the Templates folder in the Studio main window.
2. Select Create Template from the pop-up menu to display the Template Properties dialog box.

Figure 5-2 Template Properties Dialog Box



3. In the Name field, enter a unique, meaningful name for the workflow template. All workflow template definitions defined within this workflow template use this name for identification, once they are placed into run time and become workflow instances.

4. In the Organizations section of the dialog box, select the organization (or organizations) with which you want to associate this workflow template. To make the workflow template available to all organizations, click All Orgs; to clear organizations and reset them, click Clear Orgs.

   Note: If you associate a template with multiple organizations, any changes you make to the template are automatically reflected when viewing the template from different organizations.

5. Click OK. The new workflow template appears under the Templates folder in the folder tree.

Updating Template Properties

You can assign a template to additional organizations after it has been created or imported (for procedures, see "Importing Workflow Packages" on page 11-5), by using the Template Properties dialog box.

To update template properties:

1. From the Organization field above the folder tree, select an organization with which the template which you want to update is defined.
2. In the folder tree, expand the Templates folder, right-click the template, and from the pop-up menu, select Properties to display the Template Properties dialog box.
3. Make any additional changes to the organizations associated with this template.
4. Click OK to save your changes and exit the dialog box.

Deleting a Template

When you delete a workflow template, the following occurs:

- The template is removed from all organizations associated with it.
- All template definitions contained in the template are deleted.
- All instances of the template's definitions are deleted (including all tasks regardless of their current status).
- All history relating to the template's definitions is deleted and is, therefore, no longer available in statistical or workload reports.

Note: To delete a template, you must have Delete Template permission. For details about permission levels, see "Assigning Permissions to Users and Roles" on page 3-26.

To delete a workflow template:

1. If the template contains open template definitions, close them by clicking the "X" in the upper right corner of the drawing window, or right-clicking on the template definition in the folder tree, and selecting Close from the pop-up menu.

2. Right-click on the workflow template name in the folder tree.
3. From the pop-up menu, select Delete.
4. When prompted by the Delete Template warning message, click Yes to delete the workflow template and all of its workflow template definitions, or click No to cancel the delete.

   If there are instances of the template, you are prompted to delete the instances also. Click Yes to delete the instances, or click No to cancel the delete.

Working with Template Definitions

In the folder tree, a workflow template definition name consists of its effective and expiry dates and times. The effective and expiry dates and times represent the range of time within which the workflow template definition is available for *instantiation*, or run-time execution. Note that you may have multiple template definitions with the same effective and expiry dates.

To make a template definition available for instantiation, you must *activate* it first. At design time, you can activate as many template definitions as you want, with the only restriction being that you cannot activate template definitions with the same *effective* (starting) date.

The advantage of the effective and expiry feature is that it allows you to make workflow template definitions valid for exact periods of time, without having to manually inactivate and activate different template definitions over the course of the calendar year. For example, perhaps you have a cyclical business, which requires you to run a particular workflow template definition from January to March, followed by a second workflow template definition having different tasks or variables from April to June, followed by a third defined for July to September. Rather than having to manually inactivate one template definition and activate another for each quarterly period, you simply specify different effective and expiry dates, mark them all as active, and the server automatically selects the correct version when the workflow is instantiated.

On the other hand, although you can have multiple active template definitions, only one template definition can actually be instantiated at run time, which means that you should take care to design your effective and expiry dates in a *rolling* fashion, and

avoid overlapping dates. If you have active template definitions with overlapping dates, for example, one from January to March, and another from February to April, the process engine will simply pick up the first one that it finds in the database, which is usually the first one to have been created at design time.

Note: If you want to specify different flows according to different business conditions, you should use multiple start nodes in the same template definition. For more information, see "Defining Start Properties" on page 5-33.

Creating a Workflow Template Definition

When you create a template definition, you specify its effective and expiry dates.

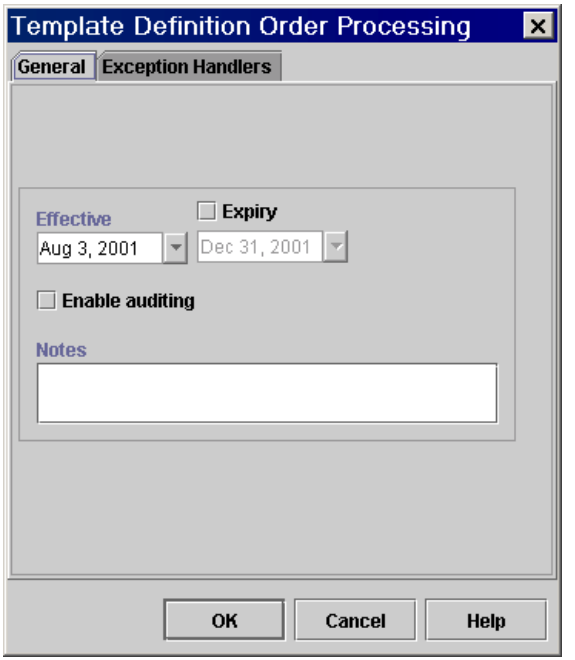
You can also enable auditing, which enables logging of run-time workflow information, such as client access and execution times, and allows you to make custom entries at points throughout a workflow by using the Make Audit Entry action (for more information, see "Making an Audit Entry" on page 6-42). The audit information is posted in an XML message to the default JMS audit topic, `com.bea.wlpi.AuditTopic`, and is written to a text file, `myserver.log`, located in the `logs` directory of the active WebLogic Integration domain on the server.

Note: To create a template definition, you must have Create Template permission. For details about permission levels, see "Assigning Permissions to Users and Roles" on page 3-26.

To create a workflow template definition:

1. From the Organization field above the folder tree, select an organization with which the template is defined to which you want to add a template definition.
2. In the folder tree, expand the Templates folder, right-click the template, and select Create Template Definition from the pop-up menu to display the Template Definition dialog box, which appears with the name of the template to which the template definition belongs.

Figure 5-3 Template Definition Dialog Box



3. On the General tab, specify the following:

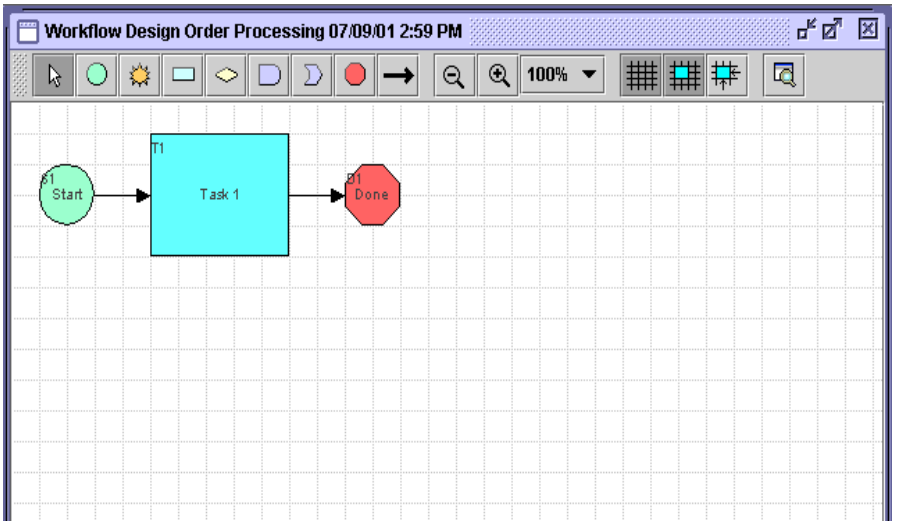
   - Effective — select the exact date when this workflow template definition will become effective. The time is automatically supplied by the system, and corresponds to the time the template definition is first created.
   - Expiry — optionally select this check box to set an expiry date for the workflow template definition. The time is automatically supplied by the system, and is always 11:59 PM of the expiry date.

   Note: If the Expiry option is not selected, the workflow template definition will always be valid and effective.

4. Optionally, select Enable auditing, which allows run-time workflow information to be logged.
5. Optionally, enter a note in the Notes field to record a general comment describing the template definition.
6. Click OK to display the workflow design area.

In the workflow design area, a default workflow template definition is presented along with a toolbar containing drawing shapes used for defining the workflow template. The default workflow template definition contains three shapes: Start, Task, and Done.

Figure 5-4 Workflow Design Area



You can begin to define the template definition by adding nodes and connections, as described in "Working with Nodes" on page 5-19 or by adding variables, as described in "Working with Variables" on page 5-28.

Opening an Existing Template Definition

Although you can view read-only properties for a template definition by right-clicking any of its folders in the folder tree and selecting Properties from the pop-up menu, you cannot modify a template definition unless it is opened. Once you open a definition, it is locked and can only be opened in read-only mode by another user.

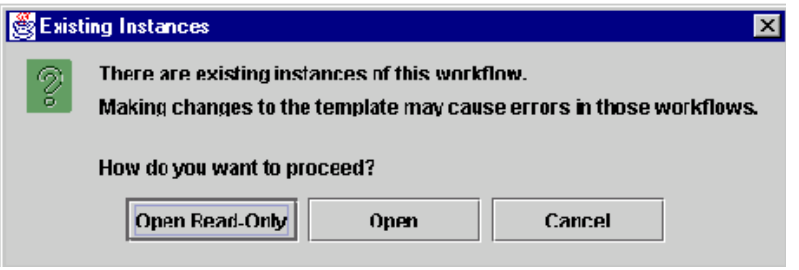
Note: To open a template definition, you must have Create Template permission. For details about permission levels, see "Assigning Permissions to Users and Roles" on page 3-26.

To open an existing workflow template definition:

1. From the Organization field above the folder tree, select an organization with which the template definition which you want to open is defined.
2. In the folder tree, expand the Templates folder, expand the template folder containing the desired template definition, right-click the template definition, and select Open from the pop-up menu.

   If there are existing instances of the selected workflow template definition, meaning that the selected workflow template definition has been started and an instance of the template definition exists in the server, a warning message will appear, as in the following figure.

Figure 5-5 Existing Instances Dialog Box



3. Select from the following options:
   - Open Read-Only — opens the selected workflow template definition for viewing purposes only. All user interface options relating to making changes to workflows are not accessible, and you cannot modify the workflow.
   - Open — opens the selected workflow template definition for viewing and updating.
   - Cancel — closes this dialog box and does not open the workflow template definition.

Note: You should not try to modify a template definition that has running instances, as this can cause unpredictable exceptions in those instances. If you want to make changes to such a template definition, you should do the following:

- In a production environment, first allow all running instances to complete, and then create and activate a new template definition with different effective and expiry dates. To copy existing workflow information to the new template definition, follow the procedure in "Copying a Workflow Template Definition" on page 5-14.

- In a development environment, delete all instances of the template definition before making changes to it. For procedures, see "Deleting Workflow Instances" on page 10-11.

Saving and Closing a Template Definition

In the Studio folder tree, an asterisk (*) appears to the left of each workflow template definition that needs to be saved.

To save a template definition, do one of the following:

- In the folder tree, right-click the template definition and select Save from the pop-up menu.
- Anywhere in the workflow design area, right-click and select Save from the pop-up menu.

When you close a template definition, it is unlocked and available for use by another user.

To close a workflow template definition, do one of the following:

- In the folder tree, right-click the template definition and select Close from the pop-up menu.
- In the upper right corner of the workflow design window for the template definition, click the "X".

Updating, Labeling, and Activating a Template Definition

After you have created and opened a template definition, you can update the properties you specified when you created it (as described in "Creating a Workflow Template Definition" on page 5-8), add a label to it, and activate it.

The workflow label you create here is displayed to a Worklist user in the Workflow Label column of the task list at run time. It also appears in the Workflow Label field of the Workflow Instances dialog box in the Studio at run time (for more information, see "Viewing Workflow Instance Status" on page 10-5). It is used to help identify the instance of the workflow, and could include information such as date and time, invoice

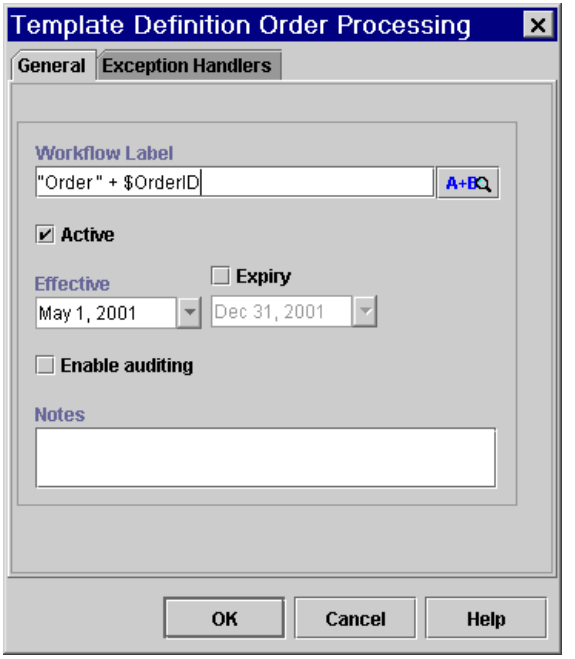
number, customer name, or other relevant information that differentiates it from others. The label is formulated in workflow expression language, and can include constants, variables, operators, and other expression components. For more information about workflow expression features and syntax, see Chapter 8, "Using Workflow Expressions."

When you create a new template definition, it defaults to inactive status. Often template definitions that are currently in development are inactive to prevent premature invocation. However, before a workflow can be instantiated, or placed into the run-time environment, you must activate one of its template definitions.

To update the properties of a workflow template definition:

1. With the workflow template definition open, right-click anywhere in the design window, or right-click the template definition in the folder tree, and choose Properties from the pop-up menu to display the Template Definition dialog box.

Figure 5-6 Updating Template Definition Properties



2. In the General tab of the Template Definition dialog box, in the Workflow Label field, enter an expression that will be evaluated at run time to generate the label. For information on constructing workflow expressions, see Chapter 8, "Using Workflow Expressions."
3. To activate the template definition, select the Active check box.
4. Optionally, update the Effective and Expiry dates, and enable or disable auditing.
5. Optionally, in the Exception Handlers tab of the Template Definition dialog box, add, update, or delete any Exception Handlers as necessary. For detailed information on exception handling, see Chapter 8, "Using Workflow Expressions."
6. Click OK to save changes and exit the dialog box.

Copying a Workflow Template Definition

The Studio allows you to copy an existing workflow template definition within the same template. When this is done, all of the assigned workflow template definition properties are copied as well. This saves you time when you must create more than one workflow template definition with similar properties. The properties of the new workflow template definition can be modified as needed.

Note: The copied (new) workflow template definition will not be marked active. If you want to make it available for instantiation, you must activate it first. See "Updating, Labeling, and Activating a Template Definition" on page 5-12 for details.

To copy an existing workflow template definition:

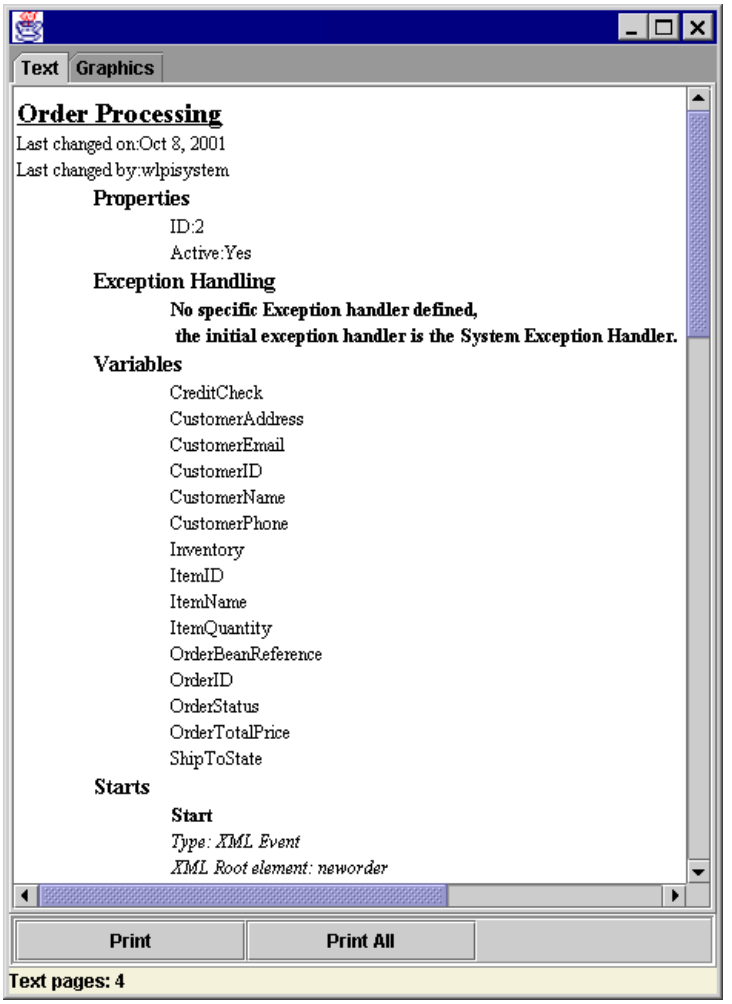
1. Right-click the workflow template definition to be copied, and choose Copy from the pop-up menu. A copy of the selected workflow template definition is instantly pasted as the last workflow template definition in the folder tree with the same effective and expiry dates as the original workflow template definition, and is opened in the design area.
2. To rename the template definition or change its properties, follow the procedure in "Updating, Labeling, and Activating a Template Definition" on page 5-12.

Printing a Template Definition

You can print workflow template definition diagrams from the Studio. There are two methods for invoking the print facility.

- Right-click the workflow template definition in the folder tree of the Studio and select Print from the pop-up menu.
- Right-click anywhere in the workflow template definition diagram in the drawing area and select Print from the pop-up menu.

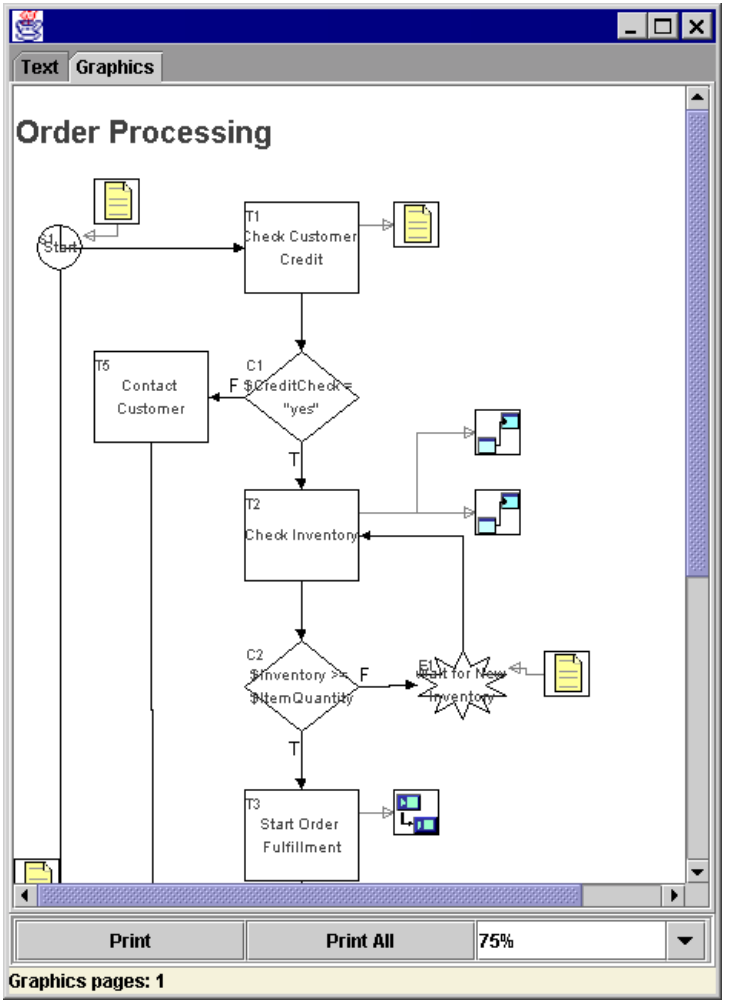
Figure 5-7 Print Workflow Text



The Text tab contains information relevant to each workflow node, including action and node notes, as well as details on actions within tasks, events, decisions and done nodes, including sub-actions within actions.

The Graphics tab contains a diagram of the workflow template definition.

Figure 5-8 Print Workflow Graphics



Click Print to print the information on the selected tab, or click Print All to print both the information contained in the Text tab and the diagram contained in the Graphics tab.

In the Print dialog box, select the appropriate settings to print the workflow.

Deleting a Template Definition

When you delete a workflow template definition, the following occurs:

- The template definition is removed from all organizations associated with the template that contains the template definition.
- All instances of that workflow template definition are deleted (including all tasks regardless of their current status).
- All history relating to that workflow template definition is deleted and is therefore no longer available in statistical reports.
- Workload reports that refer specifically to the deleted workflow template definition are changed to refer instead to all remaining workflow template definitions.

Note: To delete a template definition, you must have Delete Template permission. For details about permission levels, see "Assigning Permissions to Users and Roles" on page 3-26.

To delete a workflow template definition:

1. If the template definition is open, close it by clicking the "X" in the upper right corner of the drawing window, or right-clicking on the template definition in the folder tree, and selecting Close from the pop-up menu.
2. Right-click on the workflow template definition in the folder tree.
3. From the pop-up menu, select Delete.
4. When prompted by the Delete Workflow warning message, click Yes to delete the workflow template definition, or click No to cancel the delete.

   If there are instances of the template definition, you are prompted to delete the instances also. Click Yes to delete the instances, or click No to cancel the delete.

Working with Nodes

After you first create a workflow template definition, by default, the design area contains three shapes (nodes): Start, Task, and Done. The start node is set to a manual start by default and the task node assigns the task to the Worklist user who initiates the workflow. These are the minimal properties required to create a workflow that can be run. (For more information on editing Start node properties, see "Defining Event-Triggered Start Properties" on page 5-46, and for more information on editing Task node properties, see "Defining Task Properties" on page 5-53.)

The following table lists workflow shapes, their node name, and purpose.

Table 5-1 Workflow Shapes and Connections

Symbol	Node Type	Purpose
	Start	Indicates the start of the workflow, which can be triggered by different means: manually, at a specific time, by an event, or by another workflow. For more information about Start nodes, see "Defining Start Properties" on page 5-33.
	Event	Represents an event that can be triggered by an XML message received on an internal JMS queue from an external application or from another workflow, or by a plug-in-defined event. For more information about Event nodes, see "Defining Event Properties" on page 5-49.
	Task	Represents a node in which various actions can be defined. Also defines a user-assigned task. For more information about Task nodes, see "Defining Task Properties" on page 5-53.
	Decision	Represents a condition in the workflow that evaluates to True or False. True and False results branch into different workflow paths. For more information, see "Defining Decision Properties" on page 5-52.
	And Join	Merges two separate paths with an AND gate. Both paths must have finished executing before the flow can proceed. You can also change an AND to an OR join after you have added it to the workflow; for more information, see "Defining Join Properties" on page 5-57.

Table 5-1 Workflow Shapes and Connections

Symbol	Node Type	Purpose
	Or Join	Merges two separate paths with an OR gate. Only one path must have finished executing before the flow can proceed. Once control has passed from a single path to the nodes succeeding the Join, all preceding unexecuted tasks are not executed. You can also change an OR to an AND join after you have added it to the workflow; for more information, see "Defining Join Properties" on page 5-57.
	Done	Indicates the end of the workflow. For more information, see "Defining Done Properties" on page 5-58.
	Connection	Used to connect workflow nodes. The arrow indicates the next node to be executed in the flow.

Adding, Arranging, and Connecting Nodes

To manipulate shapes in the design area, you can do the following:

- Place a shape in the design area by clicking the shape on the toolbar, placing your cursor on the design area, and then clicking again to drop the shape onto the design area. A node is also created in the folder tree after you have placed a shape in the design area.
- Move a shape within the design area by clicking and dragging the shape with your mouse.
- Connect shapes by clicking the Draw Connection button in the toolbar, then clicking on the source node, dragging to the target node, and releasing the mouse button. When you create a connection from a Decision shape, you are prompted by the Create Connection dialog box to specify whether the connection is True or False.

Note: For more information about the toolbar, see "Using the Toolbar" on page 2-11.

Once you have added a node to the design area, it immediately appears in the folder tree as well. Nodes are given default names, with a number indicating the order in which you placed the shape in the design area, such as T1, T2, T3, etc. for Task nodes. To rename a node and edit its properties, see "Working with Node Properties" on page 5-23.

Deleting a Node or Connection

To delete a node:

1. Do one of the following:
   - In the design area, right-click the node you want to delete, and select Delete from the pop-up menu.
   - In the folder tree, expand the folder that contains the node you want to delete, right-click the node, and from the pop-up menu, select Delete.
2. When prompted by a warning message, click OK to confirm the deletion, or Cancel to cancel.

To delete a connection:

1. Right-click a connection and select Delete from the pop-up menu.
2. Confirm the deletion when prompted. All connections that have been made to and from the node are also deleted.

Workflow Design Guidelines and Tips

As you are adding, connecting and arranging shapes, you may wish to keep in mind the following design guidelines:

- A workflow must include at least one Start node, and it can also include multiple Start nodes to start different paths within the same template definition.
- If a workflow does not contain a Done node, it never terminates.

- You can add multiple Done nodes to end different paths in the flow, but as soon as the first Done node is reached at run time, the workflow terminates, regardless of whether other paths have finished executing.
- You should ideally create one Task node for each major activity to be performed by an action, to keep your logic as graphically visible as possible. For more information on the relationship between actions and tasks, see Chapter 6, "Defining Actions."
- To split a flow into multiple paths, connect a single node to multiple nodes.
- To merge multiple paths back into a single flow, connect multiple nodes to a single Join node. Use an AND join to ensure that all paths are executed before the merge. Use an OR join to ensure that only one path is executed before the merge.
- BPM does not process multiple paths in parallel. Nodes are processed in the order that they are created.
- To create loops in the workflow, follow these guidelines:
  - To create a loop, connect a node in a backwards fashion to a previous node in the flow, and include a Decision node in the loop to test the current value of a counter. If you do not implement a counter or decision, the result will be an infinite loop.
  - To create a loop to the same node, you cannot use the Draw Connection button, but must use the Next tab in a node's properties dialog box to specify the successor node as the current one. For information, see "Specifying or Updating Successor Nodes" on page 5-23. To evaluate the value of a counter in this case, you would need to use the Evaluate Condition action embedded in the node. (For more information on this action, see "Embedding a Conditional Sequence" on page 6-41.)
- For very complex workflows, you may want to divide the flow into several workflows that can call each other. For more information on using sub-flows, see "Calling a Sub-Workflow" on page 6-36. For information on using XML messaging to communicate between workflows, see "Defining Event And Event-Triggered Start Properties" on page 5-38 and "Posting an XML Message to a JMS Topic or Queue" on page 6-82.

For additional design guidelines, see *Best Practices in Designing BPM Workflows*.

Working with Node Properties

Once you have placed node shapes into the workflow design area, you can begin specifying their properties. One of the first tasks you will want to do, for example, is to rename nodes to give them an easily recognizable view of the function they are to perform in the workflow. The following sections provide explanations and procedures for working with properties that are common to all types of nodes. For properties that are specific to each type of node, refer to the sections that discuss each node type.

To access node properties, do one of the following:

- In the design area, double-click the node.
- In the design area, right-click the node and select Properties from the pop-up menu.
- In the folder tree, expand the folder for the node type, right-click the desired node, and select Properties from the pop-up menu.

Renaming Nodes

You can provide a meaningful name for all node types, except Joins and Dones. Also, Decisions require that you enter a conditional expression to identify them. For information, see "Defining Decision Properties" on page 5-52.

To rename a node:

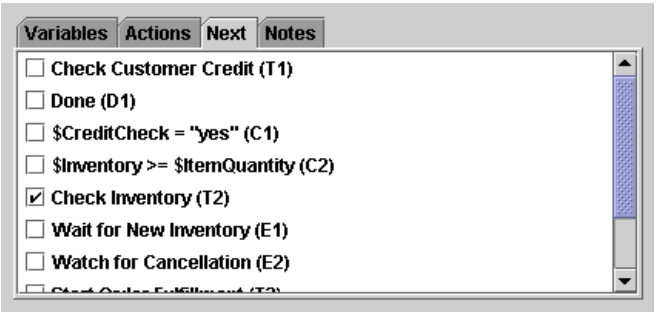
1. Display the properties dialog box for an Event, Start, or Task node.
2. In the Name field, enter a meaningful name which describes the action it will perform, such as `Check Inventory`.
3. Click OK to save your changes.

Specifying or Updating Successor Nodes

All node properties dialog boxes contain a Next tab which displays a list of all nodes in the current workflow. The next node in the flow is indicated by a check next to the node name. You can use this tab to specify the successor node (or nodes) to the current

node or change the successor nodes defined in the design area. Selecting or deselecting check boxes on this tab automatically redraws the connection lines drawn in the design area.

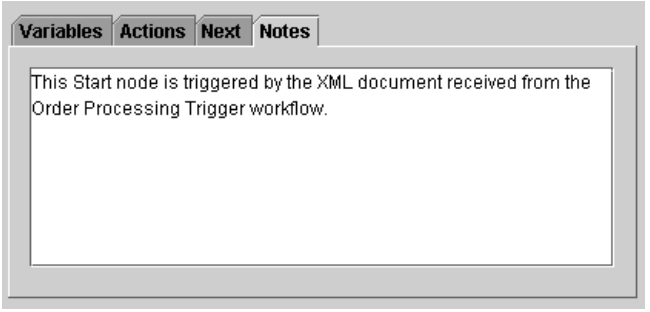
Figure 5-9 Next Tab of Node Properties Dialog Boxes



Adding Notes to a Node

All node properties dialog boxes contain a Notes text box that you can use to enter a comment about the node or actions contained in it. This is helpful for other users who access the same workflow and need to understand the workflow logic or design.

Figure 5-10 Notes Tab of Node Properties Dialog Box

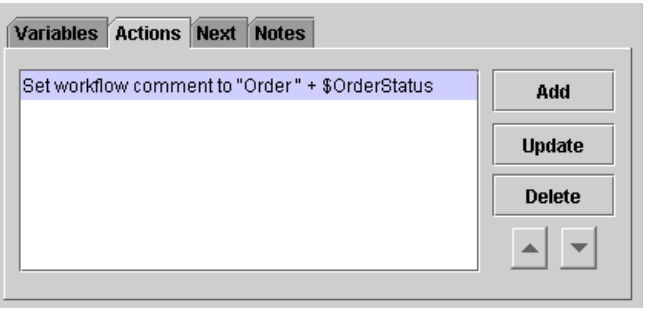


Additionally, Decision nodes and Task nodes also contain an Action Notes tab that lets you view notes defined for an action (see “Adding Notes to an Action” on page 6-21). Select an action in the left pane of the Task Properties or Decision Properties dialog box and the note that has been entered for it in the action definition is displayed.

Adding, Updating, Reordering, and Deleting Workflow Actions

All nodes, with the exception of Joins (AND and OR), allow you to add, update, reorder and delete actions within their properties dialog boxes.

Figure 5-11 Actions Tab of Node Properties Dialog Boxes



Actions define the operations that you want to perform when the node is activated. Actions specified on the Actions tabs of node properties dialog boxes are performed before the workflow proceeds to the next node (and the actions contained in it).

While Task nodes require that you add actions to them, adding actions to other nodes is optional and not recommended, since in many cases the same logic can be implemented by adding successor Task nodes. Complete information for adding, updating, deleting re-ordering, and defining actions is provided in "Working with Actions" on page 6-17.

Copying Nodes

You can copy a shape representing a node within a workflow template definition and paste it into the current workflow template definition or into another open workflow template definition. Since any actions and properties that have been defined within the node are also copied, you can use the copying function to create reusable design patterns which may only require minor modifications.

Note: If you are copying nodes between template definitions, be sure that any variables referenced by the node and its actions have been created in the target template definition, and that other referenced objects, such as roles, users and business calendars, are defined for the organization with which the template is associated.

Any properties and actions that have been defined within the node are also copied.

To copy a node and its properties within and between template definitions:

1. Do one of the following:
   - In the design area, right-click the node that you want to copy, and from the pop-up menu, select Copy.
   - In the folder tree, expand the folder which contains the node you want to copy, right-click the node you want to copy, and, from popup menu, select Copy.
2. Do one of the following:
   - Place your cursor in the design area of the target template definition, right-click and from the pop-up menu that appears, select Paste.
   - In the folder tree, right-click the folder for the appropriate node type, and from the pop-up menu, select Paste.

   The copied node appears in the design area and in the folder tree. The properties dialog box for the action is displayed, with all settings copied from the source action.
3. Modify the properties of the node as needed.

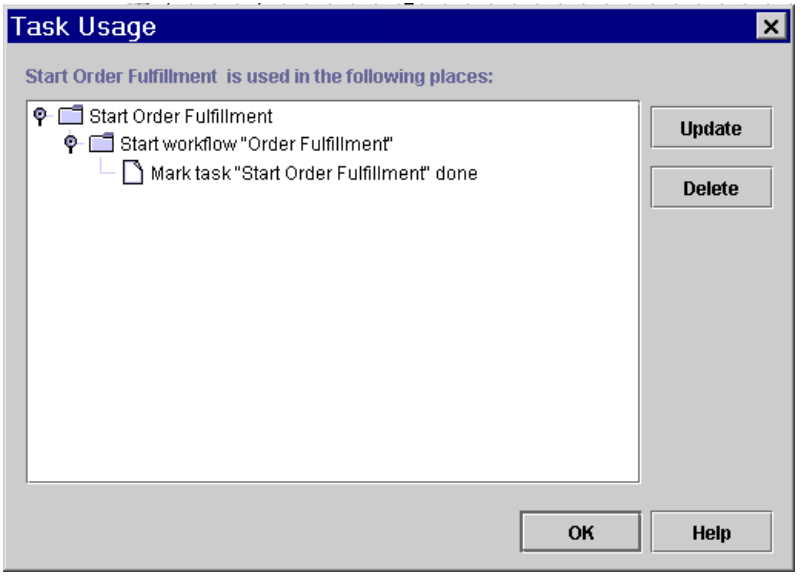
Viewing Task and Event Usage

You can view the different places where Event or Task nodes may be referenced, for example, by Task actions or the Cancel Workflow Event action. For more information, see “Action Categories” on page 6-2.

To see where a Task or Event node is used:

1. Right-click on the desired Event or Task node in the design area or in the folder tree, and from the pop-up menu, select Usage to display the Task or Event Usage dialog box.

Figure 5-12 Task Usage Dialog Box



2. Expand the folders until you see the item that references the selected node.
3. Optionally, use the following buttons in the dialog box to do the following:
   - Update — select to open the dialog box for the selected object that references the node.
   - Delete — select to delete the selected object.
4. Click OK to close the Task or Event Usage dialog box.

Working with Variables

Each workflow template definition can have a set of variables associated with it. Variables can hold values that are returned by business operations, that are extracted from XML documents, or that are set explicitly by workflow actions. Variables can also be used by the workflow for several other purposes, such as to evaluate a condition in a decision node, or to store the result of a response from the Worklist client application.

Not all workflow template definitions require variables. However, for those workflow template definitions containing processes that require variables, you can define these before you begin defining other workflow components such as node properties and actions, or add variables during the design process.

Workflow variables have a workflow-global scope. That is, a single variable is shared by all objects within a workflow template definition instance. Variables are, therefore, defined at the workflow template definition level in the folder tree and are referred to as workflow variables.

Variables can be of the following types:

Table 5-2 Workflow Variable Types and Initial Values

Variable Type	Contains . . .	Initialized to . . .
Boolean	Boolean value True or False	false
Date	Java date object	current date
Double	Double-precision floating-point number	0.0
Entity EJB	Reference to an Entity EJB called by the Perform Business Operation action (see “Calling a Business Operation” on page 6-78)	null
Integer	Long type integer	0
Java Object	Reference to a Java class called by the Perform Business Operation action (see “Calling a Business Operation” on page 6-78)	null

Table 5-2 Workflow Variable Types and Initial Values

Variable Type	Contains . . .	Initialized to . . .
Session EJB	Reference to a Session EJB called by the Perform Business Operation action (see "Calling a Business Operation" on page 6-78)	null
String	Character string	" " (empty string)
XML	XML document (for information, see "Setting a Variable Value" on page 6-21)	null

Note: If a plug-in is defined for variable types, additional variable types may be available.

Note: Initial values listed in the table above are determined by a setting in the server startup script. You can change initial values for all data types to be NULL by modifying this setting. For more information, see "Configuring BPM to Support Null Variables" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

In addition, if the workflow is to be called by another workflow (for more information, see "Defining Start Properties" on page 5-33), you need to specify whether the variable is to serve as an *input* or *output* parameter. An input parameter indicates that the variable is to receive its value from the calling, or parent, workflow. An output parameter indicates that the variable contains a value that will be passed back to the calling workflow.

Additionally, for input parameters, you can specify whether or not they are *mandatory*. If the input parameter is mandatory, the workflow does not start until the value for the variable is received from the calling, or parent, workflow. If the value is not received, the workflow does not start, and an exception is thrown.

For details about passing parameters to called workflows, see "Calling a Sub-Workflow" on page 6-36.

To set the initial value of a variable, you must use the Set Workflow Variable action inside a node (for information, see "Setting a Variable Value" on page 6-21), or use the Variables tab of a Start or Event node, an Exception Handler, or the Send XML to Client action (for information, see "Initializing Variables from Event Data" on page 5-45).

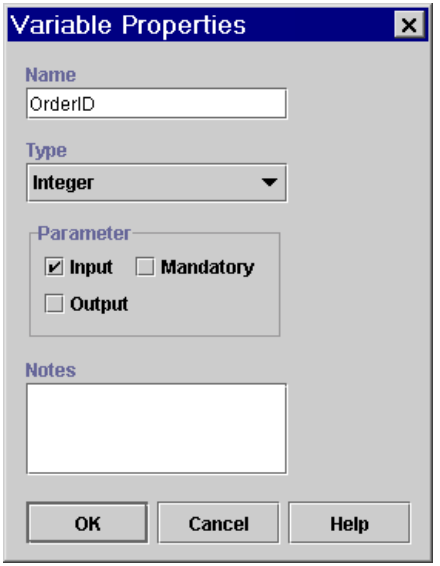
When a variable is used in a workflow expression, the variable name is preceded by the dollar sign ($) or colon (:) or other characters. For more information on variable notation, see "Using Variables" on page 8-4.

Creating a Variable

To create a variable:

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

Figure 5-13 Variable Properties Dialog Box



2. In the Name field, enter a meaningful name for the variable, such as `OrderID`.

   Note: Variable names cannot contain spaces.

3. From the Type drop-down list, select a variable type, as listed in Table 5-2.

4. Optionally, if the workflow is a called workflow, specify a Parameter, and whether it is Input or Output. For Input parameters, specify whether the parameter is mandatory.

   The parameter is used by a calling workflow to pass values to and receive values from a called subworkflow. The input parameter contains the values passed to the subworkflow, and the output parameter contains the return values from the subworkflow.

   Note: When instantiating the workflow programmatically, you can set the variables that are specified as input variables only. For more information about instantiating the workflow programmatically, see "Manually Starting Workflows" in *Programming BPM Client Applications*.

   Once the workflow has been instantiated, you can set the value of any variable, including input and output variables, as described in "Monitoring Run-time Variables" in *Programming BPM Client Applications*.

   For details about starting another workflow, see "Calling a Sub-Workflow" on page 6-36.

5. Optionally, in the Notes text box, enter a note about the variable.
6. Click OK to save the variable definition. The new variable appears in the folder tree under the Variables folder.

Updating a Variable

To update an existing variable:

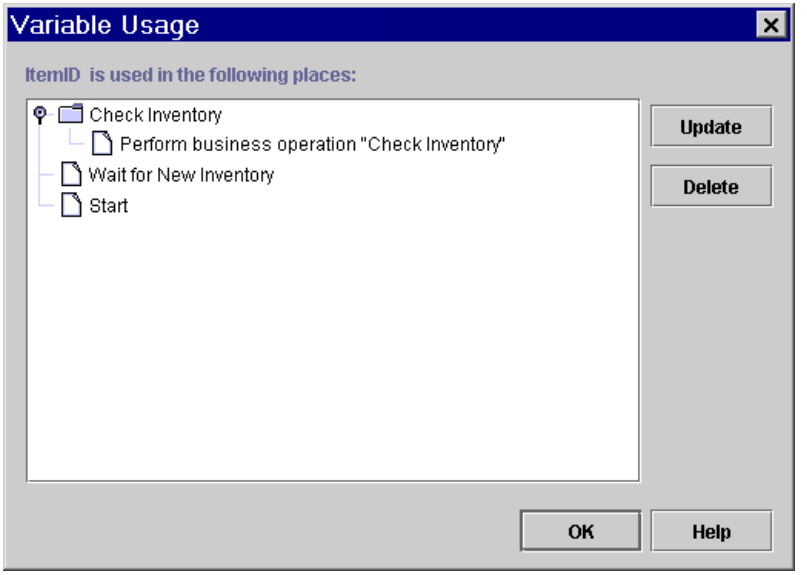
1. Right-click an existing variable in the folder tree and choose Properties from the pop-up menu. The Variable Properties dialog box is displayed.
2. Make changes to the variable as needed, and click OK.

Viewing Variable Usage

To see where a variable is used within the workflow:

1. In the folder tree, right-click the variable and select Usage from the pop-up menu to display the Variable Usage dialog box, which lists the places within the workflow where the selected variable is used or a value is assigned to it.

Figure 5-14 Variable Usage Dialog Box



2. Expand the folders until you see the item that references the selected variable.
3. Optionally, use the following buttons in the dialog box to do the following:
   - Update — select to open the dialog box for the selected object that references the variable.
   - Delete — select to delete the selected object.
4. Click OK to close the Variable Usage dialog box.

Deleting a Variable

You can only delete variables that have not yet been referenced in any workflow nodes, actions, or expressions. To view the places where a referenced variable is used, follow the procedure in “Viewing Variable Usage” on page 5-31.

To delete a variable:

1. In the folder tree, expand the Variables folder, right-click the variable you want to delete, and from the pop-up menu, select Delete.
2. When prompted by a warning, click OK to confirm, or Cancel to cancel the deletion.

Defining Node Properties

This section describes how to define node properties, according to each specific type of node:

- Defining Start Properties
- Defining Event And Event-Triggered Start Properties
- Defining Decision Properties
- Defining Task Properties
- Defining Join Properties
- Defining Done Properties

Defining Start Properties

Every workflow has at least one start shape that indicates the beginning of the workflow. The first node after the start will be the first activated node of the workflow, which can be a Task, Decision, or Event node.

Note: If no Start node is specified, no node activation can occur in the workflow.

Start nodes can have four types of triggers:

- Manual — the workflow is started manually by an end user from the Worklist or custom client application. This type of start requires that the end user know the specific business conditions that determine when to start the workflow. You

might create a manual start when you cannot create a trigger that captures all required conditions.

- Called — the workflow is started by a call from another workflow using the Start Workflow action. For details, see "Calling a Sub-Workflow" on page 6-36.
- Event — the workflow starts upon an external event trigger, such as the receipt of an XML document on a JMS queue, or a plug-in-defined event. For more information, see "Defining Event-Triggered Start Properties" on page 5-46.
- Timed — the workflow runs as a time-scheduled job, starting on the date and time you define. For more information, see "Defining a Timed Start Node" on page 5-36.

Note: When you create a template definition, the default Start node is set to a manual start.

You can also specify more than one Start node, for various purposes, for example:

- To specify separate, independent paths of work to start simultaneously. This can be accomplished by setting all the start nodes to a manual start, to the same start time, or to the same event trigger.
- To specify separate, independent paths of work to start at different times. This can be done by specifying different trigger types in each node.
- To specify different conditions, that is, different triggering *events*, to start the same path of work.

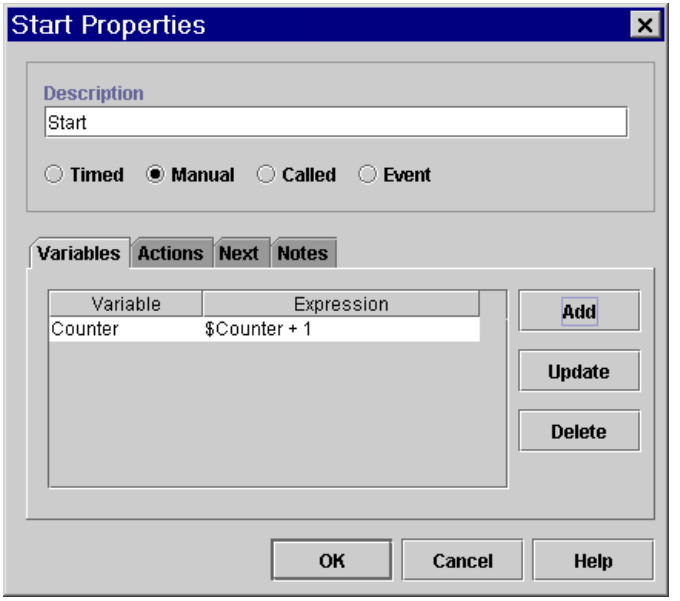
Note: To specify workflows that use sequential, rolling times, so that one flow expires and another one starts, you should define separate template definitions with the appropriate effective and expiry dates. For more information, see "Working with Template Definitions" on page 5-7.

Finally, you can use the Start node to initialize any variables created for the workflow. For example, you might have a counter in your workflow that you wish to take the value of 1 at the start of the workflow. You can assign this value to the counter variable upon activation of the start node. For more information, see "Initializing Variables from Event Data" on page 5-45.

To define a Start node:

1. Double-click the Start node or right-click it in the folder tree and choose Properties to display the Start Properties dialog box.

Figure 5-15 Start Properties Dialog Box

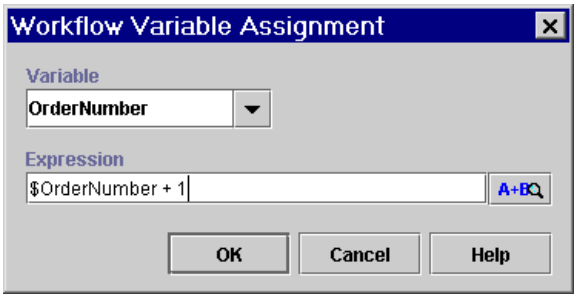


2. Optionally, in the Description field, modify the name of the Start node as needed to create a unique, identifiable name.

3. Select a workflow triggering method. If you select Timed, follow the procedure given in "Defining a Timed Start Node" on page 5-36 to specify additional options. If you select Event, follow the procedure given in "Defining Event-Triggered Start Properties" on page 5-46.

4. Optionally, to initialize variables when the workflow starts, select the Variables tab and click add to display the Workflow Variable Assignment dialog box.

Figure 5-16 Workflow Variable Assignment Dialog Box



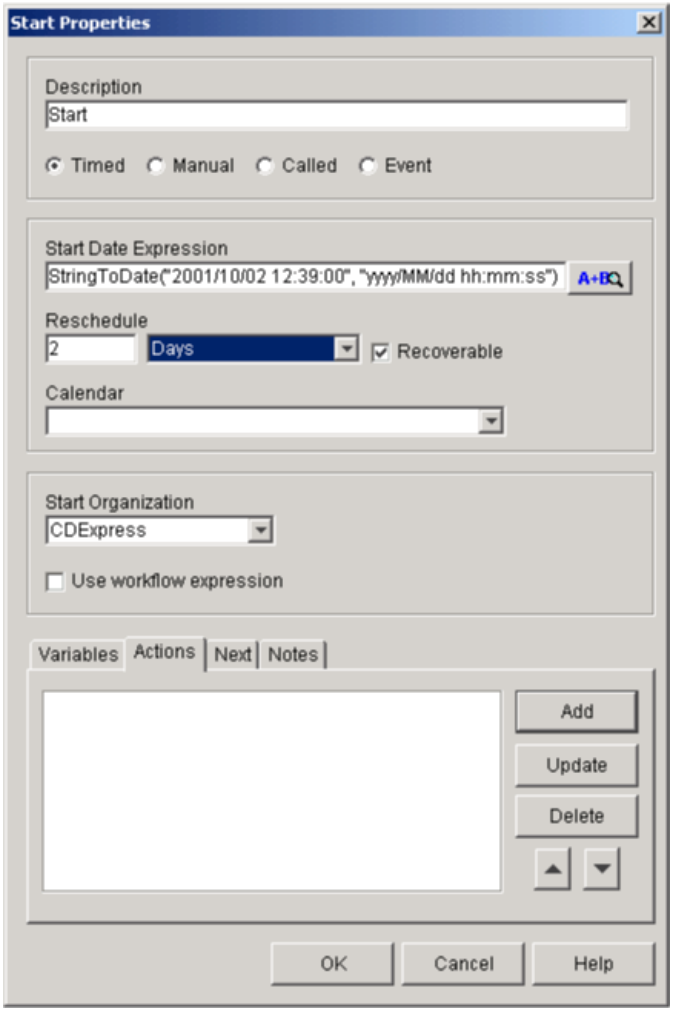
5. From the Variable drop-down list, select a variable to initialize.
6. In the Expression field, enter the expression that is evaluated at run time to produce the value for the variable. To specify a constant, use the syntax provided in "Using Literals" on page 8-2.
7. Optionally, add actions to be performed when the start node is initiated. For further details about actions, see Chapter 6, "Defining Actions."
8. Click OK to save your changes.

Defining a Timed Start Node

You can start a workflow at an exact time and date by specifying a start date expression. To start a workflow, you also need to specify the organization in which the workflow should be started.

You can also specify an interval at which the workflow will be restarted, such as every two days. In this case, an instance of the workflow will start every two days until the template expiry date. After the template expiry date, no additional workflows will be started.

Figure 5-17 Start Properties Dialog Box: Timed Option



To define a timed Start node:

1. In the Start Properties dialog box, select the Timed option.
2. In the Start Date Expression field, enter an expression that specifies the start date and time for the workflow, as an absolute or relative value. The expression must return a Date object, so you must use a date function, as follows:

- Use the `StringToDate()` function to specify an absolute date and time value. For details, see "StringToDate()" on page 8-19.
- Use `DateAdd()` to specify a value relative to a constant base date and time. For details, see "DateAdd()" on page 8-21.

3. Optionally, in the Reschedule field, specify an interval of time at which the workflow will be restarted by entering a value in the field, and selecting a unit of time from the drop-down list.

   Set the Recoverable checkbox to indicate whether or not you would like a timed workflow to be recovered (that is, deferred until the server restarts) or skipped, if the server is not running at its scheduled start time.

4. Optionally, select a business calendar that is used to evaluate the start date.
5. In the Start Organization field, select an organization in which to start the workflow, by doing one of the following:
   - Select an organization from the drop-down list.
   - Select the Use workflow expression check box, and in the Start Organization field, enter a string, surrounded by quotation marks, that specifies the organization, or an expression that will be evaluated at run time and become the name of an organization.
6. Click OK to save the Start node.

Defining Event And Event-Triggered Start Properties

A workflow can be started, or nodes within a workflow triggered, by an event. An event is an asynchronous notification from another workflow or from an external source, such as another application. Start nodes can be defined as event-triggered, and Event nodes can only be triggered by an external event.

An event notification most typically takes the form of an XML document contained in a Java Message Service (JMS) message and received on a JMS queue, although it may also be plug-in defined, which means that the event notification can be a custom trigger rather than an XML document. (For more information, see *Programming BPM Plug-Ins for WebLogic Integration*).

The JNDI name of the default internal JMS queue for WebLogic Integration, from which messages are consumed by workflows, is `com.bea.wlpiEventQueue`. However, you can also set up alternate message queues; for more, information see "Configuring a Custom Java Message Service Queue" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

In an XML event type, the actual trigger is either the document type declaration (DOCTYPE) specified in the prolog of the XML message, or it is the root element of the XML message. You specify the DOCTYPE or root element with which you want to trigger the event or start the workflow in a Start or Event node's properties dialog box. The event is not triggered unless the DOCTYPE or root element specified in the node's properties dialog box matches that in the incoming XML message.

In addition to using the DOCTYPE or root element, you can further qualify an event with an event *key* and an event *condition*. These are described below.

Understanding Event Keys

An event key allows you to specify the contents or JMS header or property values of incoming XML messages that will trigger a Start or Event node. That is, rather than allowing all incoming XML documents with a particular DOCTYPE or root element to trigger the node, you can filter the instances of incoming XML messages according to specific values contained in the XML body or JMS header fields, so that only a particular message, or messages, containing those values can trigger the node in the running workflow.

An event key consists of two parts:

- *Key value expression*

  You specify the key value expression in the Properties dialog box for Start or Event nodes. The key value expression is a workflow expression that is evaluated at run time to specify the exact data that the incoming message must contain for the node to be triggered. In a Start node, the expression typically contains a constant that refers to particular, recurring data contained in the incoming XML document or a JMS header. In an Event node, the expression typically contains variables or functions to obtain a unique value at run time. If you use event keys, each event node should specify a unique key.

  Examples of key value expressions are given in the following sections, and steps for defining key value expressions are given in "Defining Event-Triggered Start Properties" on page 5-46 and "Defining Event Properties" on page 5-49.

- *Event key expression*

  This is an expression that returns the key value from the header or body of the incoming message at run time and converts it to the data type required by the corresponding key value expression in a Start or Event node. You specify the event key expression in an event key expression dialog box that you access from the Configuration menu. The expression typically contains an XPath language expression to parse the XML document, or an EventAttribute() function expression to extract a value from a JMS message header. Once an event key expression is configured, it is available for all workflows in all organizations. Examples of event key expressions are given in the following sections, while procedures for configuring event key expressions are given in "Configuring Event Keys" on page 4-18.

Note: A detailed description of the workflow expression language is provided in Chapter 8, "Using Workflow Expressions." Specific information on XPath and `EventAttribute()` functions is provided in "Extracting Run-Time Event Data" on page 8-7.

Using XML Content as an Event Key

As a simple example, imagine that you have regularly incoming XML messages that, among other things, report customer account information from an accounts receivable application. These messages contain the following elements, among other data:

Listing 5-1 Example Incoming XML Document

```
<account>
        .
        .
        .
        <number>847365</number>
        <customer>John Doe</customer>
        <balance>
                <status>past due</status>
                <date_due>7-11-2001</date_due>
                <amount_due>5670.85</amount_due>
        </balance>
        <credit_limit>7500.00</credit_limit>
</account>
```

Let us say that you have a workflow that processes overdue accounts, so that only incoming documents with a balance status of "past due" (as opposed to "OK" or "pending", for example) should trigger the workflow. First, you specify that the root element of the incoming document must be `<account>` for the document to even be considered as a trigger for this workflow. Then, you create an event key to correspond to the value of `past due`. At run time, the event processor compares the value returned by the balance status element in the incoming XML document with the value specified in the Start node. If there is a match, the workflow is triggered.

Start nodes typically use a *constant* as an event key so that multiple instances of the workflow can be started by multiple instances of the incoming XML document. In contrast, an Event node inside of a workflow will typically need to be triggered only by a specific instance of an XML document that contains some specific data already captured elsewhere in the current workflow; for example, in a variable initialization in the Start node (see "Initializing Variables from Event Data" on page 5-45). Since this value can not be determined at design time, it must be expressed as a workflow *variable* or function that can return the desired value at run-time. For example, using the document in Listing 5-1, an event key could specify the value of the account number, for example, to ensure that the event instance is only triggered by the XML instance containing the correct account number, in this case, `847365`. At run time, the event processor compares the value returned by the account number in the incoming XML document with the value returned by the expression specified in the Event node. If there is a match, the event is triggered.

Let us now look at how we would construct our key value and event key expressions in light of our example. For our Start node, our key value expression would consist of a constant (surrounded by quotation marks, as required by workflow expression syntax) as follows:

```
"past due"
```

The event key expression required to return this value from the XML document, that you specify in the event key configuration, is:

```
ToString(XPath("/account/balance/status/text()"))
```

For our Event node, our key value expression would consist of a variable (expressed by the dollar sign in workflow expression syntax) that we have created in the workflow, and whose value has presumably been set earlier by the running workflow instance:

```
$AccountNumber
```

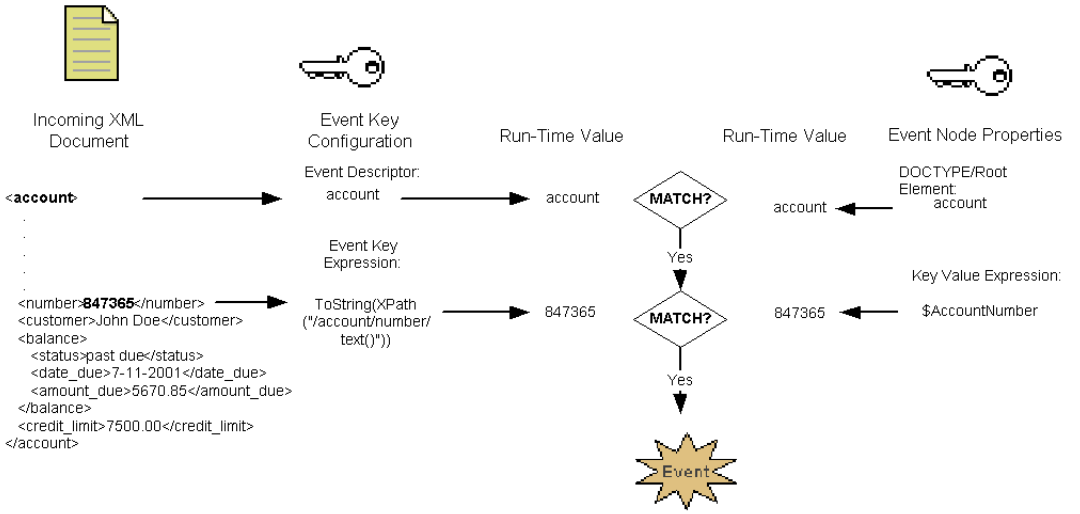
Assuming that the account number is stored as a string in the workflow variable, the expression required in the event key configuration to return this value from the XML document is:

```
ToString(XPath("/account/number/text()"))
```

Note: An event key expression must evaluate to the same data type as that used by the key value expression in a Start or Event node. Since XPath expressions return a *node list* type, you will usually need to use a typecasting function provided in the workflow expression language to return the correct data type. For more information on these functions, see "Converting Data Types" on page 8-17. If the returned data type is a string, you can also use the XML dot notation. See "XML Element Dot Notation" on page 8-12 for details.

The following figure summarizes the event key mechanism for XML content.

Figure 5-18 Event Key Mechanism



Using JMS Header or Property Data as an Event Key

You can also use the `EventAttribute()` function to retrieve specific values from JMS headers or properties. The mechanism functions exactly the same way as for an XML document, but rather than parsing the XML document for the target value, the value can be extracted from a JMS property.

For example, let us say the sending application uses a property field to indicate the country from which a message is originating, that we will call `Country`, and that you have different workflows that should be started according to the country of origin. Your event key expression would look like the following:

```
ToString(EventAttribute("Country"))
```

Note: An Event key configuration expression must evaluate to the same data type as that by the key value expression in a Start or Event node. Since `EventAttribute()` expressions return an *object* type, you will usually need to use a typecasting function provided in the workflow expression language to return the correct data type. For more information on these functions, see "Converting Data Types" on page 8-17.

For a workflow that should only process information pertaining to Canada, the key value expression in your start node could be:

```
"Canada"
```

Now let us say that also contained in this message is a property called `Province`. Within your start node, you extract this information and store it in a variable for containing province names. Once the workflow is instantiated, you want to ensure that only messages coming in for that particular province are used to trigger another event instance in the workflow, say for example, to call a business operation that will calculate a sales tax value. In this case, you could create an event key for the province. The event key expression would be:

```
ToString(EventAttribute("Province"))
```

The key value expression would consist of the variable name:

```
$ProvinceName
```

Understanding Event Conditions

To even further qualify the trigger of an Event or Start node, you can specify a condition that must be evaluated. In this way, even if the event processor has identified an event key match, the event will still not be triggered unless the condition is met.

Note: Although you can use an event condition without an event key, this is not recommended. The event key mechanism provides significantly better performance than a simple event condition, as it stores the target value in memory and involves much less DOM parsing on the incoming XML

document. You should only use an event condition as an *additional* filter, along with an event key, when you wish to further restrict the XML message instance that should trigger the event.

Again using the example of the XML document listed in Listing 5-1, imagine there is an event within a workflow that issues a credit freeze on accounts that are past due and have balances over a certain amount, say 75 percent of the credit limit on the account. The workflow has previously extracted the values from the `<amount_due>` and `<credit_limit>` elements and stored them in two variables, Amount_Due and Credit_Limit, respectively. You could use the following expression as a condition to ensure that the event is only triggered (and the operations to perform the credit freeze) if the balance exceeds 75 percent of the credit limit:

```
$Amount_Due > .75 * $Credit_Limit
```

In the case of the example document instance in Listing 5-1, the condition would evaluate to true, and the event would be triggered.

You can also use `XPath()` and `EventAttribute()` functions in event conditions to directly extract content from XML or JMS header or property data and compare it with other data, including constants, variables or even other functions. To continue the country example, you could have an event that should only be fired if the country of origin is one that you specify. Imagining that the country information were embedded in an XML element, such as `<country>`, your condition would look like this:

```
ToString(XPath("/root_element/child_element/country/text()")) =
"Canada"
```

If the country value were embedded in a JMS property called `Country`, your condition would look like this:

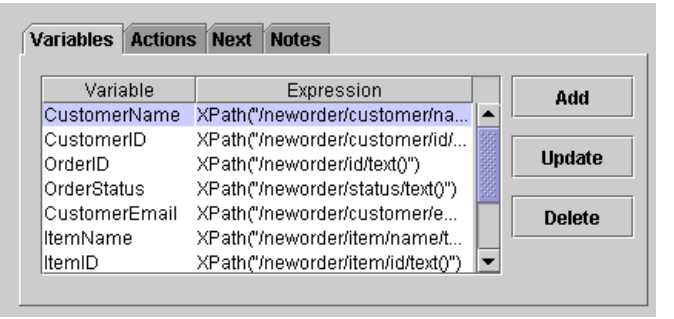
```
ToString(EventAttribute("Country")) = "Canada"
```

Note: If you use conditions with functions such as `XPath()` or `EventAttribute()`, keep in mind that the expressions on both sides of the equation must evaluate to the same data type, or the server will not be able to process the condition. For more information on Studio typecasting functions, see "Converting Data Types" on page 8-17. You can also use XML dot notation for string values. See "XML Element Dot Notation" on page 8-12 for details.

Initializing Variables from Event Data

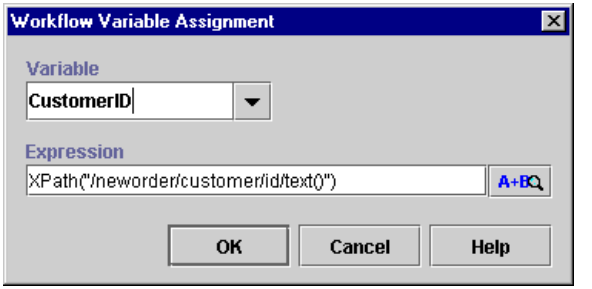
The properties dialog boxes of Start and Event nodes contain a Variables tab that you can use to add, update, or delete variables whose values you want to set when a workflow is started or an event is triggered. (For information on variables, see "Working with Variables" on page 5-28.)

Figure 5-19 Variables Tab of Start and Event Node Properties Dialog Boxes



Clicking Add displays the Workflow Variable Assignment dialog box, which you can use to assign values to any variables already defined for the workflow. These are listed in the Variable drop-down list, from which you select the variable you want to initialize.

Figure 5-20 Workflow Variable Assignment Dialog Box



Note: You can also use the Set Workflow Variable action on the Actions tab to accomplish the same thing. In fact, for all nodes except Starts or Events, or if you want to assign an XML document to an XML variable, you *must* use this action. (For more information, see "Setting a Variable Value" on page 6-21.)

Note, however, that in Start or Event nodes, when the workflow is executed, variables specified on the Variables tab are initialized *before* any actions are executed, in the sequence in which they are listed on the Actions tab.

Although you can use this feature in a Start node to initialize variables to constant values when the workflow starts, it is probably most useful for capturing incoming event data that is to be used to set multiple variable values. For example, if the node is to be triggered by an incoming XML document in a JMS message, you can use multiple expressions to initialize variables with values contained in the document, or specified by JMS header or property fields.

You can also use track attributes of other workflows such as instance IDs or template names that may be passed via XML documents or JMS properties to the current workflow. For example, you will need to extract these values and store them in variables if you want to use them later in addressed message replies to the workflows that initiated the conversation. You can also use the mechanism to gather together multiple workflow attributes that you can forward in a single JMS property header in a message delivered to an external application via a JMS topic or queue. (For more information about addressed messaging and inserting workflow attributes as JMS properties, see "Posting an XML Message to a JMS Topic or Queue" on page 6-82.)

To update a variable value, you highlight the variable name in the list, and click Update to invoke the Workflow Variable Assignment dialog box.

To delete a variable assignment, you select the variable name in the list, and click Delete.

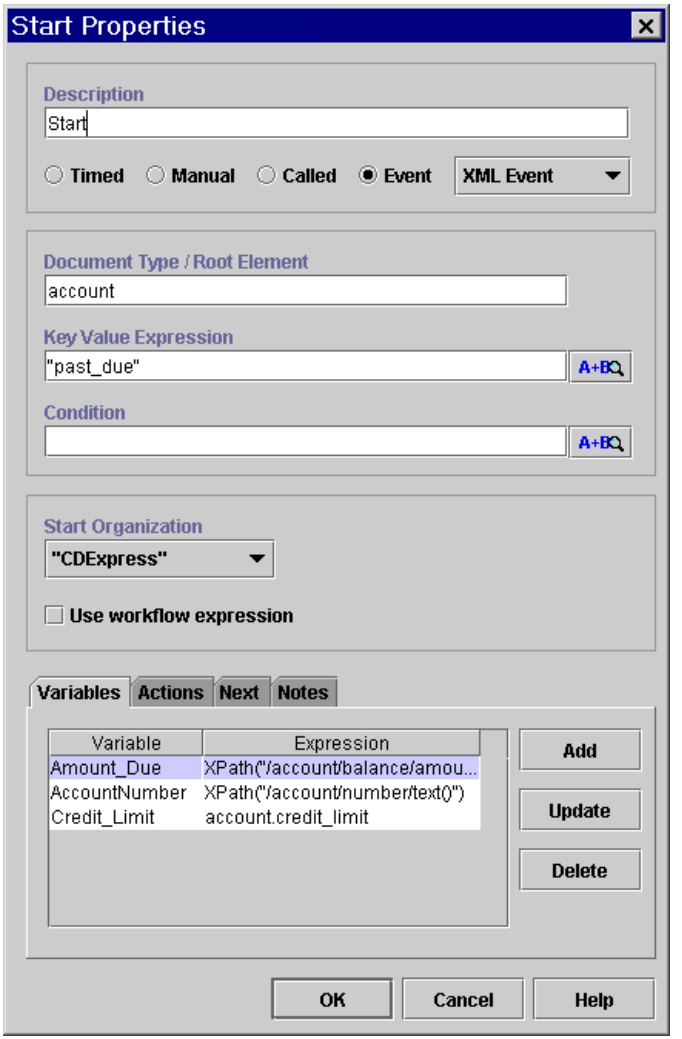
Defining Event-Triggered Start Properties

When you define an event-triggered start, you need to specify the organization in which the workflow should be started. You can specify the organization at design time or use an expression that determines the organization at run time, for example, by extracting the data specified in the incoming event message.

To define an event-triggered Start node:

1. In the Start Properties dialog box, select the Event option.

Figure 5-21 Start Properties Dialog Box: Event Option



2. In the Document Type/Root Element field, enter the DOCTYPE or root element in the XML message that should trigger the start of the workflow.

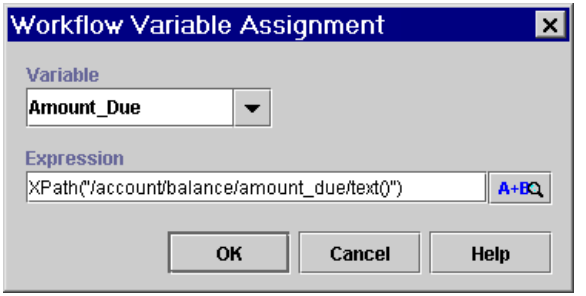
3. In the Key Value Expression field, optionally define a key value for the XML message by entering a workflow expression that will evaluate to the exact XML content or JMS header or property field value at run time that should trigger the

event. The expression will typically consist of a constant literal. For more information on key value expressions, see "Understanding Event Keys" on page 5-39. For more information on constructing expressions, see Chapter 8, "Using Workflow Expressions."

Note: You also need to define an event key configuration that locates the key value in the incoming XML message, so the process engine can compare it against the key value you specify in this field. For details, see "Configuring Event Keys" on page 4-18.

4. In the Condition field, optionally define a condition that needs to be evaluated before the workflow can start. For more information on event conditions, see "Understanding Event Conditions" on page 5-43.
5. In the Start Organization field, select an organization in which to start the workflow, by doing one of the following:
   - Select an organization from the drop-down list.
   - Select the Use workflow expression check box, and in the Start Organization field, enter a string, surrounded by quotation marks, that specifies the organization, or an expression that will be evaluated at run time and become the name of an organization. This could be an expression that extracts organization information from the incoming XML message, with the `XPath()` or `EventAttribute()` functions.
6. Select the Variables tab to initialize variables from the incoming event data or otherwise, and click Add to display the Workflow Variable Assignment dialog box.

Figure 5-22 Workflow Variable Assignment Dialog Box



7. From the Variable drop-down list, select a variable to store incoming data.

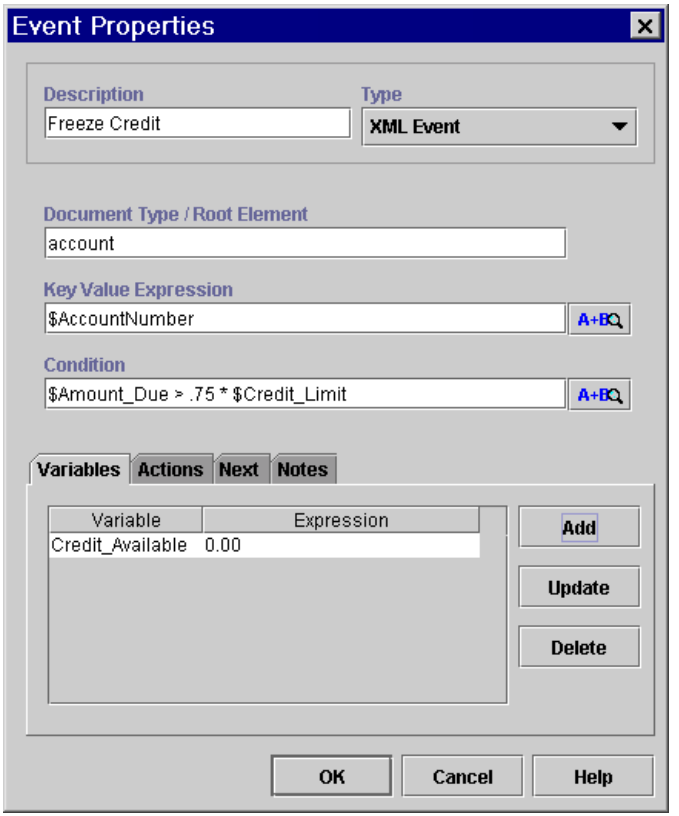
8. In the Expression field, enter the expression that is evaluated at run time to produce the value for the variable, by doing one of the following:
   - To specify a constant, use the syntax provided in "Using Literals" on page 8-2.
   - To capture incoming JMS header data, use an `EventAttribute()` function. (For information, see "EventAttribute()" on page 8-8.
   - To capture incoming XML content, use an `XPath()` function (for information, see "XPath()" on page 8-10), or the dot notation for XML elements to be returned as strings (for information, see "XML Element Dot Notation" on page 8-12). You can also use the Expression button  to invoke the XPath Wizard, from which you can generate XPath expressions automatically from a sample incoming document. For information, see "Creating XPath Expressions Using the XPath Wizard" on page 8-31.
9. Click OK. The variable initialization appears in the list on the Variables tab of the Start Properties dialog box.
10. Repeat steps 6 to 9 for all variables you want to initialize.
11. Click OK to save your changes.

Defining Event Properties

To define an Event node:

1. Double-click the event node or right-click it in the folder tree and choose Properties to display the Event Properties dialog box.

Figure 5-23 Event Properties Dialog Box



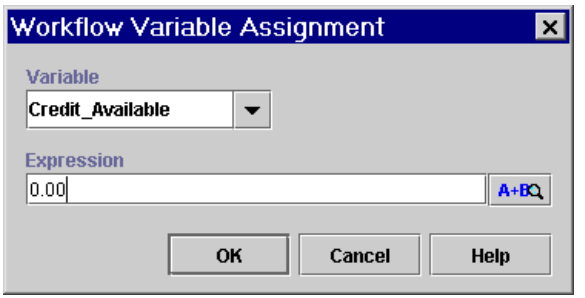
2. In the Description field, modify the default name to create a unique, identifiable name for the event, such as Wait for New Inventory.
3. In the Document Type/Root Element field, enter the DOCTYPE or root element in the XML message that triggers the event.
4. In the Key Value Expression field, optionally define a key value for the XML message by entering a workflow expression that will evaluate to the exact XML content or JMS header or property field value at run time that should trigger the event. The expression will typically consist of a variable or workflow function. For more information on key value expressions, see “Understanding Event Keys” on page 5-39. For more information on constructing expressions, see Chapter 8, “Using Workflow Expressions.”

Note: You also need to define the expression that locates the key value in the incoming XML message, so the process engine can compare it against the key value you specify in this field. For details, see "Configuring Event Keys" on page 4-18.

5. In the Condition field, optionally define a condition that needs to be evaluated before the event can be triggered. For more information on event conditions, see "Understanding Event Conditions" on page 5-43.

6. Select the Variables tab to initialize variables from the incoming event data or otherwise, and click Add to display the Workflow Variable Assignment dialog box.

Figure 5-24 Workflow Variable Assignment Dialog Box



7. From the Variable drop-down list, select a variable to store incoming data.

8. In the Expression field, enter the expression that is evaluated at run time to produce the value for the variable, by doing one of the following:

   - To specify a constant, use the syntax provided in "Using Literals" on page 8-2.
   - To capture incoming JMS header data, use an `EventAttribute()` function. (For information, see "EventAttribute()" on page 8-8.
   - To capture incoming XML content, use an `XPath()` function (for information, see "XPath()" on page 8-10), or the dot notation for XML elements to be returned as strings (for information, see "XML Element Dot Notation" on page 8-12). You can also use the Expression button  to invoke the XPath Wizard, from which you can generate XPath expressions automatically from a sample incoming document. For information, see "Creating XPath Expressions Using the XPath Wizard" on page 8-31.

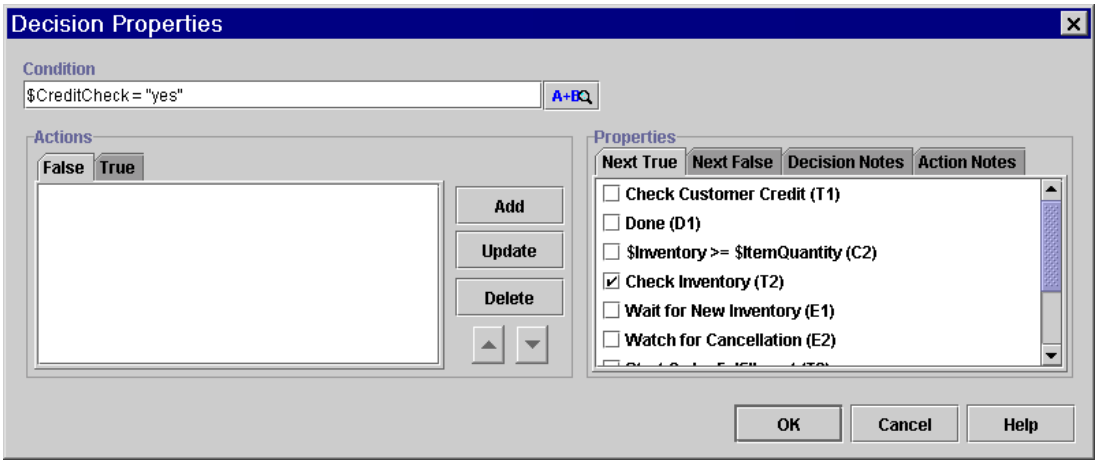
9. Click OK. The variable initialization appears in the list on the Variables tab of the Event Properties dialog box.
10. Repeat steps 6 to 9 for all variables you want to initialize.
11. Optionally, add actions to be performed when the event is triggered. For further details about actions, see Chapter 6, “Defining Actions.”
12. Click OK to save your changes.

Defining Decision Properties

A workflow can use any number of decision. Each decision node contains a condition that is evaluated when a transition to that decision node occurs. The result is either True or False, with subsequent flow of control passed to different paths, according to the result.

Additionally, it is possible to specify actions to be executed on both a True and a False evaluation of the condition. Actions defined on the True and/or False tab are performed before nodes specified as targets of a true or false branch. They are also listed under True and False folders under Decision folders in the folder tree.

Figure 5-25 Decision Properties Dialog Box



To define a Decision node:

1. Double-click the Decision node or right-click it in the folder tree and choose Properties to display the Decision Properties dialog box.
2. In the Condition field, specify the conditional expression that will be evaluated at run time. The condition can contain constants, variables, or functions. For more information on constructing expressions, see Chapter 8, "Using Workflow Expressions."
3. Optionally, add actions to the False and/or True tabs specify actions to be performed depending on the result (true or false) of the condition at run time. These actions will be performed before actions specified in the successor node.

   For further details about actions, see Chapter 6, "Defining Actions."
4. Click OK to save your changes.

Defining Task Properties

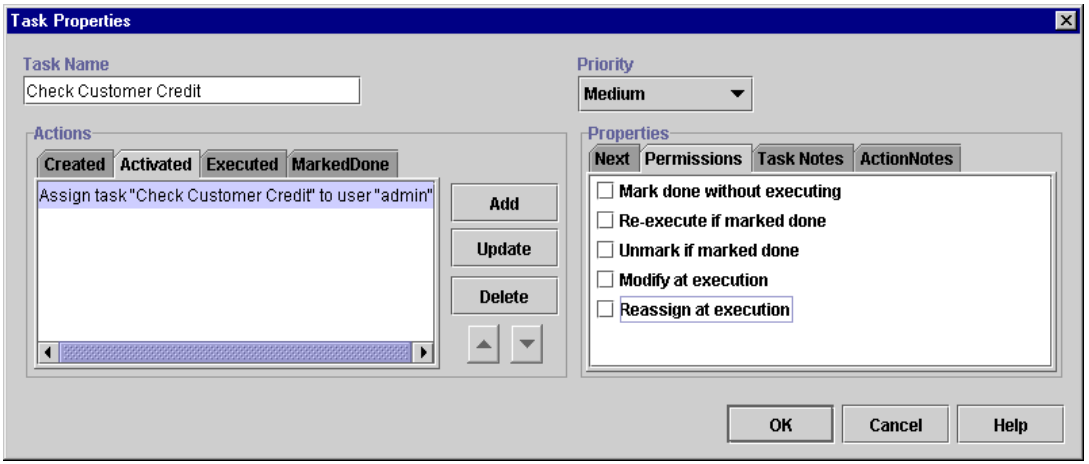
A task node represents the following:

- A unit of work, containing actions that are performed manually or automatically by software components
- A task that is assigned to a Worklist or custom client application user for manual execution

Task nodes are the building blocks of a workflow and should ideally represent a single operation within the flow, although this may need to be implemented by multiple workflow actions.

In the workflow transaction model, workflow nodes enter states of activation and execution that determine how a transaction is processed, as described in "Understanding the BPM Transaction Model" in *Programming BPM Client Applications*. For all nodes other than tasks, these states are not explicitly represented in the Studio. For Task nodes, however, you must specify actions according to one of four task states, and it is important that you understand their meaning to be able to effectively manage control of your flow.

Figure 5-26 Task Properties Dialog Box



Understanding Task States

Each task node progresses through four different states: Created, Activated, Executed, and Marked Done. For each state, you can specify a list of actions that are performed when the task reaches that state. The following table shows each task state, and when the actions specified in each state are executed:

Table 5-3 Task States

Task states	Actions specified in this state are executed . . .	Actions typically specified here are...
Created	When a new workflow instance is created. The task remains in the Created state until the the Task node is reached in the workflow.	You are unlikely to need to specify any actions in this state, since you can initialize variables in a Start node.
Activated	When the previous node has completed its processing. The task remains in Activated state until it is executed by: ■ a Worklist or custom client user ■ the Execute task action, ■ programmatically by an API call.	■ Actions that precede or do not include any user-assigned task actions. ■ User-assigned task actions.

Table 5-3 Task States

Task states	Actions specified in this state are executed . . .	Actions typically specified here are...
Executed	■ When a user has manually executed the task using the Worklist or custom client application ■ When an Execute Task action has been specified ■ Programmatically, using the WebLogic Integration APIs The task remains in the Executed state, and the flow will not proceed, until the task is marked done.	Actions that follow user-assigned task actions.
Marked Done	■ When a user manually marks the task done in the Worklist or Studio applications ■ When the Mark Task as Done action has been specified ■ Programmatically using the WebLogic Integration APIs	You are unlikely to need to specify any actions in this state, since you can specify actions in the following node.

All task nodes, unlike other nodes, must be marked done to signal the completion of the node, or the workflow will not proceed. For manually assigned tasks, you can set a permission (see below) that allows a user to mark the task done at run time. More typically, however, you explicitly mark a task done at design time by adding a Mark Task as Done action.

The tab on which the Mark Task as Done action should be placed depends on whether the task can be executed by any of the means listed in Table 5-3. If it is executed, you specify the Mark Task as Done action on the Executed tab; if it is not, you specify the Mark Task as Done action on the Activated tab. For more information, see "Marking Tasks Done" on page 6-11.

Actions defined in the Task properties dialog box are also displayed in the folder tree under Created, Activated, Executed, and Marked Done folders according to their placement.

About Task Permissions

You can assign permissions to a task, to control the types of operations that can be performed for a task at run time by a Worklist (or custom client) user or by an administrator monitoring instances in the Studio. For more information on performing operations on tasks at run time in the Studio, see “Changing Task Permissions and Priority” on page 10-14 and “Changing Task Status and Assignment” on page 10-16. For more information on performing operations on tasks at run time in the Worklist, see *Using the WebLogic Integration Worklist*.

When you create a Task node, by default, no permissions are initially assigned, but you can enable any of the available task permissions as listed in the following table.

Table 5-4 Task Permissions

Permission	Explanation
Mark done without executing	Allows a Worklist user or Studio administrator to mark the task done without it having been executed, and manually set the task's completed date.
Re-execute if marked done	Allows a Worklist user to re-execute a task even if it has already been completed.
Unmark if marked done	Allows a Worklist user or Studio administrator to change the status of the task back to not done when the task is marked as done. (If a user marks a completed task as not done, the task status is set to Active; it does not, however, reverse the effects of any actions that have been executed.)
Modify at execution	Allows a Worklist user or Studio administrator to change the permissions for a task before it has been executed.
Reassign at execution	Allows a Worklist user to take or reassign the task, or a Studio administrator to reassign the task to another user or role before it has been executed.

About Task Priority

For manually assigned tasks, you can assign a priority level. The priority has no effect on how the node or its actions are executed at run time, but is simply displayed to Worklist users, who can execute and sort their tasks accordingly.

Priority options are low, medium, and high. The default value is medium.

Defining the Task Node

To define a Task node:

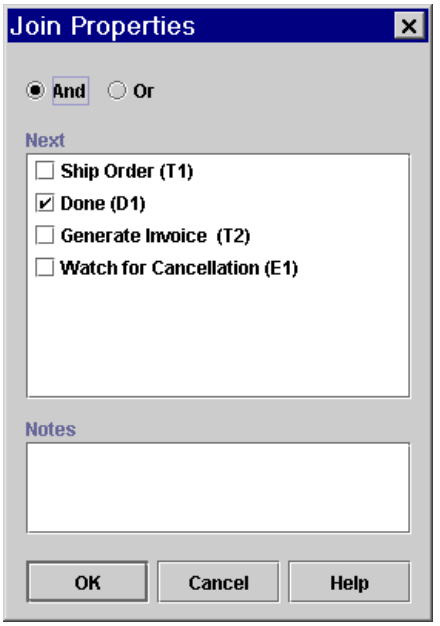
1. Double-click the task node or right-click the task in the folder tree and choose Properties to display the Task Properties dialog box.
2. In the Task Name field, modify the default name to create a unique, identifiable name, such as `Confirm Order`. This value is used primarily in the Worklist application, but it is visible from the various Studio monitoring reports.
3. Add actions to be performed for each task state under the appropriate tabs. For more information about these states, refer to Table 5-3. For details about specifying actions, see Chapter 6, "Defining Actions."
4. Optionally, for manually assigned tasks, assign permissions to the task by selecting check boxes on the Permissions tab, as described in Table 5-4.
5. Optionally, for manually assigned tasks, assign a priority to the task by selecting Low, Medium, or High.
6. For non-manually assigned tasks, or manually assigned tasks that do not enable Mark Done Without Executing permission, mark the task done. For more information, see "Marking Tasks Done" on page 6-11.
7. Click OK to save your changes.

Defining Join Properties

Join nodes are used to merge multiple paths of Task, Event, and Decision nodes into a single path. An AND join causes the workflow to wait until all preceding paths have finished executing before proceeding to the next node. An OR join causes the workflow to proceed to the next node after a single preceding path has finished executing, blocking any preceding unexecuted nodes from executing.

Once you have created a Join, you can change it from AND to OR and vice versa. The shape for the Join is automatically updated in the design area.

Figure 5-27 Join Properties Dialog Box



To define a Join node:

1. Double-click the Join node or right-click it in the folder tree and choose Properties to display the Join Properties dialog box.
2. Enable one of the following options:
   - And—changes the node to an AND join.
   - Or—changes the node to an OR join.
3. Click OK to save your changes. The shape is updated in the design area.

Defining Done Properties

A Done node represents the point within the workflow where the process completes. Once a done node is reached in the workflow, the running instance of the workflow is marked done, regardless of whether all the done nodes have been reached or not.

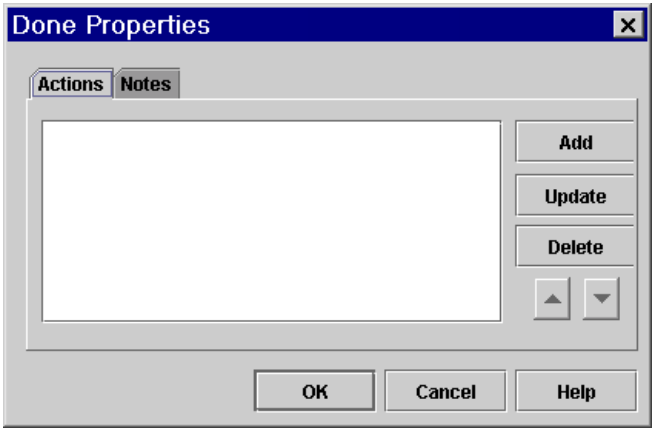
You can specify any number of Done nodes in a workflow. If a workflow can exit from various points, you have the option of placing separate done nodes wherever needed.

Although you can add actions to a Done node, this is not recommended.

To define a Done node:

1. Double-click the Done Node or right-click it in the folder tree and choose Properties to display the Done Properties dialog box.

Figure 5-28 Done Properties Dialog Box



2. Optionally, add actions to be performed when the Done node is reached.
3. Click OK to save your changes.

Working with Exception Handlers

Exception handlers function like sub-flows of actions within a main workflow, which are triggered by the occurrence of a server exception, or can be invoked with the Invoke Exception Handler action. Complete information on defining and invoking exception handlers is provided in Chapter 9, “Handling Workflow Exceptions.”

CHAPTER

6 Defining Actions

The following sections introduce and explain how to use WebLogic Integration actions to accomplish various workflow activities:

- Understanding Actions
- Overview of Action Definition Tasks
- Working with Actions
- Setting a Variable Value
- Controlling Program Flow
- Using Timed Operations
- Using Sub-Workflows
- Monitoring Run-Time Status
- Setting Up Manual Tasks
- Sending E-Mail Messages
- Invoking Components
- Posting an XML Message to a JMS Topic or Queue
- Transforming XML Documents
- Handling Exceptions

Understanding Actions

While nodes and connections serve to specify the logical sequence and control of business processes, it is actions that perform the real work of a workflow definition. An action is the most basic unit of a workflow that can perform some activity. This activity may be as simple as initializing a variable, or as complex as calling a custom application on a client system. Actions can be added to all types of nodes (except Joins), to exception handlers, and even to other actions.

The following sections provide important background information for using actions to perform activities in workflows. It includes:

- Action Categories
- Understanding Action Types and Placement
- Placing Actions in Task Nodes

Action Categories

Actions are organized into categories in the Studio. In this guide, however, we categorize actions into the major activities that you can perform. The following table lists the categories, the actions contained within them, the activities they accomplish, and the sections of the guide where each one is described in more detail.

Category	Actions	Use to . . .
Task	■ Mark Task as Done ■ Unmark Task Done ■ Execute Task	Control main program flow For information, see "Controlling Program Flow" on page 6-23
	■ Assign Task to User ■ Assign Task to Role ■ Assign Task Using Routing Table ■ Unassign Task ■ Set Task Comment ■ Set Task Priority	Assign and set properties of manual tasks For more information, see "Setting Up Manual Tasks" on page 6-44.
	■ Set Task Due Date	Set a due date for manual and non-manual tasks or introduce a time delay For more information, see "Setting Up Manual Tasks" on page 6-44.
Workflow	■ Mark Workflow as Done ■ Abort Workflow	Control main program flow For information, see "Controlling Program Flow" on page 6-23
	■ Start Workflow	Call a sub-workflow For more information, see "Using Sub-Workflows" on page 6-35.
	■ Set Workflow Variable	Initialize variables For more information, see "Setting a Variable Value" on page 6-21.
	■ Set Workflow Comment	Monitor run-time status For more information, see "Monitoring Run-Time Status" on page 6-42.

Category	Actions	Use to . . .
Integration	■ Perform Business Operation ■ Call Program	Integrate the workflow with external software components, such as EJBs and Java classes, or executable programs. For more information, see "Invoking Components" on page 6-76.
	■ Send XML to Client	Perform different operations on a Worklist or custom client system. For more information, see "Setting Up Manual Tasks" on page 6-44.
	■ Post XML Event	Exchange XML messages between workflows, components, and applications. For more information, see "Posting an XML Message to a JMS Topic or Queue" on page 6-82.
	■ XSL Transform	Transform an XML document from one format to another according to a style sheet. For more information, see "Transforming XML Documents" on page 6-96.
Exception Handling	■ Set Workflow Exception Handler ■ Exit Exception Handler	Manage the exception handler(s) to be used for a workflow For more information, see Chapter 9, "Handling Workflow Exceptions."
	■ Invoke Exception Handler	Call a sub-workflow to handle exceptions For more information, see Chapter 9, "Handling Workflow Exceptions."

Category	Actions	Use to . . .
Miscellaneous	■ No Operation ■ Cancel Workflow Event	Control main program flow For information, see "Controlling Program Flow" on page 6-23.
	■ Evaluate Condition	Embed a conditional sub-workflow For more information, see "Using Sub-Workflows" on page 6-35.
	■ Timed Event	Embed a timed sub-workflow and introduce a time delay For more information, see "Using Timed Operations" on page 6-31.
	■ Send E-Mail Message	Send e-mail to a user, role or external client For more information, see "Sending E-Mail Messages" on page 6-72.
	■ Make Audit Entry	Monitor run-time status and debug workflow design For more information, see "Monitoring Run-Time Status" on page 6-42.

Understanding Action Types and Placement

In addition to the action categories presented by the Studio, such as Task, Workflow, Integration, Exception Handling, and Miscellaneous categories, you should be aware of other distinctions between actions and their behavior:

- Terminal actions versus non-terminal actions, which can contain other actions
- Synchronous versus asynchronous execution of actions or sub-actions, which allows parallel processing

These distinctions are significant, as they can affect the order in which workflow operations are executed. Each of these distinctions is described in more detail in the following sections.

Terminal Actions and Non-Terminal Actions

Most actions are terminal, meaning that they cannot contain any other actions. However, some actions allow you to define sub-actions inside of them that are performed according to specific conditions, depending on the nature of the parent action. The following actions can have sub-actions defined within them and are, therefore, non-terminal:

- Evaluate Condition—sub-actions may be performed according to the true or false result of a condition that you define.
- Timed Event—sub-actions are executed according to a date and time, and may be re-executed according to a schedule.
- Send XML to Client—sub-actions may be performed when the workflow receives a reply from the client application.
- Start Workflow—sub-actions may be performed when the called workflow is completed.
- Set Task Due Date — sub-actions may be performed after a due time on a task has passed.

Note that sub-actions are only visible in the Properties dialog box for these non-terminal actions, but do not appear in the folder tree.

The special relevance of non-terminal actions is that, while most terminal actions are always executed in a *synchronous* fashion, which means that the workflow waits until they have finished executing, non-terminal actions can perform operations that are executed in both a synchronous or *asynchronous* fashion. Asynchronous execution of non-terminal actions means that the workflow does not wait for the action's operations, or its sub-actions, to complete before proceeding, but continues to be processed in parallel. This distinction is explained in more detail in the next section.

Synchronous and Asynchronous Execution of Actions

All actions are synchronous, which means that the workflow does not proceed to the next action until the operations performed by the current action have completed, with the exception of the following:

- Call Program—This action invoke an executable which always runs in parallel with the workflow.

- Post XML Event—This action sends off an XML message in a "fire and forget" model, so that any workflows or applications that are triggered by the receipt of the message are executed in parallel. (However, for information on forcing a synchronous functioning of this action, see "Posting an XML Message to a JMS Topic or Queue" on page 6-82.)

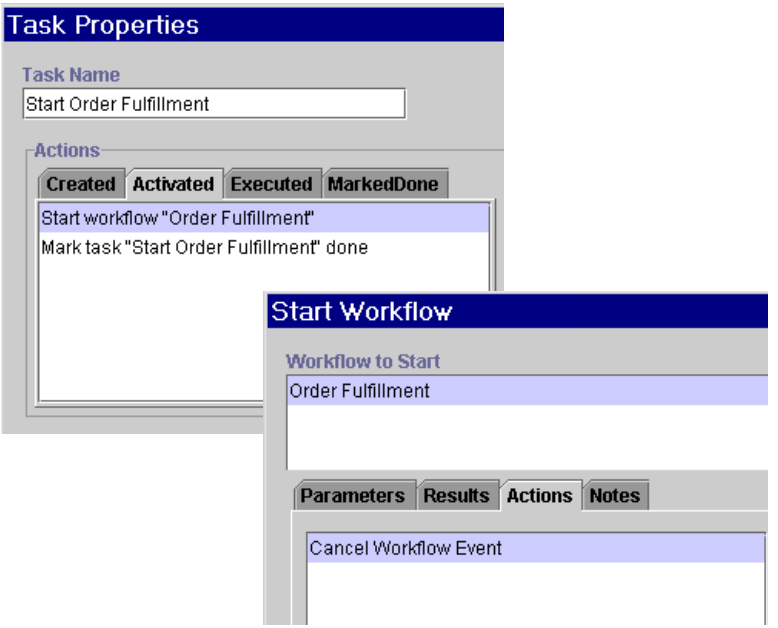
However, the following actions, and any sub-actions they may contain, can be executed in a synchronous *or* asynchronous fashion, depending on how you place the action:

- Timed Event—sub-actions may be performed asynchronously or synchronously
- Send XML to Client—XML message transmission, receipt of reply, and optional callback sub-actions may be performed asynchronously or synchronously
- Start Workflow—the sub-workflow and optional sub-actions may be executed asynchronously or synchronously
- Set Task Due Date —optional overdue sub-actions may be performed asynchronously or synchronously

These non-terminal actions and their sub-actions are performed asynchronously in the following conditions:

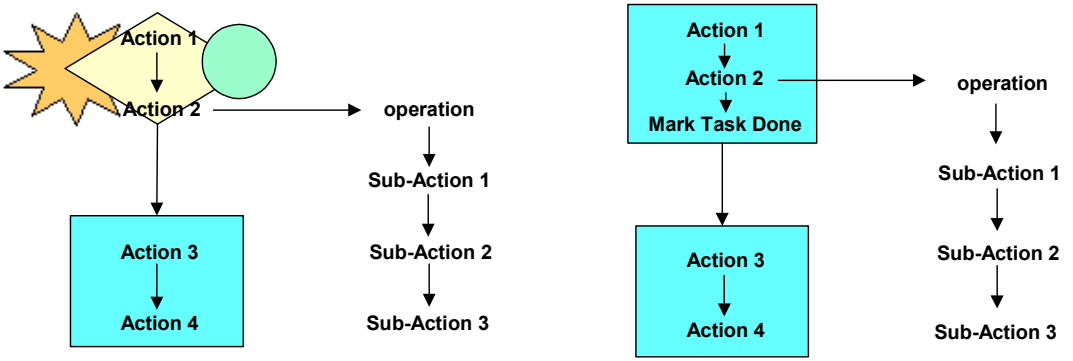
- If the action is placed in a Start, Event, or Decision node.
- If the action is placed in a Task node, and the Mark Task as Done action is placed in the Task Properties, as in the following figure.

Figure 6-1 Non-Terminal Action in Task Node with Task Marked Done from Task Properties



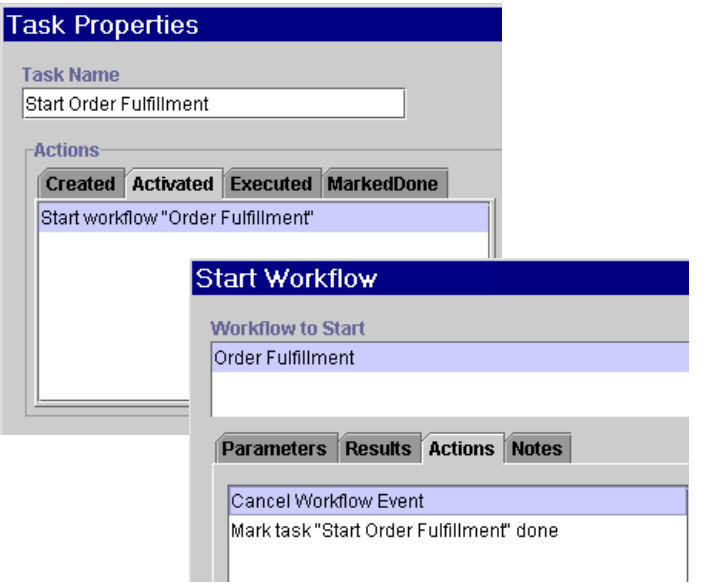
In these cases, the action's operations, and any specified sub-actions, are simply performed in parallel with the subsequent nodes in the workflow, as illustrated in the following figure.

Figure 6-2 Asynchronous Execution of Workflow Actions



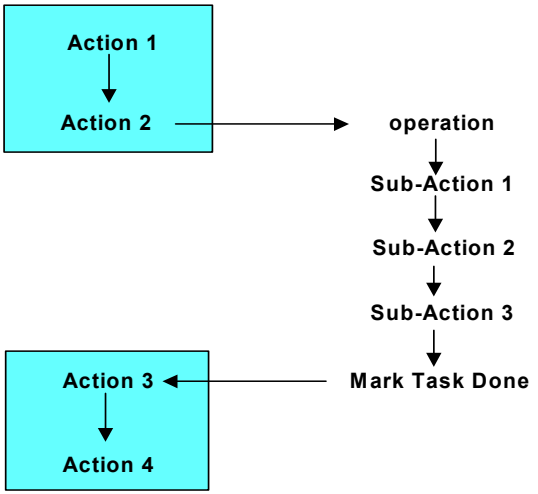
By contrast, non-terminal actions and their sub-actions are performed synchronously if the Task node in which they are placed is marked done in the action's properties, as illustrated in the following figure.

Figure 6-3 Non-Terminal Action with Task Marked Done in Action Properties



That is, the workflow waits until all operations and sub-actions are completed before proceeding to the next node, as illustrated in the following figure.

Figure 6-4 Synchronous Execution of Workflow Actions



More information on action design issues is provided in the following topics:

- "Calling a Sub-Workflow" on page 6-36
- "Embedding a Timed Sequence" on page 6-32
- "Setting a Task Due Date" on page 6-51
- "Sending an XML Message to a Client Application" on page 6-58

More information on marking tasks done is provided in the following sections.

Placing Actions in Task Nodes

When you use Task nodes, you need to know how to place actions in order to perform a particular activity. This requires that you do the following: place the action on the correct tab of the Task Properties dialog box (Activated or Executed), and mark the task done at the appropriate point. For more information on Task Properties dialog box tabs, see "Understanding Task States" on page 5-54.

Using the Activated and Executed Tabs

The only ways a task can be executed are:

- By using the Assign Task to User action to assign the task to a Worklist or custom client user, who executes it manually
- By using the Execute Task action
- Programmatically, through an API call.

Thus, in general, you can follow these guidelines when considering action placement:

- Any actions that should be performed before a user manually executes a task (or it is executed by the other means listed above) should be placed on the Activated tab.
- The Assign Task to User action is always placed at the end of the list on the Activated tab. Any actions listed after this action on the Activated tab are not performed.
- Place any actions to follow an Assign Task to User action on the Executed tab.

A detailed table with guidelines for using Task Properties dialog box tabs to perform each of the major activities identified in this guide is provided in "Guidelines for Action Placement in Task Nodes" on page 6-13.

Marking Tasks Done

All tasks must be marked done, either manually by a Worklist or custom client user, if the Mark Done Without Executing permission is assigned to the task (for more information, see "About Task Permissions" on page 5-56), or explicitly at design time by using the Mark Task as Done action.

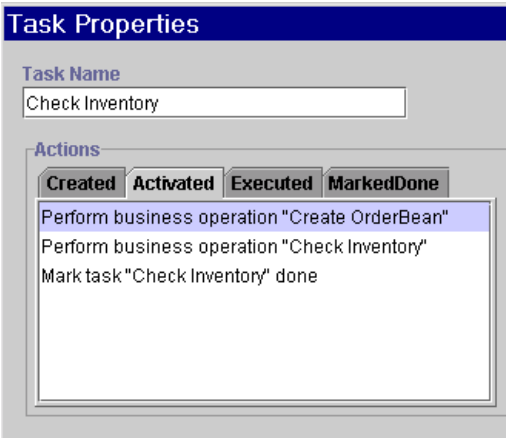
If a task is not marked done, the workflow will not proceed from the Task node. Conversely, any actions that are listed after a Mark Task as Done action in the Task Properties dialog box are not performed, unless they are placed on the Marked Done tab of the Task Properties dialog box.

Note: In a similar fashion, any actions placed after the Mark Task as Done action in the action properties dialog box for a non-terminal action (as described in "Synchronous and Asynchronous Execution of Actions" on page 6-6) are not performed.

These are general guidelines you can use for marking tasks done:

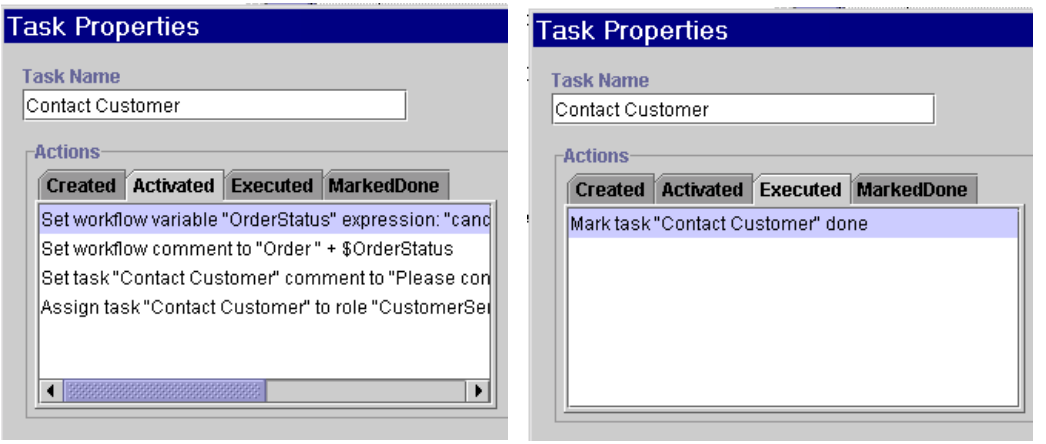
- If the task does not contain an action that can execute it, place the Mark Task as Done action as the last action on the Activated tab, as shown in the following figure.

Figure 6-5 Marking Non-Executed Tasks Done



- If the task does contain an action that can execute it, such as Assign Task to User, or Execute task, place the Mark Task as Done action as the last action on the Executed tab.

Figure 6-6 Marking Executed Tasks Done



- To perform a non-terminal action and its subactions asynchronously, place the Mark Task as Done action in the Task Properties dialog box, as the last action on the Activated or Executed tab as appropriate. (See Figure 6-1.)
- To perform a non-terminal action and its subactions synchronously, place the Mark Task as Done action as the last action in the action's properties dialog box. (See Figure 6-3.)

A detailed table with guidelines for using Task Properties dialog box tabs to perform each of the major activities identified in this guide is provided in the following section.

Guidelines for Action Placement in Task Nodes

In general, you should try to place actions in Task nodes only, and avoid using them in other types of nodes. You should also use shorter action lists in more Task nodes, rather than longer action lists in fewer Task nodes. Using a "shallow" or "flat" design approach ensures that the logic and transactional units of your workflows remain immediately apparent in their graphical representations.

In some cases, however, it will be necessary to specify more than one action in the same node, for example, when you need to specify various properties of a user-assigned task. (For more information, see "Setting Up Manual Tasks" on page 6-44.) Similarly, you may find it more efficient in some cases to simply add an action to nodes other than Tasks, or even to the action list in a Task node, rather than to

complicate the graphical flow with an extra Task node. This may be the case for simple activities such as setting variable values, setting task or workflow comments, or making audit entries.

The following table provides guidelines you can use as a reference for designing reusable node patterns that perform major common activities.

Table 6-1 Guidelines for Action Placement

Activity	In a Task node, place this action . . .	on this tab . . .	Mark the task done on this tab . . .	For more information, see . . .
Calling a Java Component	Perform Business Operation	Activated	Activated	"Calling a Business Operation" on page 6-78.
Calling an Executable Program	Call Program	Activated	Activated	"Calling an Executable Program on the Server" on page 6-76
Transforming XML Documents	XSL Transform	Activated	Activated	"Transforming XML Documents" on page 6-96
Sending E-mail	Send E-mail Message	Activated	Activated	"Sending E-Mail Messages" on page 6-72
Posting an XML Message to a JMS Topic or Queue	Post XML Event	Activate	Activated	"Posting an XML Message to a JMS Topic or Queue" on page 6-82
Using a Plug-In Action	custom action	Activated	Activated	

Table 6-1 Guidelines for Action Placement

Activity	In a Task node, place this action . . .	on this tab . . .	Mark the task done on this tab . . .	For more information, see . . .
Assigning a Manual Task	Set Task Comment (optional) Set Task Due Date (optional) Assign Task to User/Role/Using Routing Table	Activated	Executed	"Setting Up Manual Tasks" on page 6-44
	Send XML to Client (optional)	Executed	Callback Actions in the Send XML to Client dialog box	"Sending an XML Message to a Client Application" on page 6-58
Calling a Sub-Workflow Synchronously	Start Workflow	Activated	Actions tab in the Start Workflow dialog box	"Calling a Sub-Workflow" on page 6-36
Introducing a Time Delay	Set Task Due Date	Activated	Overdue Actions tab in the Set Task Due Date dialog box	"Setting a Task Due Date" on page 6-51
	Timed Event	Activated	Actions when triggered tab in the Timed Event dialog box	"Embedding a Timed Sequence" on page 6-32

Overview of Action Definition Tasks

Once you have created a workflow and configured required resources, you can begin to add actions to nodes and other objects. Adding and defining actions is an iterative process, which will undoubtedly entail adding additional nodes, creating additional variables, reconfiguring data and resources, defining XML documents, and so on. Defining actions involves the following tasks, that you can perform in any order:

Note: You may also want to familiarize yourself with the workflow expression language and the Studio's Expression Builder and XPath Wizard tools before beginning to define actions, since most actions require entering expressions into dialog box fields. Complete information on workflow expressions is available in Chapter 8, "Using Workflow Expressions."

- Define business operations as described in "Configuring Business Operations" on page 4-7, in order to define a Perform Business Operation action.
- Optionally, define users, as described in "Maintaining Users" on page 3-14, to specify users for an Assign Task to User, Assign Task Using Routing Table, or Send E-mail Message action.
- Optionally, define roles, as described in "Maintaining Roles" on page 3-20, to specify roles for an Assign Task to Role, Assign Task Using Routing Table, or Send E-mail Message action.
- Optionally, define business calendars, as described in "Administering Business Calendars" on page 3-4, for a Timed Event or Set Task Due Date action. Alternatively, you can import previously exported calendars from existing workflow packages; see procedures in "Importing Workflow Packages" on page 11-5.
- Add nodes to a template definition. Procedures are given in "Working with Nodes" on page 5-19.
- Create variables to be referenced by actions. Procedures are given in "Working with Variables" on page 5-28.
- Add actions to Task nodes, and if necessary, Start, Event, and Decision nodes. Procedures for adding actions are given in "Adding an Action" on page 6-17.
- Add actions to non-terminal actions.

- Add the Mark Task as Done action to Task nodes or non-terminal actions. Procedures are given in "Marking a Task Done" on page 6-25.
- Add Exception Handlers. Procedures for defining exception handlers are given in "Defining Exception Handlers" on page 9-4.
- Add actions to exception handlers. Procedures for defining exception handlers are given in "Defining Exception Handlers" on page 9-4.
- Add additional nodes to the workflow to refine your design to ensure that each discrete activity is represented by a node. Procedures for adding nodes are given in "Adding, Arranging, and Connecting Nodes" on page 5-20.
- Compose or import XML documents that are required for any of the actions that embed XML documents into workflows, such as Post XML Event, Send XML to Client, and Set Workflow Variable. Procedures for working with XML documents with action properties dialog boxes are given in Chapter 7, "Working with XML Entities."

Working with Actions

You add and update actions in Task, Decision, Event, Done and Start node properties dialog boxes, as well as in custom Exception Handlers, and the action properties dialog boxes listed in "Terminal Actions and Non-Terminal Actions" on page 6-6. Each dialog box has an Actions tab which you use for action placement and maintenance. This section describes the functions that are common to all actions.

Adding an Action

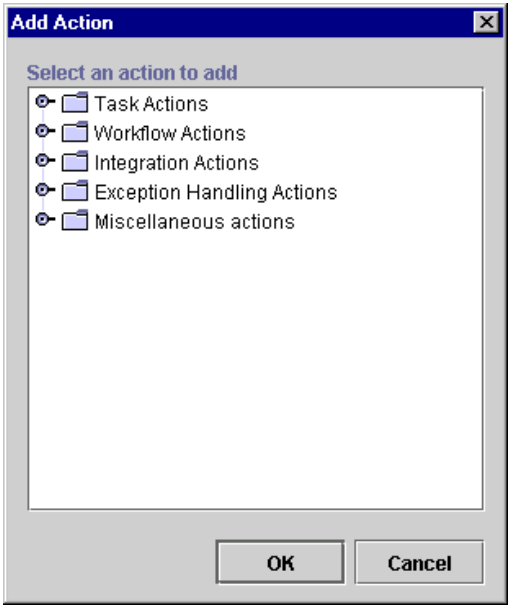
To add an action:

1. Do one of the following:
   - In the design area, double-click the node to which you want to add the action, and in the Properties dialog box that appears, select the appropriate Action tab and click Add.

- In the folder tree, expand the folder representing the object to which you want to add the action, right-click on the Actions folder, and, from the dialog box that appears, select Create Action.

The Add Action dialog box is displayed.

Figure 6-7 Add Action Dialog Box



Note: If a plug-in is defined for actions, this dialog box contains the new action.

2. Double-click an action type folder to expand it, select an action, and click OK. A properties dialog box specific to the selected action is displayed.
3. Complete the fields in the dialog box and click OK. Detailed procedures for each action type are provided in the remainder of this guide.

   The action is added to the properties dialog box for the node you selected and immediately appears in the folder tree under the folder representing the node or exception handler that contains it.

Updating an Action

To update an action:

1. Do one of the following:
   - In the properties dialog box which contains the action you want to update, select the appropriate action and click Update.
   - In the folder tree, expand the folder that contains the action you want to update, right-click the action, and from the pop-up menu, select Properties.

   The properties dialog box for the selected action is displayed.
2. Make changes to the action definition as desired, and click OK when done.

Deleting an Action

To delete an action:

1. Do one of the following:
   - In the properties dialog box which contains the action you want to delete, select the appropriate action and click Delete.
   - In the folder tree, expand the folder that contains the action you want to delete, right-click the action, and from the pop-up menu, select Delete.
2. When prompted by a warning message, click OK to confirm the deletion, or Cancel to cancel.

Copying an Action

You can copy actions within or between a workflow node within or between workflow template definitions (and templates) to create reusable design patterns. Since any properties that have been defined within the action are also copied, you can save time by simply making minor modifications to the action.

Note: If you are copying actions between template definitions, be sure that any variables referenced by the action have been created in the target template definition, and that other referenced objects, such as roles, users, or business calendars, are defined for the organization with which the template is associated.

To copy an action and its properties within or between nodes:

1. Do one of the following:
   - In the design area, double-click the node that contains the action you want to copy. In the Properties dialog box, select the appropriate Actions tab, right-click the action, and select Copy from the popup menu.
   - In the folder tree, expand the folder which contains the action you want to copy, right-click the action you want to copy, and, from popup menu, select Copy.
2. If you are pasting to a different node or template definition, do one of the following:
   - Click OK to close the node Properties dialog, and from the design area of the target template definition, right-click the node shape to which you will paste the defined action. In the Properties dialog box, select the appropriate Actions tab, right-click, and select Paste from the pop-up menu.
   - In the folder tree, expand the folder in which you want to paste the action, right-click the Actions folder, and select Paste from the pop-up menu.

   The properties dialog box for the action is displayed, with all settings copied from the source action.
3. In the action's properties dialog box, make any changes to the settings as necessary.
4. Click OK to save the changes and add the action to the target object.

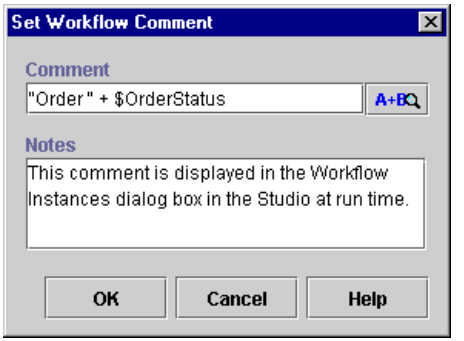
Reordering Actions

Actions are displayed in the sequence that they will be executed. To re-sequence actions, select an action from the list in the Properties dialog box and press the up or down arrow to move its position in the list.

Adding Notes to an Action

All action properties dialog boxes contain a Notes text box in which you can enter a comment about the action. This is helpful for other users who access the same workflow and need to understand the workflow logic or design.

Figure 6-8 Notes Text Box



Setting a Variable Value

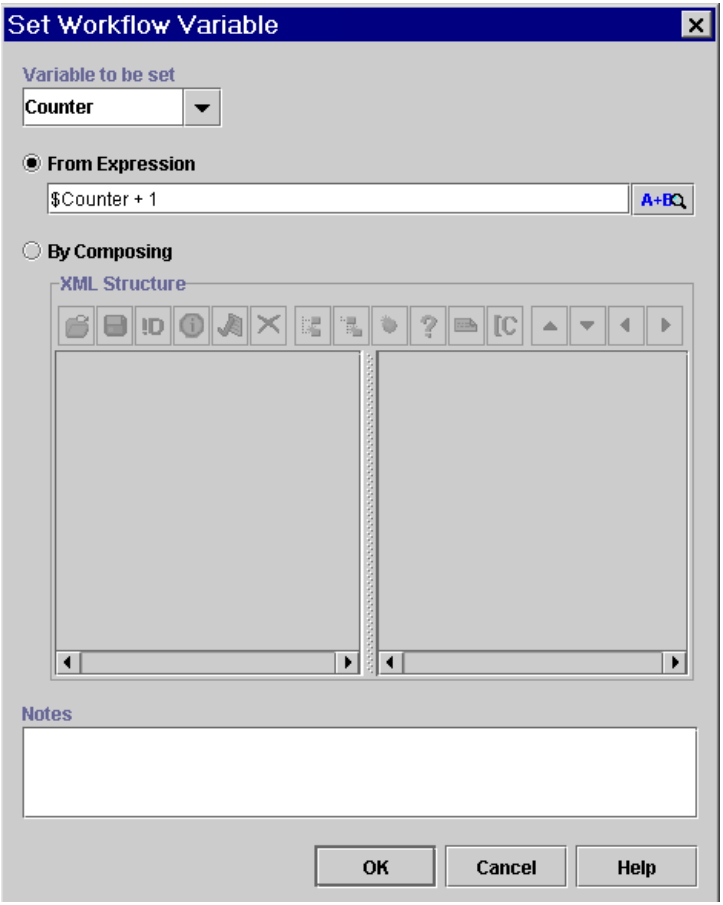
You use the Set Workflow Variable action to assign a value to an existing workflow variable at any point in the workflow, in any type of node. For example, to design a loop, you can use this action to increment a counter. For details about defining variables, see “Working with Variables” on page 5-28.

The value you assign to the variable can be an expression or a constant. The expression is evaluated at run time when the action is executed, and the result assigned to the variable you specify.

You can also use this action to compose or import an existing XML document and store the content in an XML or string type variable. You can then reuse the variable to reference the XML content in various places in a workflow, such as for posting an XML event action (see “Posting an XML Message to a JMS Topic or Queue” on page 6-82).

Note: For non-XML variables, you can also use the Variables tab of Start and Event node, Send XML to Client, and Exception Handler properties dialog boxes to set variable values.

Figure 6-9 Set Workflow Variable Dialog Box



To assign a value to a variable:

1. From the Workflow Actions folder in the Add Action dialog box, select Set Workflow Variable and click OK to display the Set Workflow Variable dialog box.
2. From the Variable to be set drop-down list, select an existing variable to which you will assign a value, or type one in. (For details, see "Working with Variables" on page 5-28.)
3. Choose one of the following options:
   - From Expression — enter an expression that is evaluated at run time to produce the value. For information on workflow expression components and syntax, see Chapter 8, "Using Workflow Expressions." To enter a constant, use the syntax described in "Using Literals" on page 8-2.
   - By Composing — set the value for an XML or string type variable by specifying an XML document. To create a new free-form document, click the Add Child button to begin adding nodes. To specify an existing XML document, click the Import button to load the document and edit as necessary. To create a new type-specified XML document, click the Set Content Type button to load a Schema document. For detailed procedures on all these options, see "Composing and Editing XML Documents" on page 7-2.

   The XML document you define is stored in the workflow template definition.

   Note: An XML document that is stored in a string type variable will first be converted to a string.
4. Click OK to add the Set Workflow Variable action.

Controlling Program Flow

In general, you should try to control the program flow through the use of nodes and connections. In some cases, however, you may need to specify alternate paths of execution from within nodes. For example, you may have an event defined that should only be allowed to be triggered up until a certain node in the workflow, and then should be disabled. Or you may need to automatically execute a task from within a different

node, depending on a condition. Most importantly, you can control whether sub-actions or even entire sub-workflows are to be performed synchronously or asynchronously by the placement of the Mark Task as Done action.

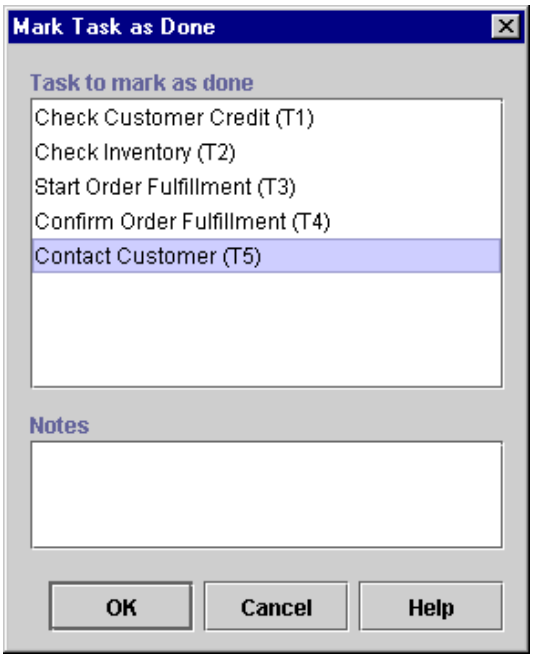
Thus, the following actions may be used for program control:

- Mark Task as Done—In addition to manual tasks executed by users, you need to mark task nodes as done that contain entirely automatic actions, to ensure that the flow proceeds correctly. This action is key to ensuring your workflows execute as desired, and is described in "Marking a Task Done" on page 6-25.
- Unmark Task Done—Unmarks a task done that has already been marked done. Described in "Unmarking a Task Done" on page 6-26.
- Cancel Workflow Event—Cancels an event that you specify, ensuring that it can no longer be triggered after this point in the workflow. Described in "Canceling a Workflow Event" on page 6-27.
- Mark Workflow Done—This action has the same effect as proceeding to a done node. Described in "Marking a Workflow Done" on page 6-28.
- Abort Workflow—Terminates a running instance and removes it from the database. Described in "Aborting a Workflow" on page 6-28.
- Execute Task—Automatically executes a task node and performs any actions listed on the Executed tab of the node. Described in "Executing a Task Automatically" on page 6-29.
- No Operation—This action does not actually have any effect on the workflow, but acts as a placeholder for the workflow designer. It is described in "Adding a Placeholder Action" on page 6-30.
- Timed Event—Can be used to create a time delay in a workflow. Described in "Embedding a Timed Sequence" on page 6-32.
- Set Task Due Date—Can be used to create a time delay in a workflow. Described in "Setting a Task Due Date" on page 6-51.

Marking a Task Done

Unless a user-assigned task is defined with a permission to allow a user to manually mark the task done at run time (for information, see “Defining Task Properties” on page 5-53), every Task node must specify the Mark Task as Done action either in its Task Properties dialog box, or in the Actions tab of a non-terminal action. For more information on using this action, see “Understanding Action Types and Placement” on page 6-5.

Figure 6-10 Mark Task as Done Dialog Box



To mark a task done:

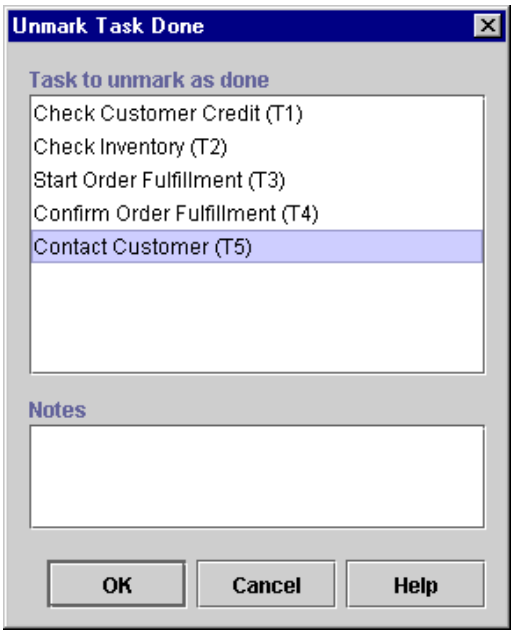
1. From the Task Actions folder in the Add Action dialog box, select Mark Task as Done and click OK to display the Mark Task as Done dialog box.
2. In the Task to mark as done field, select the task that is to be marked done when the action is executed. Only one task at a time can be selected.
3. Click OK to add the Mark Task as Done action.

Unmarking a Task Done

The Unmark Task Done action marks the task as not done. A task that is assigned this action becomes Active on the Worklist and can be executed again. Use this action to allow a task to be executed again in, for example, a loop or a parallel workflow.

Note: A task that is marked not done does not have its activated state triggered again. To trigger the activated state for the task, the workflow must initiate the task again. For details about task states, see "Understanding Task States" on page 5-54.

Figure 6-11 Unmark Task Done Dialog Box



To unmark a task done:

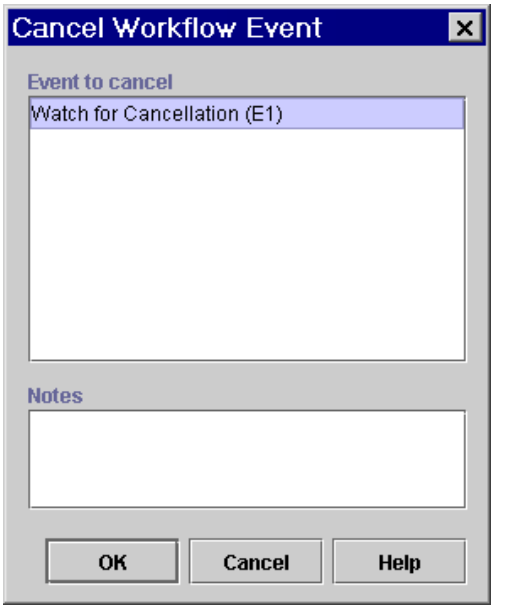
1. From the Task Actions folder in the Add Action dialog box, select Unmark Task as Done and click OK to display the Unmark Task Done dialog box.
2. In the Task to unmark as done field, select the task that is to be marked done when the action is executed. Only one task at a time can be selected.

3. Click OK to add the Unmark Task as Done action.

Canceling a Workflow Event

Use the Cancel Workflow Event action when you want to cancel an Event node or nodes that have been defined within a workflow. This action prevents an event from being triggered after a certain point in a workflow, so you can use it as an "expiry" mechanism for the Event node. For example, in an order processing workflow, you may have an event that waits for a cancellation notice from a customer, and performs some actions accordingly; to ensure that the cancellation cannot be issued after the order is shipped, you can define the Cancel Workflow action at the point where the shipping task is executed in the workflow, to disable the order cancellation event from being triggered.

Figure 6-12 Cancel Workflow Event Dialog Box



To cancel an event:

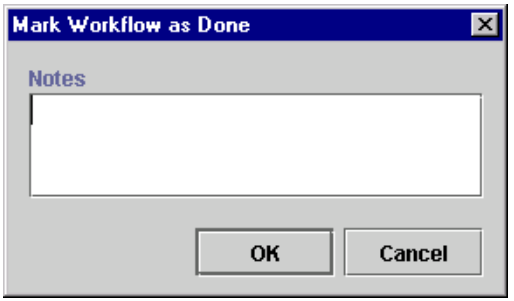
1. From the Miscellaneous Actions folder in the Add Action dialog box, select Cancel Workflow Event and click OK to display the Cancel Workflow Event dialog box.

2. In the Event to Cancel field, highlight the workflow event to be canceled. (Hold down the CTRL key to highlight more than one event.)
3. Click OK to add the Cancel Workflow Event action.

Marking a Workflow Done

The Mark Workflow as Done action marks the current workflow as done and serves the same purpose as a workflow reaching a Done node. It could be used, for example, in a Decision node, where you would like to skip all intervening actions before the Done node and terminate the workflow at that point.

Figure 6-13 Mark Workflow as Done



To mark a workflow as done:

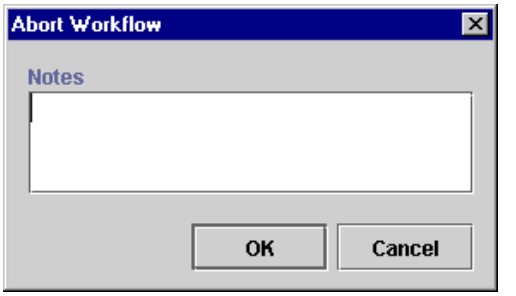
1. From the Workflow Actions folder in the Add Action dialog box, select Mark Workflow as Done and click OK to display the Mark Workflow as Done dialog box.
2. Click OK to add the Mark Workflow as Done action.

Aborting a Workflow

The Abort Workflow action permanently stops a workflow that is currently in process. This action can be used in exceptional conditions where an instance must be terminated.

Note: A workflow instance that is aborted will be deleted from the database, and cannot be monitored. To specify that a workflow should simply terminate, while retaining a record of the instance, you should use a Done node or the Mark Workflow as Done action.

Figure 6-14 Abort Workflow Dialog Box



To abort a workflow:

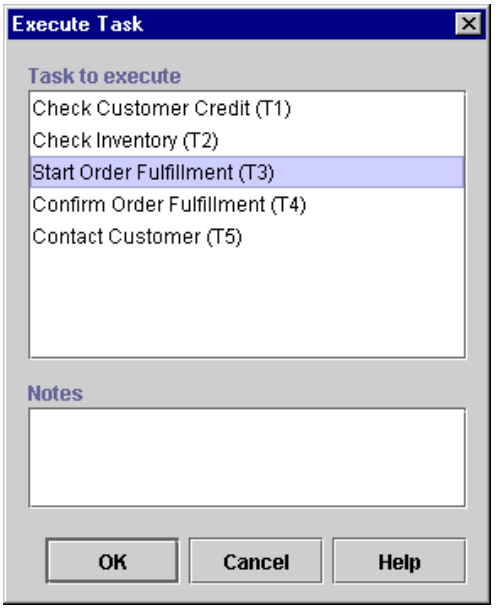
1. From the Workflow Actions folder in the Add Action dialog box, select Abort Workflow and click OK to display the Abort Workflow dialog box.
2. Click OK to add the Abort Workflow action.

Executing a Task Automatically

You can use the Execute Task action to allow the workflow, rather than a user, to execute any task in the workflow explicitly. This action places the task in the executed state, and performs all the actions listed in the Executed tab of the Task Properties dialog box. For information on task states, see “Understanding Task States” on page 5-54.

You may use this action, for example, from within an exception handler, when you want to re-execute actions in a task node after the actions in the exception handler have been completed. Or you might use it to create a loop that would be too difficult to represent graphically.

Figure 6-15 Execute Task Dialog Box



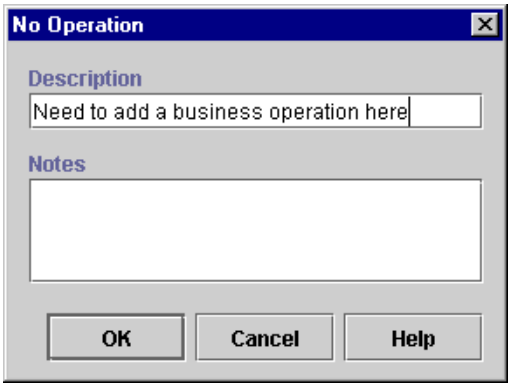
To execute a task automatically:

1. From the Task Actions folder in the Add Action dialog box, select Execute Task and click OK to display the Execute Task dialog box.
2. In the Task to execute field, select the task that you want to execute. Only one task at a time can be selected.
3. Click OK to add the Execute Task action.

Adding a Placeholder Action

The No Operation action does not affect the workflow; it simply acts as a placeholder to remind an analyst to add an action at a later time.

Figure 6-16 No Operation Dialog Box



To add a placeholder action:

1. From the Miscellaneous Actions folder in the Add Action dialog box, select No Operation and click OK to display the No Operation dialog box.
2. In the Description field, enter a descriptive name for the No Operation action, preferably a name that will remind you to add an action at a later time.
3. Click OK to add the No Operation action.

Using Timed Operations

The following actions allow you to set up timed operations:

- Timed Event—Specifies a series of sub-actions that should be performed and, optionally, re-executed, according to an exact time schedule. Can also be used to create a time delay in a workflow. Described in "Embedding a Timed Sequence" on page 6-32.
- Set Task Due Date—Specifies a date by which a task should be executed and, optionally, a series of sub-actions to be performed after the due date. Can also be used to introduce a time delay in a workflow. Described in "Setting a Task Due Date" on page 6-51.

Embedding a Timed Sequence

You can use the Timed Event action to create a timed sequence of actions that will be triggered at an exact time and date and be optionally re-executed according to a specified schedule.

About Execution Schedules

If you want to reschedule timed events, you must specify an execution stop method, according to two options:

- When the workflow is done. In this case, the timed event is re-executed until the workflow is done.
- When the task is done. In this case, the timed event is re-executed until the Task node that contains it is marked done. This option should only be used if two conditions are met:
  - The Timed Event action is specified in a Task node.
  - The containing Task node is marked done from outside the task node itself, such as from a Decision node, or another action. The node cannot be marked done from within the Task node, or from within the Timed Event dialog box's sub-actions tab, because marking the task done in this fashion would effectively terminate any repeated execution of the timed event.

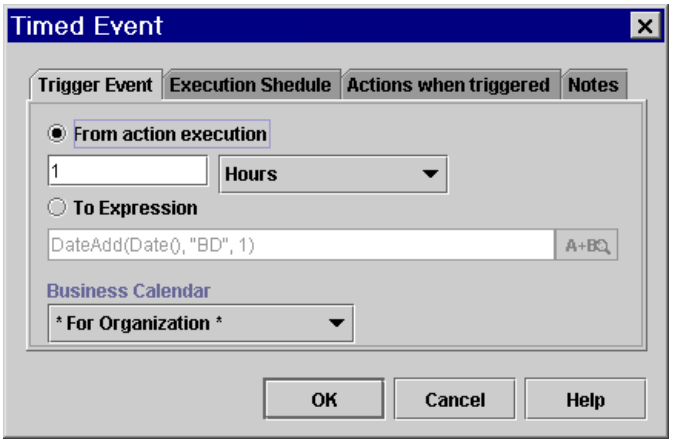
Executing Triggered Actions Asynchronously and Synchronously

Actions specified in a timed sequence can be performed in either a synchronous or asynchronous mode. In asynchronous mode, the sub-actions are performed in parallel, while the workflow continues to the next node. To set up the action in this way, you use the Timed Event in a Task node and mark the task containing the action done in the task node itself, or in any other type of node.

In synchronous mode, this action can serve to create a time delay in the workflow, as the workflow must wait until the trigger date and time are reached, and any sub-actions are performed, before it proceeds. To set up the action in this way, you must use the action in a Task node, and add the Mark Task as Done action to the Actions when Triggered tab of the Timed Event dialog box.

Note that in all of these cases, if you want to reschedule the timed event, you should stop the execution only when the workflow is done.

Figure 6-17 Timed Event: Trigger Event Tab



Defining a Timed Event

To define a timed event:

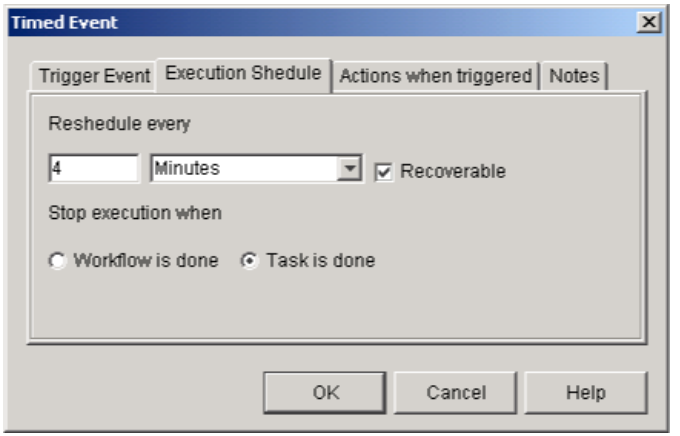
1. From the Miscellaneous Actions folder in the Add Action dialog box, select Timed Event and click OK to display the Timed Event dialog box.
2. On the Trigger Event tab, specify the time at which the sub-actions should be performed, by selecting one of the following options:
   - From action execution—Select this option to start the triggered actions at a specific interval of time after the timed event is executed, that is, when the Timed Event action is reached in a node's action list. Use the drop-down list to the right to select the unit of time, and in the field to the left, enter a number. For example, if you choose "4 Minutes," the sub-actions will be executed four minutes after the Timed Event action is reached in the workflow.
   - To Expression—Select this option to specify an absolute or relative date and time that you define by entering an expression in the field.
3. If you selected To Expression, enter a date function in the field to return a Java Date object, as follows:

- Use the `StringToDate()` function to specify an absolute date and time value. For details, see "StringToDate()" on page 8-19.
- Use the `DateAdd()` function to specify a value relative to a constant or variable base date and time. For details, see "DateAdd()" on page 8-21.

4. If you do not specify a business calendar as a parameter in the DateAdd() function, optionally, specify a business calendar as follows:
   - For Organization — use the business calendar assigned to the organization for which the workflow template definition instance will be executed.
   - For Assignee — only if the timed event is defined in a task node that assigns a manual task to a user, use the calendar that belongs to the user or role assigned the task that contains the event to be triggered.
5. Optionally, on the Execution Schedule tab, to repeat the triggered actions, in the Reschedule every field, enter a value to represent the interval of time that should elapse between each re-execution of the triggered actions, and select a unit of time from the drop-down list. For example, if you choose "4 Minutes," the sub-actions will be performed every four minutes after the initial triggering until specified execution stop.

   Set the Recoverable checkbox to indicate whether or not you would like a timed event to be recovered (that is, deferred until the server restarts) or skipped, if the server is not running at its scheduled start time.

Figure 6-18 Timed Event: Execution Schedule tab



6. If you specified an execution schedule, select a Stop execution when option to specify when the trigger should stop:
   - Workflow is done—Select to stop the trigger when the entire workflow is completed.
   - Task is done—Select to stop the trigger will stop when the task in which this action is specified is marked as done. Select this option only if the Timed Event action is specified in a Task node, and the Task node is marked done from outside the node, and outside the action. For more information, see "About Execution Schedules" on page 6-32.
7. On the Actions when triggered tab, click Add to open the Add Action dialog box to select and define the sub-actions to be executed when the event is triggered. If you want to ensure that sub-actions are performed synchronously, or you want to use this action to create a time delay in the workflow, add the Mark Task as Done action at the end of the action list here. In the Mark Task as Done dialog box, select the Task node containing the Timed Event action.
8. Click OK to add the Timed Event action.

Using Sub-Workflows

Several actions allow you to specify sub-workflows that are not represented in the design area. These are described below.

Note: You can also invoke a sub-workflow by posting an XML message to an internal JMS queue to start an event-triggered workflow. For more information, see "Posting an XML Message to a JMS Topic or Queue" on page 6-82.

- Start Workflow—Calls an entirely different workflow, that has been defined with a called Start. Described in "Calling a Sub-Workflow" on page 6-36.
- Evaluate Condition—Embeds a decision branch within a single node, by specifying alternate sub-actions to be performed according to a true or false result of a condition. Described in "Embedding a Conditional Sequence" on page 6-41.

- Timed Event—Specifies a series of sub-actions that should be performed and, optionally, re-executed, according to an exact time schedule. Described in "Embedding a Timed Sequence" on page 6-32.
- Invoke Exception Handler—Calls a series of sub-actions that are defined within an exception handler at a given point in the workflow, regardless of whether an exception has occurred. For information about this action, see Chapter 9, "Handling Workflow Exceptions."

Calling a Sub-Workflow

You may want to organize your processes into multiple workflows to break down large, complex workflows into multiple parts, or to use sub-workflows for processes that are only invoked according to specific conditions.

You can use the Start Workflow action to start another workflow from the current one. The workflow being started is referred to as a *sub-workflow*, or as a *called* or *child* workflow. The workflow that starts the sub-workflow is called the *calling* or *parent* workflow.

In order for a workflow to be called, the workflow template must contain at least one template definition that contains a Called Start node, is marked active, and specifies currently valid effective and expiry dates. Only one workflow template definition within that workflow template can be started at a time, this being the most effective of the active workflow template definitions. For an explanation, see "Working with Template Definitions" on page 5-7.

Passing Parameters

If the sub-workflow has any variables defined as *input* parameters, this means that it expects to populate those variables with values received from the parent workflow. Thus, in the Start Workflow action properties, you must specify values to be passed as input parameters to the sub-workflow; usually these values will be taken from corresponding variables defined in the parent workflow.

Similarly, if the sub-workflow has any variables defined as output parameters, this means that the sub-workflow will pass values it has obtained or calculated back to the parent workflow. Again, in the Start Workflow action properties, you need to specify parent workflow variables that are to receive those result values.

Executing the Sub-Workflow Asynchronously or Synchronously

The Start Workflow action can be used to start a workflow in both synchronous and asynchronous modes. In synchronous mode, the calling workflow waits for the sub-workflow to complete before proceeding to the next node. To set up the action this way, you must use the Start Workflow action in a Start or Decision node, or in a Task node, and mark the task done as a sub-action of the Start Workflow action, rather than in the Task node.

In asynchronous mode, the calling workflow does not wait for the called workflow to complete, but continues to the next node, so that both workflows execute in parallel. To set up the action in this way, you can use it in a Task node, and mark the task done in the Task node itself, or use it in any other type of node, such as a Decision, Start, or Event.

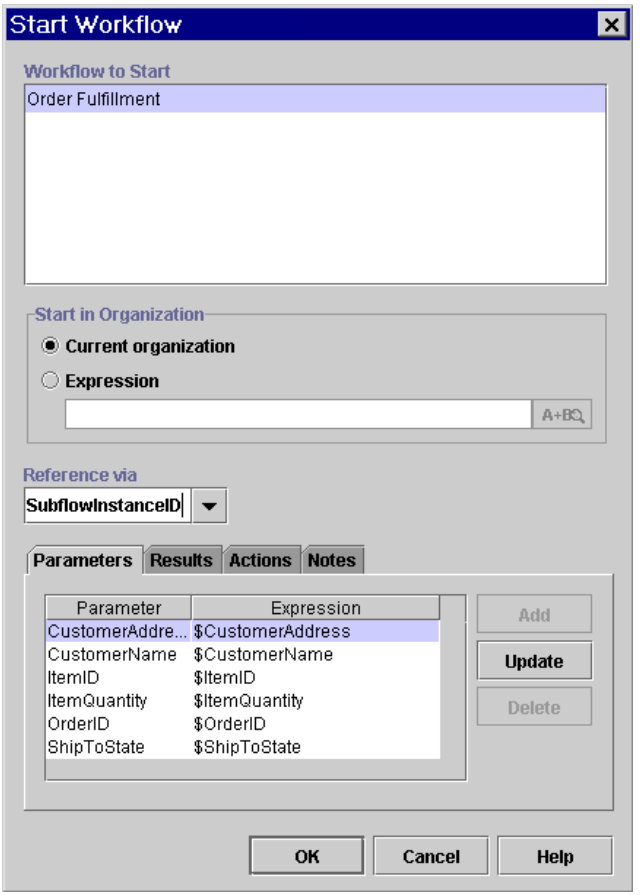
Note: You can also cause another workflow to be executed both asynchronously or synchronously by posting an XML message to an internal JMS queue to trigger an event-triggered workflow. Interaction between the two workflows, including parameter passing, is accomplished through XML/JMS messaging, and control of execution is accomplished by the use of Event nodes in the parent workflow to receive responses back from the sub-workflow. In fact, if you are running WebLogic Integration in a clustered environment, this method provides better load balancing control. For more information, see "Posting an XML Message to a JMS Topic or Queue" on page 6-82.

Regardless of whether or not the parent workflow waits for the sub-workflow to finish executing before proceeding, you can also define an optional set of sub-actions that are performed when the called workflow has completed.

Tracking the Sub-Workflow

You can assign the called workflow instance ID to a variable defined in the calling workflow. The instance ID is returned as a string, so be sure that the variable you create is defined as a String type (for procedures, see "Working with Variables" on page 5-28). The calling workflow can then use the reference variable in expressions using the `WorkflowVariable()` function to retrieve variables from the called sub-workflow. For details about using the instance ID in the `WorkflowVariable()` function, see "Obtaining Run-time Workflow Data" on page 8-13.

Figure 6-19 Start Workflow Dialog Box



To call a sub-workflow:

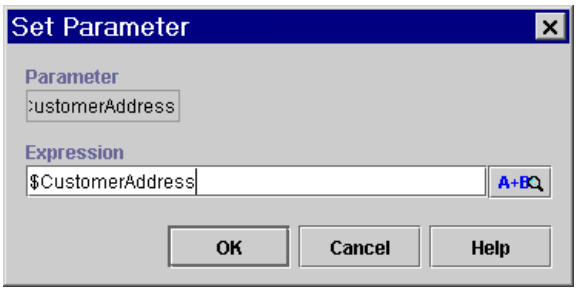
1. From the Workflow Actions folder in the Add Action dialog box, select Start Workflow and click OK to display the Start Workflow dialog box.
2. In the Workflow to Start field, select the workflow template to start when the action is executed. This field displays all workflow templates associated with the current organization that contain at least one template definition with a *called* start, currently valid effective and expiry dates, and that is marked active. When

you have selected a workflow, any variables that have been defined as input or output in that workflow appear on the Parameters and Results tabs at the bottom of the dialog box.

3. In the Start in Organization field, select one of the following options:
   - Current organization — select to start the workflow in the current organization (the organization selected from the Organization drop-down list above the folder tree).
   - Expression — select to start the workflow for an organization determined at run time by a workflow expression, and enter a string, surrounded by quotation marks, that specifies the organization, or an expression that will be evaluated at run time and become the name of an organization. For example, this could be a variable that has previously stored organization information from an incoming XML message received by another node.
4. Optionally, from the Reference via field, select a variable from the list of variables available for the current workflow. When the called workflow is instantiated, its instance ID is assigned automatically to the selected variable.

   Note: The variable must be a string type.
5. From the Parameters list on the Parameters tab, select a parameter and click Update to display the Set Parameter dialog box, which you use to specify a value to pass to the called workflow.

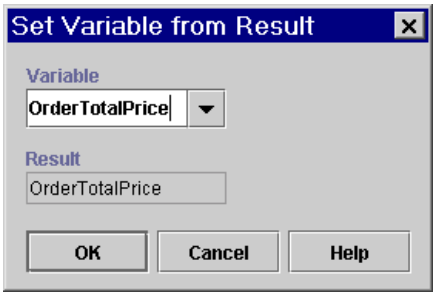
Figure 6-20 Set Parameter Dialog Box



6. In the Expression field, enter the value to be passed to the sub-workflow as an expression. The value will typically consist of a corresponding variable value in the parent workflow.

7. Click OK. The parameter and its value appear in the list on the Parameters tab of the Start Workflow dialog box.
8. Repeat steps 5 to 7 for all parameters in the list.
9. From the Results list on the Results tab, select a result and click Update to display the Set Variable from Result Parameter dialog box, which you use to specify the variable in the parent workflow in which you want to store the value returned by the sub-workflow.

Figure 6-21 Set Variable from Result Dialog Box



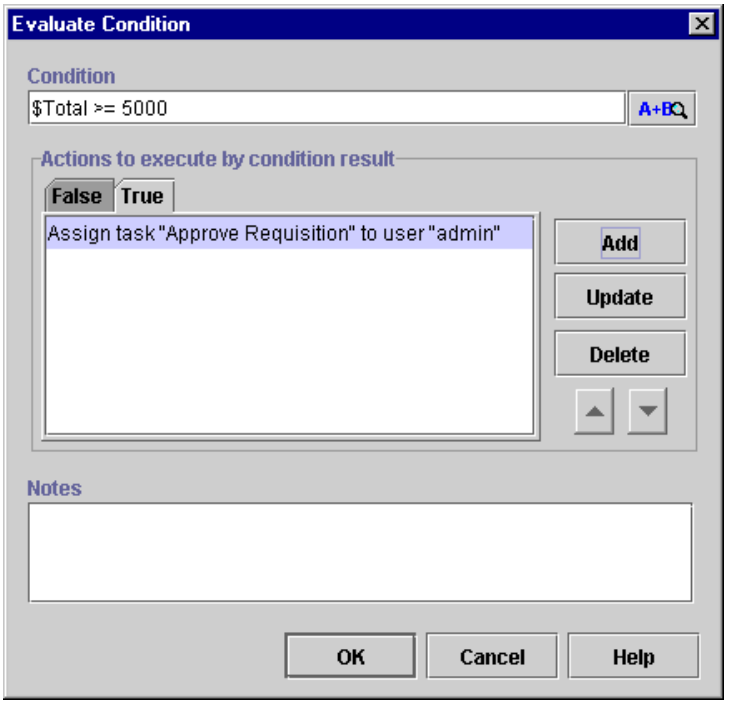
10. From the Variable drop-down list, select a variable in which to store the value returned by the corresponding output variable defined in the sub-workflow.
11. Click OK. The result and its value appear in the list on the Result tab of the Start Workflow dialog box.
12. Repeat steps 8 to 10 for all results in the list.
13. Optionally, select the Actions tab and click Add to display the Add Action dialog box to select and define sub-actions to be performed when the sub-workflow has completed. If you want the sub-workflow to complete before any other nodes are executed in the parent workflow, be sure to add the Mark Task as Done action to this tab. In the Mark Task as Done dialog box, select the Task node in which you have specified the Start Workflow action.
14. Click OK to add the Start Workflow action.

Embedding a Conditional Sequence

The Evaluate Condition action functions in the same way as a Decision node, that is, to evaluate a conditional expression at run time and perform alternative sequences of sub-actions depending on the result. Normally, you should use a Decision node to accomplish this type of flow. However, there may cases where you want to embed a conditional sub-workflow within a main workflow, by specifying a conditional set of actions inside another node, rather than as a distinct node on its own. This is necessary in cases where there is no other way to specify a condition, such as in an Exception Handler.

You define a series of actions to be performed if the condition evaluates to true, and/or a series of actions to perform if the condition evaluates to false.

Figure 6-22 Evaluate Condition Dialog Box



To evaluate a condition:

1. From the Miscellaneous Actions folder in the Add Action dialog box, select Evaluate Condition and click OK to display the Evaluate Condition dialog box.
2. In the Condition field, enter a valid workflow condition that will be evaluated at run time to determine which set of sub-actions to perform. For more information on constructing expressions, see Chapter 8, "Using Workflow Expressions."
3. On the False and/or True tab, click Add to invoke the Add Action dialog box and select and define the sub-actions to be executed if the condition evaluates to false or true, respectively, by using the following buttons:
4. Click OK to add the Evaluate Condition action.

Monitoring Run-Time Status

You can use two actions for workflow monitoring purposes:

- Make Audit Entry—Adds an entry whose content you specify to an audit log maintained for running workflow instances. Described in "Making an Audit Entry" on page 6-42.
- Set Workflow Comment—Displays explanatory or instructive comment to an administrator monitoring a workflow instance at run time in the Studio. Described in "Setting Up a Workflow Comment" on page 6-43.

Making an Audit Entry

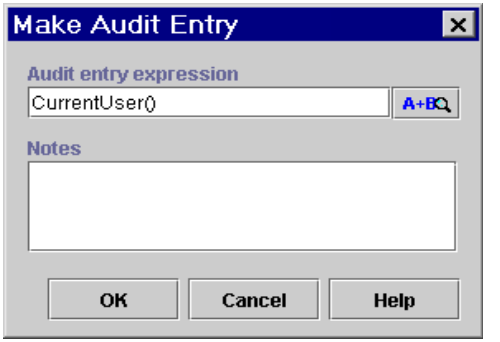
WebLogic Integration provides a default audit facility whereby all major user interactions and changes are published to a JMS topic and to a log file, `myserver.log`, located in the `logs` directory of the active WebLogic Integration domain on the server.

You can use the Make Audit Entry action in any node in your workflow to define additional run-time workflow information you would like to publish to the audit log over the course of a workflow. The date and time of each entry is noted, and you define an expression that will provide the data you want to record at run time, containing

variables, functions or constants. For example, you could use the CurrentUser() function in a task that is assigned to a role, to record the user who executed it. (For information on functions that return run-time workflow information, see "Obtaining Run-time Workflow Data" on page 8-13.)

Note: Auditing must be enabled in the Template Definition properties dialog box for the workflow for this action to take effect. For further information, see "Creating a Workflow Template Definition" on page 5-8.

Figure 6-23 Make Audit Entry Dialog Box



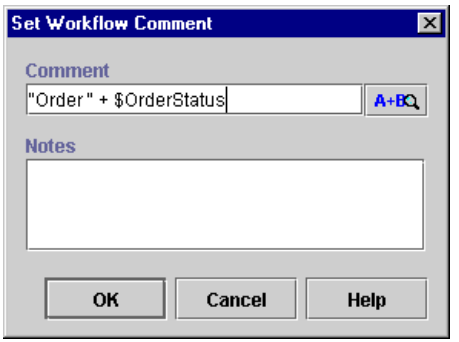
To make an audit entry:

1. From the Miscellaneous Actions folder in the Add Action dialog box, select Make Audit Entry and click OK to display the Make Audit Entry dialog box.
2. In the Audit entry expression field, enter an expression that will be evaluated to to produce the audit entry information at run time. For more information on constructing expressions, see Chapter 8, "Using Workflow Expressions."
3. Click OK to add the action.

Setting Up a Workflow Comment

You can use the Set Workflow Comment action to specify a comment for the workflow instance. The comment will be displayed in the Comment column of the Workflow Instances dialog box of the Studio when the action is executed at run time (for more information, see "Working with Workflow Instances" on page 10-2). The text is typically informative and indicates the status of the workflow at that point.

Figure 6-24 Set Workflow Comment Dialog Box



To set a workflow comment:

1. From the Workflow Actions folder in the Add Action dialog box, select Set Workflow Comment and click OK to display the Set Workflow Comment dialog box.
2. In the Comment field, enter an expression that will be evaluated at run time to produce the comment. The expression will typically consist of strings and variables. For more information on expressions and their syntax, see Chapter 8, "Using Workflow Expressions."

   Note: Workflow comments are limited to a maximum of 254 characters. The comment length is not determined until run time because the length of expressions may vary. Comments exceeding the 254-character limit are truncated at run time with no warning.
3. Click OK to add the Set Workflow Comment action.

Setting Up Manual Tasks

You can use the following Task actions to assign manual tasks or interact with WebLogic Integration Worklist or custom client application users.

- Assign Task to User—Assigns a manual task to a specific user or to a user based on workload balancing. Described in "Assigning a Task to a User" on page 6-46.

- Assign Task to Role—Assigns a manual task to a role. Described in "Assigning a Task to a Role" on page 6-47.
- Assign Task Using Routing Table—Assigns the same task to different users or roles, depending on conditions that you specify. Described in "Assigning a Task Using a Routing Table" on page 6-49.
- Set Task Due Date—Specifies a due date by which the assignee should execute the task. Described in "Setting a Task Due Date" on page 6-51.
- Set Task Comment—Displays a comment or instruction for the user executing the task at run time. Described in "Setting a Task Comment" on page 6-54.
- Set Task Priority—Displays a priority level for the task to the user at run time. Described in "Setting a Task Priority" on page 6-56.
- Unassign Task—Unassigns a task. Described in "Unassigning a Task" on page 6-57.
- Send XML to Client—Sends an XML message to the Worklist or custom client application, to display a form or prompt to a user, or call a program or custom extension on the client. Described in "Sending an XML Message to a Client Application" on page 6-58.

Guidelines for Placement of Task Actions

When assigning manual tasks, follow these guidelines for action placement:

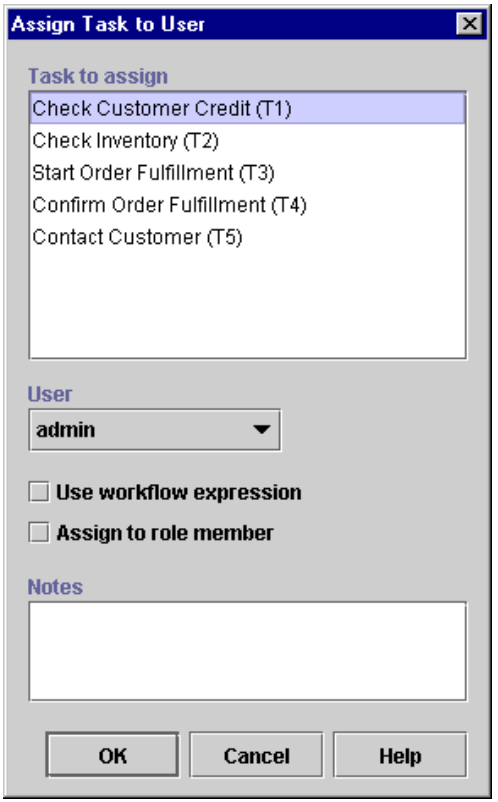
- You must place all actions pertaining to the same task in the same Task node.
- To ensure that a comment or due date is displayed to the user who views the task *before* he or she executes it, you should place the Set Task Comment and Set Task Due Date actions before the Assign Task to User/Role/Using Routing Table actions.
- Mark the task done on the Executed tab of the task Properties dialog box.
- If you want to send an XML message to the client application to display a prompt message or call additional software components on the client, first assign the task to a user or role, and then place the Send XML to Client action on the Executed tab of the Task Properties dialog box.

Assigning a Task to a User

Use the Assign Task to User action to assign a specified task to an individual user. Assigning a task to a user causes a notification to be sent to the user who views and executes the task from the Worklist or custom client application. For more information about the Worklist application, see *Using the WebLogic Integration Worklist*.

You can specify a particular user to whom to assign the task, or you can specify a role member, which causes the system to determine the user with the fewest pending tasks among all users belonging to a particular role at run time. The system then assigns the task to that user. The user or role can be specified as a constant or taken from a workflow expression to be provided at run time.

Figure 6-25 Assign Task to User Dialog Box



To assign a task to a user:

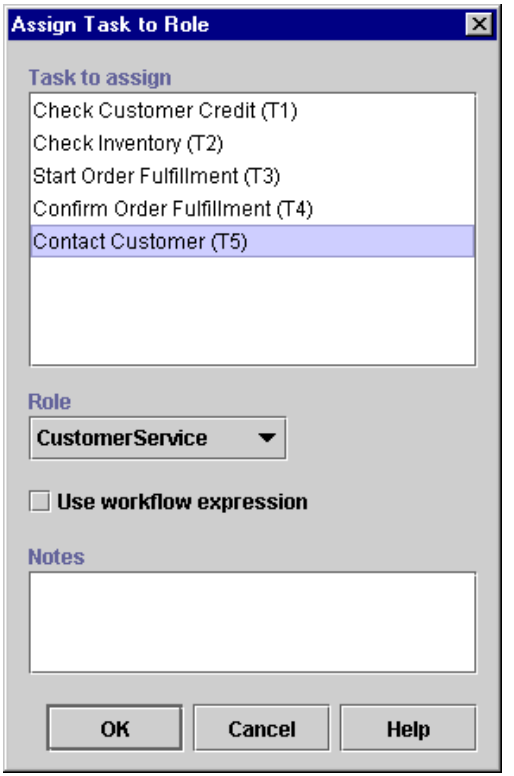
1. From the Task Actions folder in the Add Action dialog box, select Assign Task to User and click OK to display the Assign Task to User dialog box.
2. In the Task to assign field, select the task that you want to assign. Only one task at a time can be selected.
3. To specify a user, select from the following options:
   - To specify a user name, select the name from the User drop-down list. Only users associated with the current organization appear in this list.
   - To assign the task to a role member, select the Assign to role member, and in the User in Role drop-down list that appears, select the Role name. Only roles defined for the current organization appear in this list. At run-time the user belonging to that role who has the fewest tasks assigned to him/her will be assigned the task.
   - To specify a user whose identity will be determined at run time, select Use Workflow Expression, and in the User field, enter an expression that returns the appropriate user at run-time.
4. Click OK to add the Assign Task to User action.

Assigning a Task to a Role

Use can use the Assign Task to Role action when it is not desirable to assign a task to a particular user. Assigning a task to a role allows any user who belongs to the role to view and execute the task, although there is no explicit notification of the task in the Worklist.

To assign a task to a role, you specify the role name or provide an expression to determine the role at run time.

Figure 6-26 Assign Task to Role Dialog Box



To assign a task to a role:

1. From the Task Actions folder in the Add Action dialog box, select Assign Task to Role and click OK to display the Assign Task to Role dialog box.
2. In the Task to assign field, select the task that you want to assign. Only one task at a time can be selected.
3. To specify a role, select from the following options:
   - To specify a role name, select the name from the Role drop-down list. Only roles defined in the current organization appear in this list.
   - To specify a role whose identity will be determined at run time, select Use Workflow Expression, and in the Role field, enter an expression that returns the appropriate role at run-time. For more information on functions and

expressions, and expression syntax, see Chapter 8, "Using Workflow Expressions."

4. Click OK to add the Assign Task to Role action.

Assigning a Task Using a Routing Table

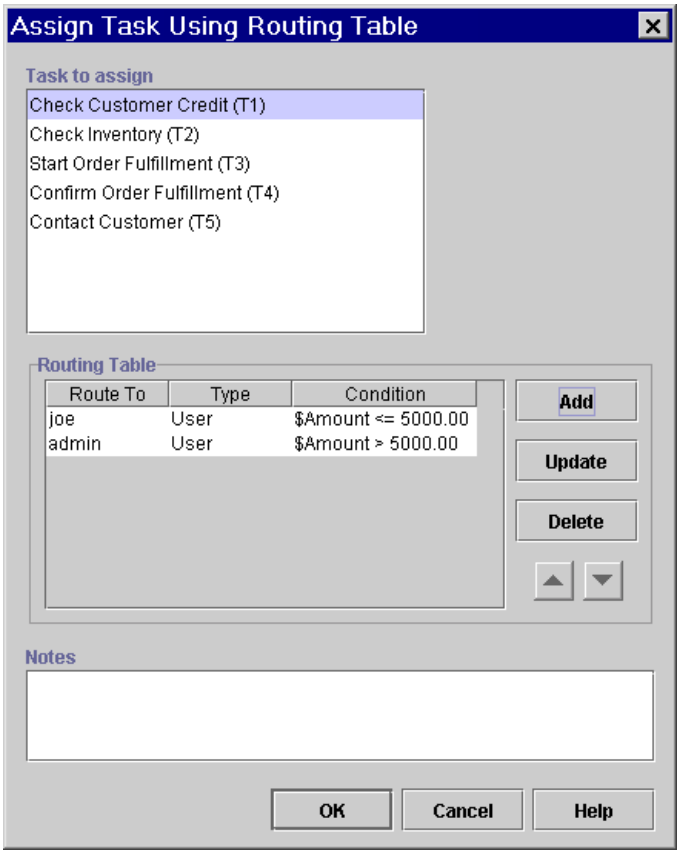
You can assign a task to different users or roles depending on a set of conditions. A routing table consists of a sequence of one or more routing conditions. A routing condition specifies a potential assignee (user or role) for the task, together with a specification of the conditions under which the assignment should be made. When the action is executed at run time, the system evaluates each condition in sequence until it encounters one that yields a True result. If such a condition is encountered, the task is assigned to the corresponding user or role, and any subsequent conditions are ignored.

Note: If no routing condition is satisfied, a run-time exception occurs.

If you would like to route *all* tasks for a particular user or role for a specific period of time, instead of when certain conditions are met, create a routing specification for an organization. For more information, see "Administering Task Routings" on page 3-29.

Note: The Routing feature reroutes only tasks assigned to a user; it does not reroute tasks assigned to a role. Rerouted tasks can, however, be directed to another user, user in role, or role. To reroute tasks assigned to a role, see "Changing the Mapping for Roles" on page 3-24.

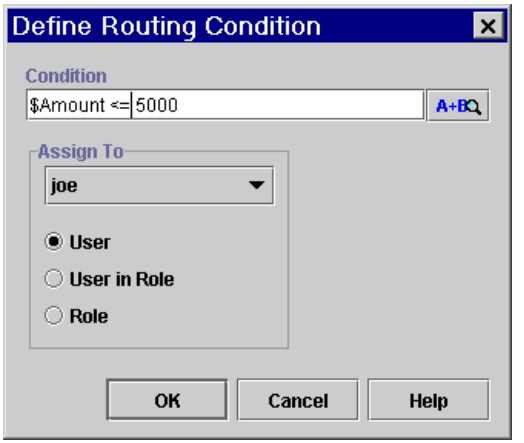
Figure 6-27 Assign Task Using Routing Table Dialog Box



To assign a task using a routing table:

1. From the Task Actions folder in the Add Action dialog box, select Assign Task Using Routing Table and click OK to display the Assign Task Using Routing Table dialog box.
2. In the Task to assign field, select the task that you want to assign. Only one task at a time can be selected.
3. Next to the Routing Table field, click add to display the Define Routing Condition dialog box.

Figure 6-28 Define Routing Condition Dialog Box



4. In the Condition field, enter a valid workflow conditional expression that yields a logical result (True or False). For information on constructing expressions, see Chapter 8, "Using Workflow Expressions."
5. In the Assign To field, select one of the following options: User, User in Role, or Role.
6. In the drop-down list, select the desired user or role name.
7. Click OK to add the routing condition to the routing table.
8. Continue to add as many routing conditions as required. Use the Delete button to delete a routing, the Update button to edit a routing, and the arrow keys to re-order routings in the table.
9. Click OK to add the Assign Task Using Routing Table action.

Setting a Task Due Date

You can use the Set Task Due Date action to set the due date by which a task should be executed. The due date is displayed to the Worklist or custom client user to whom the task has been assigned. You can express the due date in minutes, hours, days, weeks, or months, or as an expression to determine the date and time at run time.

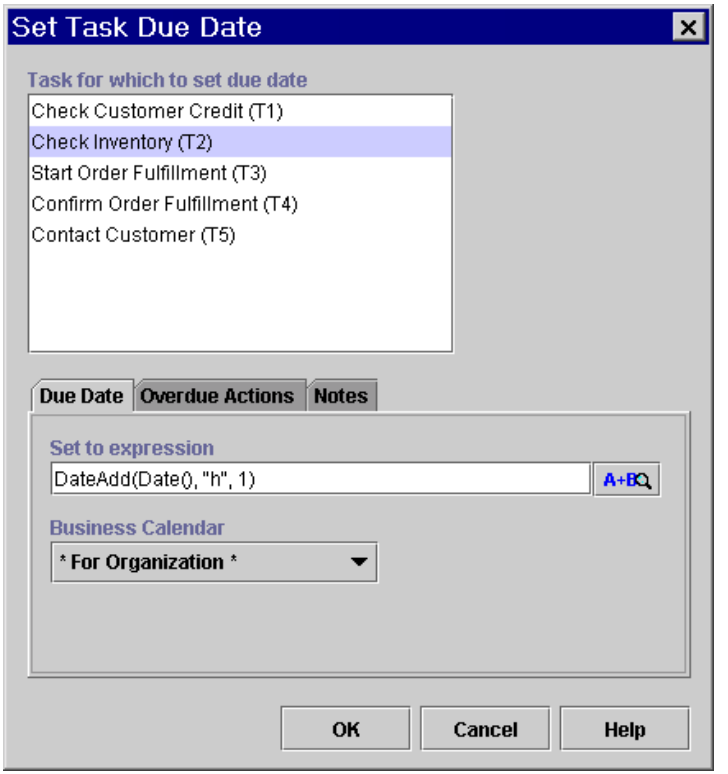
You can also use this action for non-manual tasks, without the use of the Assign Task to User action, to specify actions to be performed in parallel after a certain date, or to introduce a time delay into a workflow.

Executing Overdue Actions Asynchronously and Synchronously

If the task is not executed by the due date, you can specify sub-actions that should be performed after the due date, in either a synchronous or asynchronous mode. In asynchronous mode, the sub-actions are performed in parallel, while the workflow continues to the next node. To set up the action in this way, you mark the task containing the action done in the Task node itself. This is normally how the action is used for manually assigned tasks.

In synchronous mode, this action can serve to create a time delay in the workflow, as the workflow must wait until the due date is reached, and any overdue actions are performed, before it proceeds. To set up the action in this way, you place the action in a Task that is assigned to a user or not, and mark the task containing the action done by adding the Mark Task as Done action to the Overdue Actions tab of the Set Task Due Date dialog box.

Figure 6-29 Set Task Due Date Dialog Box



To set a task due date:

1. From the Task Actions folder in the Add Action dialog box, select Set Task Due Date and click OK to display the Set Task Due Date dialog box.
2. In the Task for which to set due date field, select the task that is to have its due date set when the action is executed. Only one task at a time can be selected.
3. On the Due Date tab, in the Set to expression field, enter an expression to specify an absolute or relative date and time. The expression must return a Java Date object, so you must use a function, as follows:
   - Use the `StringToDate()` function to specify an absolute date and time value. For details, see "StringToDate()" on page 8-19.

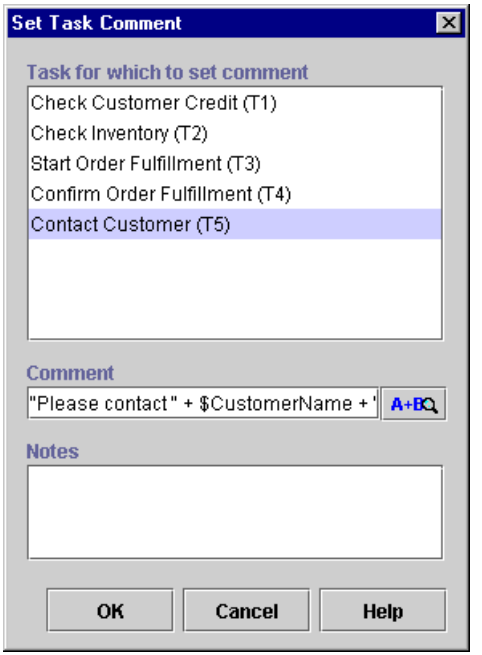
- Use the DateAdd() function to specify a value relative to a constant or variable base date and time. For details, see "DateAdd()" on page 8-21.

4. If you do not specify a business calendar as a parameter in the DateAdd() function, optionally, specify a business calendar as follows:
   - For Organization — use the business calendar assigned to the organization for which the workflow template definition instance will be executed.
   - For Assignee — only if the due date is defined in a task node that assigns a manual task to a user, use the calendar that belongs to the user or role assigned the task that contains the Set Due Date action.
5. Optionally, select the Overdue Actions tab and click Add to display the Add Action dialog box to select and define sub-actions to be performed when the due date is reached. If you want the workflow to wait until the due date is reached before proceeding, use the action in a Task node, and be sure to add the Mark Task as Done action to this tab. In the Mark Task as Done dialog box, select the Task node in which you have specified the Set Due Date action.
6. Click OK to add the Set Task Due Date action.

Setting a Task Comment

You can use the Set Task Comment action to set a comment for the task instance, which displays a text message to a user viewing the task in the Worklist or the Studio when the action is executed at run time. The text typically provides information or instructions for manual work which the user is asked to perform. The text message is displayed in the Comment column next to the task in the task list in the Worklist application, or in the Workflow Instances or Worklist dialog boxes in the Studio. (For more information, see Chapter 10, "Monitoring Workflows."

Figure 6-30 Set Task Comment Dialog Box



To set a task comment:

1. From the Task Actions folder in the Add Action dialog box, select Set Task Comment and click OK to display the Set Task Comment dialog box.
2. In the Task for which to set comment, select the appropriate task. Only one task at a time can be selected.
3. In the Comment field, enter an expression that will be evaluated at run time to produce the comment. The expression will typically consist of strings and variables. For more information on expressions and their syntax, see Chapter 8, "Using Workflow Expressions."

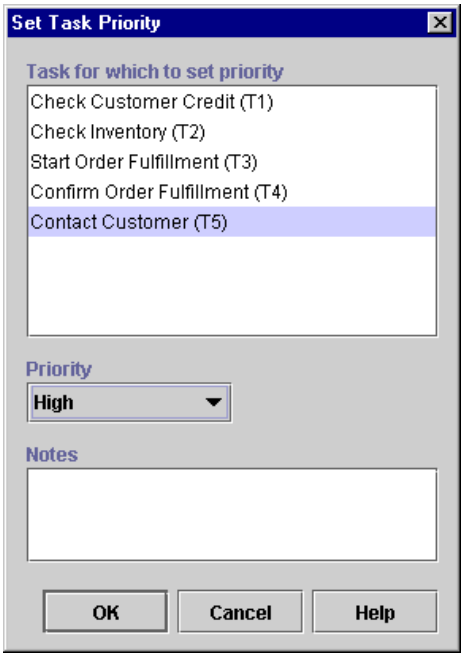
   Note: Task comments are limited to a maximum of 254 characters. The comment length is not determined until run time because the length of expressions may vary. Comments exceeding the 254-character limit are truncated at run time with no warning.
4. Click OK to add the Set Task Comment action.

Setting a Task Priority

You can use the Set Task Priority action set a task priority to Low, Medium, or High. The priority has no effect on how the node or its actions are executed at run time, but is simply displayed to Worklist users, who can execute and sort their tasks accordingly.

Since you can set a priority for the current task in the Task's properties dialog box, you may wish to use this action as the result of a condition to specify the priority of a task elsewhere in the workflow.

Figure 6-31 Set Task Priority Dialog Box



To set a task priority:

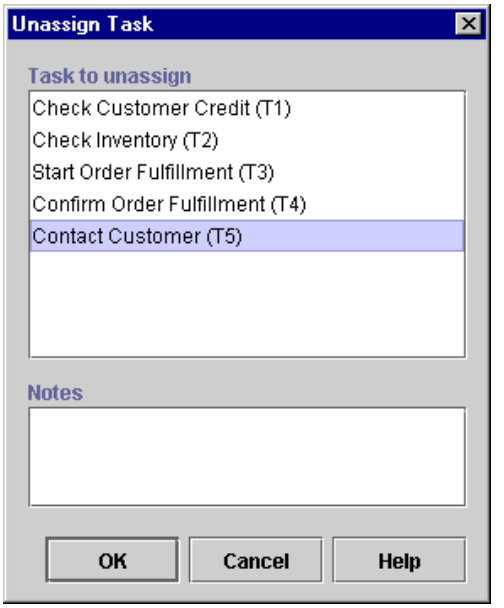
1. From the Task Actions folder in the Add Action dialog box, select Set Task Priority and click OK to display the Set Task Priority dialog box.
2. In the Task for which to set priority, select the appropriate task. Only one task at a time can be selected.
3. From the Priority drop-down list, select Low, Medium, or High.

4. Click OK to add the Set Task Priority action.

Unassigning a Task

You can use the Unassign Task action to remove the current task assignment. The task is no longer assigned to a user or role. You may wish to unassign a task as a result of a condition.

Figure 6-32 Unassign Task Dialog Box



To unassign a task:

1. From the Task Actions folder in the Add Action dialog box, select Unassign Task and click OK to display the Unassign Task dialog box.
2. In the Task to unassign field, select the task that you want to unassign. Only one task at a time can be selected.
3. Click OK to add the Unassign Task action.

Sending an XML Message to a Client Application

Once you have assigned a task to a user, you can use the Send XML to Client action to communicate between a workflow and the Worklist or custom client application by sending an XML document to the client. The client application must be programmed to identify the XML document, perform the appropriate action, and return an XML document in response to the workflow. For the Worklist application, you can send XML messages to display message prompts and forms to users, and to call custom components or executable programs on the client system. For information on developing custom client applications, see *Programming BPM Client Applications*.

Note that the Send XML to Client action does not actually specify the client machine or application to which the XML is being sent. Thus, you must first assign a task to a user or role, and the XML message will be sent to the client that executes the task. For more information, see "Setting Up Manual Tasks" on page 6-44.

Sending a Message Asynchronously or Synchronously

You can send an XML message to a client application asynchronously or synchronously. In synchronous mode, the workflow waits for a response from the client before proceeding to the next node. To set up the action this way, you must use mark the Task containing the action done as a sub-action on the Callback Actions tab of the Send XML to Client dialog box, rather than in the Task node.

In asynchronous mode, the workflow does not wait for a response from the client, but continues to the next node, while any operations on the client execute in parallel. To set up the action in this way, mark the task done in the Task node itself.

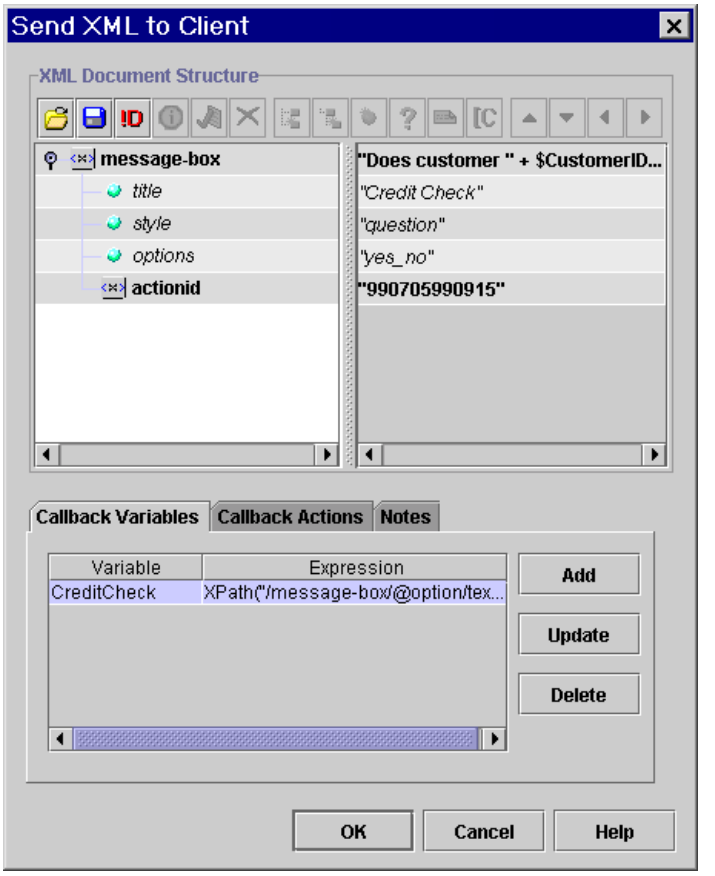
Extracting Data

If your application responds to the workflow by sending a return XML message, you must create variables to store the data returned by the response document (see "Working with Variables" on page 5-28 for procedures), and initialize those variables by using XPath expressions (or dot notation) to retrieve data values from the response XML document. If your application uses JMS properties in its messages, you can also retrieve this data by using the `EventAttribute()` function. (For more information, see "Extracting Run-Time Event Data" on page 8-7.)

Defining the Send XML to Client Action

The following procedure is generalized for any non-Worklist application. For complete information on sending an XML message to the Worklist, see “Sending an XML Message to the Worklist Application” on page 6-61. For detailed information on working with type-specified XML documents, see “Working with Type-Specified Documents” on page 7-11.

Figure 6-33 Send XML to Client Dialog Box



To send an XML message to a client:

1. From the Integration Actions folder in the Add Action dialog box, select Send XML to Client and click OK to display the Send XML to Client dialog box. A default XML document structure with a root element and `actionid` element is created.

   Note: The system-generated `actionid` element and its value are used at run time to identify the Send XML to Client action. The `actionid` element must be the first child element of the root element. Do not move, delete or edit the `actionid` element, and do not add any sub-elements to it.

2. To specify the XML Document Structure, do one of the following:

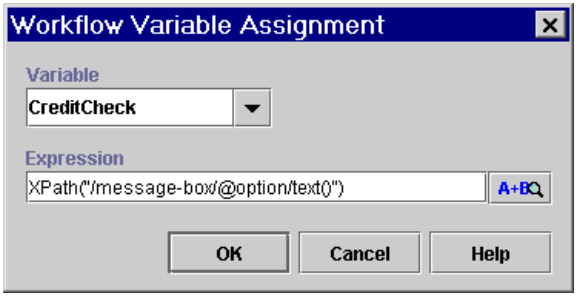
   - To create a new free-form document, begin composing the document by clicking the Add Child button to begin adding nodes.
   - To specify an existing XML document, load the document by clicking the Import button.
   - To create a new type-specified XML document, load the appropriate Schema document by clicking the Set Content Type button. The `actionid` value is removed and is reset when you save the action in Step 9.

   For detailed procedures for all these options, see "Composing and Editing XML Documents" on page 7-2.

   The XML document you define is stored in the workflow template definition.

3. Select the Callback Variables tab to specify the variables that will receive the response from the client, and click Add to display the Workflow Variable Assignment dialog box.

Figure 6-34 Workflow Variable Assignment Dialog Box



4. In the Expression field, enter the expression that is evaluated at run time to extract the data from the response XML document, by doing one of the following:
   - To capture incoming JMS header data, use an EventAttribute() function. For information, see "EventAttribute()" on page 8-8.
   - To capture incoming XML content, use an XPath() function (for information, see "XPath()" on page 8-10), or the dot notation for XML elements that return strings (for information, see "XML Element Dot Notation" on page 8-12). You can also use the Expression button  to invoke the XPath Wizard, from which you can generate XPath expressions automatically from a sample incoming document. For information, see "Creating XPath Expressions Using the XPath Wizard" on page 8-31.
5. Click OK. The variable initialization appears in the list on the Callback Variables tab of the Send XML to Client dialog box.
6. Repeat steps 4 to 5 for all data items that need to be captured from the response document.
7. Select the Callback Actions tab and click Add to display the Add Action dialog box to specify any sub-actions to be performed when a reply is received from the client. If you want the workflow to wait for the response from the client before proceeding to other nodes in the workflow, be sure to add the Mark Task as Done action to this tab, at the end of the action list. In the Mark Task as Done dialog box, select the Task node in which you have specified the Send XML to Client action.
8. If you are using a type-specified document, you are prompted to add the `actionid` value. Click Yes to update the document.
9. Click OK again to save the action definition.

Sending an XML Message to the Worklist Application

Note: The Worklist client application is being deprecated as of this release of WebLogic Integration. For information about the features that are replacing it, see the *BEA WebLogic Integration Release Notes*.

The Worklist client application is designed to perform an action in response to an XML document that conforms to four pairs of predefined Document Type Definition (DTD) files, which allow you to do the following:

- Display a message box with prompts to the user.
- Display a form with entry fields to the user.
- Call a custom add-in component on the client.
- Call an executable program on the client.

The four pairs of predefined DTD files are located in the following directory of your WebLogic Integration server installation:

WLI_HOME\docs\apidocs\com\bea\wlpi\common\doc-files

In this path, *WLI_HOME* represents the directory in which you installed WebLogic Integration, typically c:\bea\weblogic700\integration.

Each pair has one DTD file for a request to the Worklist client, and another DTD file for a response from the Worklist client. The following table lists the predefined DTD files, and the actions performed by the Worklist application when it receives an XML document that conforms to one of the request DTD files.

Table 6-2 Worklist DTD Files

DTD Pair		Use To...
Request	ClientMsgBoxReq.dtd	Display a message dialog box in which the user responds to a message or query. Reply = ok/yes/no/cancel For more information, see "Displaying a Message Prompt to a User" on page 6-63.
Response	ClientMsgBoxResp.dtd	
Request	ClientSetVarsReq.dtd	Display a prompt dialog box with entry fields in which the user enters some values. Reply = field name/value pairs For more information, see "Displaying a Form to a User" on page 6-66
Response	ClientSetVarsResp.dtd	

Table 6-2 Worklist DTD Files

DTD Pair		Use To...
Request	`ClientCallPgmReq.dtd`	Execute a program on the client machine. Reply = program exit code For more information, see "Calling an Executable Program on the Client" on page 6-68.
Response	`ClientCallPgmResp.dtd`	
Request	`ClientCallAddInReq.dtd`	Invoke a custom extension to the Worklist client application. Reply = custom For more information, see "Calling a Custom Worklist Extension on the Client" on page 6-70.
Response	`ClientCallAddInResp.dtd`	

Note: If you will be accessing these DTDs often, you may want to import them into the repository for convenient retrieval. For procedures, see "Managing Entities in the Repository" on page 4-23.

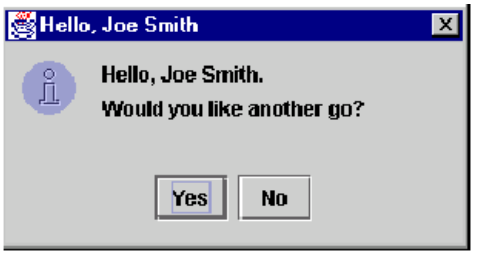
For each response received from the client, you will need to create a variable in which to place the value returned by the response. Most of these will be string-type variables. For procedures for creating variables, see "Working with Variables" on page 5-28.

The following sections provide detailed descriptions of the required document structure for each DTD pair.

Displaying a Message Prompt to a User

You can use the ClientMsgBox DTDs to display a message box to the Worklist user, as in the following example.

Figure 6-35 Example Message Prompt



The request DTD requires a document with the following structure.

Listing 6-1 ClientMsgBoxReq XML Document Structure

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

All elements and attributes are required, and are described below.

Table 6-3 ClientMsgBoxReq Elements and Attributes

Element or Attribute	Description	Valid Values
`message-box`	Text that appears in the message of the dialog box.	Any string of text.
`title`	Text that appears in the title bar of the dialog box.	Any string of text.
`style`	The Swing icon that appears in the top left corner of the dialog box:	The default is `plain`.
	No icon	`plain`
		`information`
		`question`
		`warning`
		`error`

Table 6-3 ClientMsgBoxReq Elements and Attributes

Element or Attribute	Description	Valid Values
`options`	The selection buttons at the bottom of the dialog box and the text they contain:	The default is `ok`.
	Three buttons: Yes, No, and Cancel.	`yes_no_cancel` Valid only when `style` element is set to `plain` or `question`.
	Two buttons: OK and Cancel	`ok_cancel` Valid only when `style` element is set to `error`, `plain`, or `warning`.
	One button: OK	`ok` Valid only when `style` element is set to `error`, `information`, `plain`, or `warning`.
	Two buttons: Yes and No	`yes_no` Valid only when `style` element is set to `plain` or `question`.

The response document provides the response of the user to the message box, according to the button selected. The response DTD requires the following structure.

Listing 6-2 ClientMsgBoxResp XML Document Structure

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

The element and attribute are required, and are described below, along with the expression required to return the value to a workflow variable.

Table 6-4 ClientMsgBoxResp Elements and Attributes

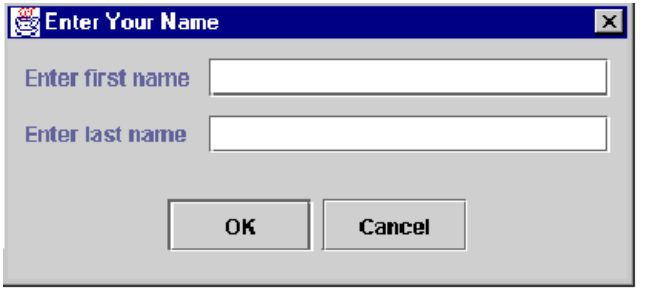
Element or Attribute	Description	Valid Values	Expression Required to Extract Value

`option`	The button selected by the client user, expressed as a string.	`ok`
		`yes`
		`no`
		`cancel`
			`XPath("/message-box/@option/text()")`

Displaying a Form to a User

You can use the ClientSetVars DTDs to display a form to the Worklist user, as in the following example.

Figure 6-36 Example Form



The request DTD requires a document with the following structure.

Listing 6-3 ClientSetVarsReq XML Document Structure

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

All elements and attributes are required. There must be at least one `<variable>` element and as many additional `<variable>` elements as you like. Element and attributes are described below.

Table 6-5 ClientSetVarsReq Elements and Attributes

Element or Attribute	Description	Valid Values
`set-variables`	Text that appears in the message of the dialog box.	Any string of text.
`title`	Text that appears in the title bar of the dialog box.	Any string of text
`name`	A name to identify the entry field.	Any string of text.
`prompt`	The text that appears as a prompt in front of the entry field.	Any string of text.

The response document provides the responses of the user to each of the entry fields. The response DTD requires the following structure.

Listing 6-4 ClientSetVarsResp XML Document Structure

```
<set-variables>
   <variable name="variable_name_1">response_1</variable>
   [<variable name="variable name_2">response_1</variable>]
   .
   .
   .
</set-variables>
```

All elements and attributes are required, and the number of `<variable>` elements should correspond to the number used in the request document. Elements and attributes are required, and the expression required to return the value to a workflow variable, are described below.

Table 6-6 ClientSetVarsResp Elements and Attributes

Element or Attribute	Description	Valid Values	Expression Required to Extract Value
`variable`	The response given by the client.	Any text string.	`XPath("/set-variables/variable[@name= "field_name"]/text()")`
`name`	The name used to identify the entry field.	Should correspond to the name used in the request document.	

Calling an Executable Program on the Client

You can use the ClientCallProgram DTDs to call a program on the Worklist client. The request document requires the following structure.

Listing 6-5 ClientCallProgramReq XML Document Structure

```
<call-program name="name" mode="{sync|async}">
    <actionid>provided by default</actionid>
    [<parm>parameter_1</parm>]
    .
    .
    .
    [<env-var name="name">environment variable
      definition_1</env-var>]
    .
    .
    .
</call-program>
```

All elements and attributes are required, except the `<parm>` and `<env-var>` elements, which are optional. You can specify zero or more `<parm>` or `<env-var>` elements.

Table 6-7 ClientCallProgramReq Elements and Attributes

Element or Attribute	Description	Valid Values
`name`	The name of the program.	Text string.
`mode`	The mode in which the executable program is to run, in relation to the Worklist:	The default is `async`.
	The program executes synchronously. The Worklist is blocked from continuing and its user interface is inaccessible until the called program has terminated.	`sync`
	The program executes asynchronously, running in parallel with the Worklist, whose user interface remains accessible while the called program is executing.	`async`
`parm`	Parameter to be passed to the program.	Any text string.
`env-var`	The definition of an environment variable you want to associate with the called program.	Text string.
`name`	The symbolic name of the environment variable.	Text string.

The response DTD requires the following structure.

Listing 6-6 ClientCallProgramResp XML Document Structure

```
<call-program exit-value="value" />
```

The element and attribute are required, and are described below.

Table 6-8 ClientCallProgramResp Element

Element or Attribute	Description	Valid Values	Expression Required to Extract Value
`exit-value`	The called program's numeric exit code, as retrieved by the operating system.	Consult the appropriate program documentation for more information on valid exit codes.	`XPath("/call-program/@exit-value")`

Calling a Custom Worklist Extension on the Client

You can use the ClientCallAddIn DTDs to call a custom extension to the Worklist client. The request document requires the following structure.

Listing 6-7 ClientCallAddInReq XML Document Structure

```
<call-addin name="name" mode="{sync|async}">
    <actionid>provided by default</actionid>
    [<parm>parameter_1</parm>]
    .
    .
    .
</call-addin>
```

All elements and attributes are required, except the `<parm>` element, which is optional. You can specify zero or more `<parm>` elements.

Table 6-9 ClientCallAddInReq Attributes and Elements

Element or Attribute	Description	Valid Values
`name`	The name of a custom Java class that implements the `com.bea.wlpi.client.worklist.WorklistAddIn` interface	The fully qualified JNDI name of the Java class.
`mode`	The mode in which the program is to run, in relation to the Worklist:	The default is `async`.
	The program executes synchronously. The Worklist is blocked from continuing, and its user interface is inaccessible, until the called program has terminated.	`sync`
	The program executes asynchronously, running in parallel with the Worklist, whose user interface is accessible while the program is executing.	`async`
`parm`	A parameter to be passed to the extension.	Any text string.

The response document is optional. Elements are optional and can consist of any number of custom elements and attributes you define.

Listing 6-8 ClientCallAddInResp XML Document Structure

```
<call-addin>
       [<tag_name_1 attribute_name_1="attribute_value"
         . . .>value</tag_name_1>]
       .
       .
       .
</call-addin>
```

Elements are described below.

Table 6-10 ClientCallAddInResp Elements and Attributes

Element or Attribute	Description	Valid Values	Expression Required to Extract Value
Any element name	Can consist of any number of attributes.	Custom-defined.	`XPath("/call-addin/path/text()")`

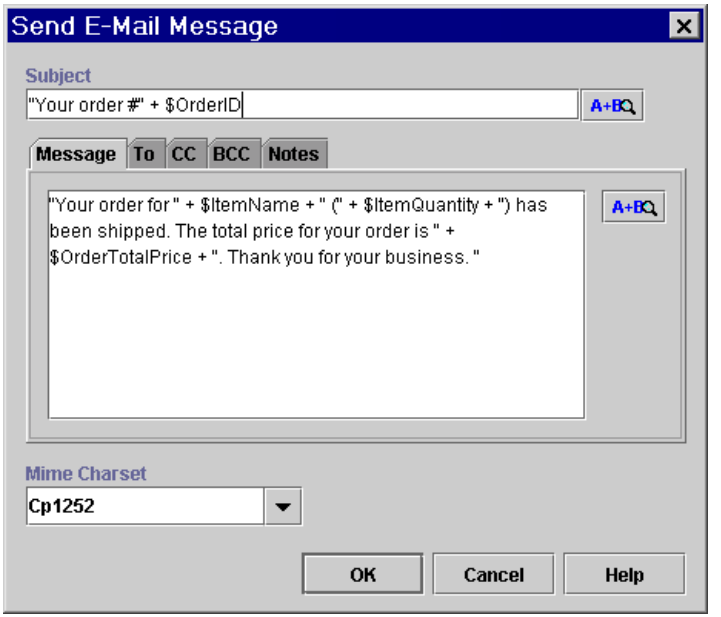
Sending E-Mail Messages

You can use the Send E-Mail Message action to send an e-mail message to a user of the WebLogic Integration system or even to an outside party. Internet standard SMTP protocol is used to transmit the message.

You can compose an e-mail message in any character set that your operating system supports at design time, and specify the character set to be used by the server at run time to the send the message. In an English locale, the default character set used by the server is cp1252, the default character set used by the Java Virtual Machine, but you can specify any character set supported by the Java language. For a list of these, see http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html.

Note: This action requires that your e-mail server be properly configured during or after the WebLogic Integration server installation process. For information on configuring mail server properties after installation, see "Customizing Mail Session Properties" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

Figure 6-37 Send E-Mail Message Dialog Box



To send an e-mail message:

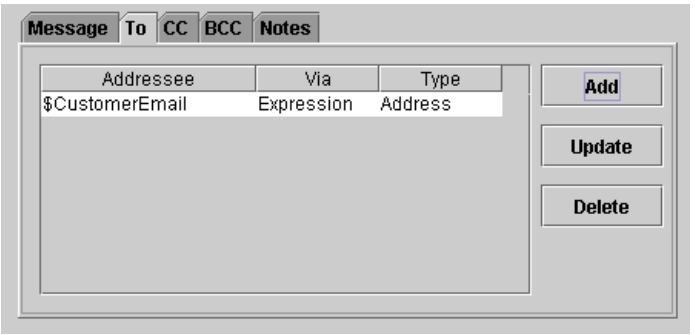
1. From the Miscellaneous Actions folder in the Add Action dialog box, select Send E-Mail Message and click OK to display the Send E-Mail Message dialog box.
2. In the Subject field, enter a valid workflow expression that will be evaluated at run time to produce the subject text of the e-mail. Your expression can consist of literals, variables, operators and functions. For more information on constructing expressions, see Chapter 8, "Using Workflow Expressions."
3. On the Message tab, in the text box, enter a valid workflow expression that will be evaluated at run time to produce the message text of the email.
4. Optionally, in the Mime Charset field, enter or select any MIME character set supported by the Java language in which you would like the server to send the message. For example, if you would like to send a message containing double-byte language characters, you can select UTF-8 (one of the Unicode standard formats). Note that the encoding you select must also be supported by the email client program used by your recipient in order for the message to be displayed correctly to him or her.

Note: The encoding you use to input characters is independent of the setting in the Mime Charset field, and is determined by your operating system and locale.

5. Select the To tab, which displays the following information about recipients:

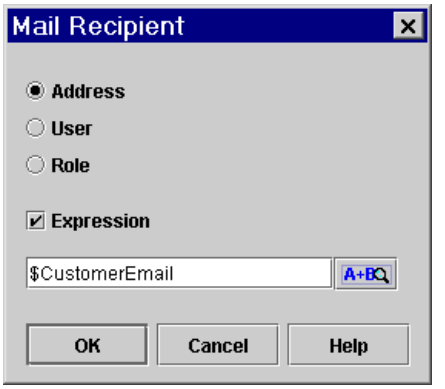
Addressee	The e-mail address, user or role name, or expression defined to produce the e-mail address at run time.
Via	Indicates whether the address is specified as a constant or an expression.
Type	Address, user, or role.

Figure 6-38 Send E-mail Message Dialog Box: To Tab



6. Click Add to add a recipient. The Mail Recipient dialog box is displayed.

Figure 6-39 Mail Recipient Dialog Box



7. Specify an addressee by selecting one of the following options:

   - Address — enter an e-mail address in the field, or enable the Expression check box and enter a workflow expression that will produce the e-mail address at run time.
   - User — select this option to display a drop-down list, from which you select the appropriate user from among all users associated with the current organization. The e-mail address is obtained from the user's properties, as described in "Creating a User" on page 3-15. Alternatively, enable the Expression check box and enter a workflow expression, such as a `CurrentUser()`, `WorkflowAttribute("Initiator")` or `TaskAttribute("Assignee")` function, that will produce a user's email address at run time, or a specific e-mail address surrounded by quotation marks.
   - Role — select this option to a display a drop-down list, from which you select the appropriate role from among all roles defined in the current organization. The message will be sent to all users belonging to that role. Alternatively, enable the Expression check box and enter a workflow expression that will produce the role name at run time, such as a `TaskAttribute()` function that can return a role name at run time.

   For more information on workflow functions and constructing expressions, see Chapter 8, "Using Workflow Expressions."

8. Click OK. The new recipient is added to the list.

9. Repeat steps 6 to 8 to add more recipients. To update a recipient, select it in the list and click Update. To delete a recipient, select it in the list and click Delete. Confirm the deletion when prompted.
10. Optionally, on the CC or BCC tabs, repeat steps 5 to 8 to specify recipients who you want to carbon copy or blind carbon copy.
11. Click OK to add the Send E-mail Message action.

Invoking Components

You can call software components such as EJBs, Java classes, and executable programs, and pass input and output parameters between the workflow and the components directly, by using the following actions:

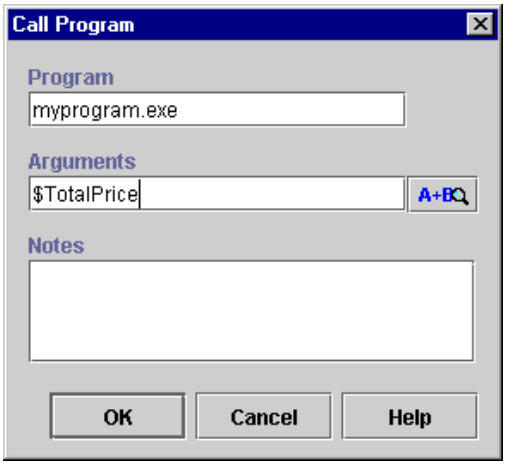
- Call Program—Calls an executable program on the server, which executes in parallel with the workflow. Described in "Calling an Executable Program on the Server" on page 6-76.
- Perform Business Operation—Calls a pre-configured business operation, representing a method on an EJB or Java class. Described in "Calling a Business Operation" on page 6-78.

Calling an Executable Program on the Server

You can use the Call Program action to call an executable program on the WebLogic Integration server. This action is always executed asynchronously, meaning that any actions following this action in the workflow are executed simultaneously, without waiting for the called program to complete.

Note: When using the Call Program action within a Send XML to Client action, you should be careful when granting access to a shell program, for example, `cmd.exec`. Doing so may compromise the security of the application by enabling full access to the client computer.

Figure 6-40 Call Program Dialog Box



To call an executable program on the server:

1. In the Add Action dialog box, expand the Integration Actions folder, select Call Program and click OK to display the Call Program dialog box.
2. In the Program field, enter the name of the executable file, including the extension.

   If you are running a DOS script file, such as a `.cmd` or `.bat` file that does not include any DOS shell-specific commands (e.g., `echo`), do one of the following:

   - If the WebLogic Integration server path, as set in your environment variables, includes the path for this file, enter the full name of the file including extension, for example: `testscript.bat`
   - If the WebLogic Integration server path, as set in your environment variables, does not include the path for this file, enter the fully qualified path and file name, for example: `c:\mydirectory\myfiles\testscript.bat`

   If you are running a DOS script file, such as a `.cmd` or `.bat` file that includes DOS shell-specific commands (e.g., `echo`), enter the following:

   ```
   c:\winnt\system32\cmd
   ```

3. In the Arguments field, enter a valid workflow expression, such as a variable name, that will be evaluated at run time to produce the argument to pass to the program.

If you are running a DOS script file, such as a `.cmd` or `.bat` file that includes DOS shell-specific commands (e.g., `echo`), enter the following:

```
"/c start c:\\path\\filename.extension expression"
```

4. Click OK to add the Call Program action.

Calling a Business Operation

You use the Perform Business Operation action to call a method on a Java component, such as an EJB or Java class, that performs a business activity.

The business operation you want to invoke must already be defined. Additionally, Java Object, Session EJB, and Entity EJB variables must already be defined to store references to the Java class or EJB instances whose methods are being called by the business operation. For details about defining business operations, see "Configuring Business Operations" on page 4-7. For details about defining variables, see "Working with Variables" on page 5-28.

Also note that before you can perform business operations that call methods on EJBs or non-static methods on Java classes, you must first call the business operation that serves to create an instance of the Java class or EJB on the WebLogic Integration server, according to the following rules:

- To call a business operation representing a static method on a Java class, you do not need to call a constructor method.
- To call a business operation representing a non-static method on a Java class, you must first call the business operation representing a constructor method for the class. This business operation only needs to be called a single time in the workflow.
- To call a business operation representing a method call on an Entity EJB, you must first call the business operation that creates the EJB instance. This business operation only needs to be called a single time in the workflow.
- To call a business operation representing a method call on a Session EJB—stateless or stateful—you must first call the business operation that creates the EJB instance. This business operation must be called once per transaction in which other methods are invoked on that EJB. For information on transactions and boundaries in a workflow, see "Understanding the BPM Transaction Model" in *Programming BPM Client Applications*.

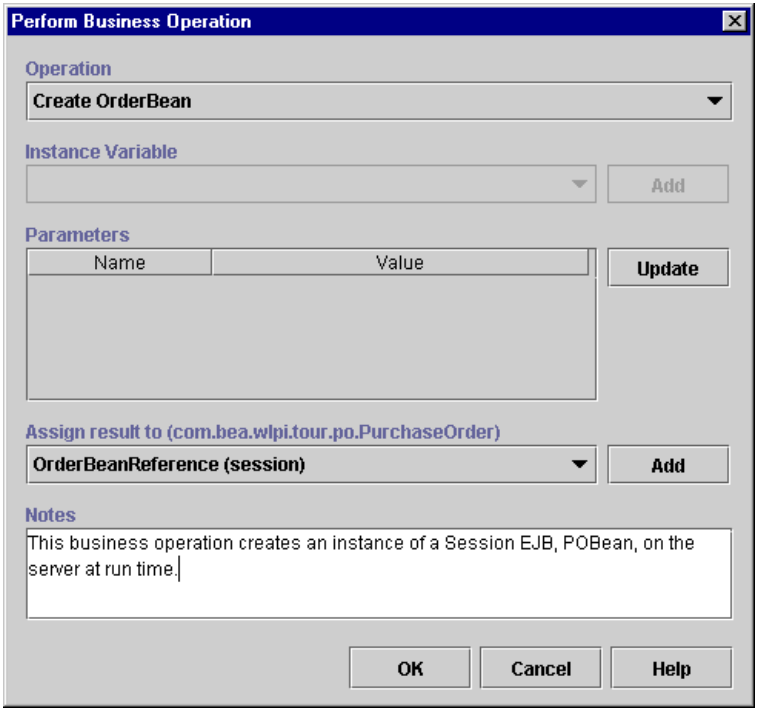
When you perform the business operation that creates the instance, you assign a reference to the instance to a variable, called an *instance variable*. You then identify the instance variable that references the EJB or Java class instance when calling other business operations representing methods contained in the same EJB or class. More information is provided in the following sections.

Calling the Business Operation to Create an EJB or Java Class Instance

When you call a business operation that creates an EJB or Java class instance, you must assign a reference to the instance to a variable of the same data type, as follows:

- for a Java class, the instance variable must be a Java object type
- for a Session EJB, the instance variable must be a Session EJB type
- for an Entity EJB, the instance variable must be an Entity EJB type

Figure 6-41 Perform Business Operation Dialog Box



To call a business operation to create an EJB or Java class instance:

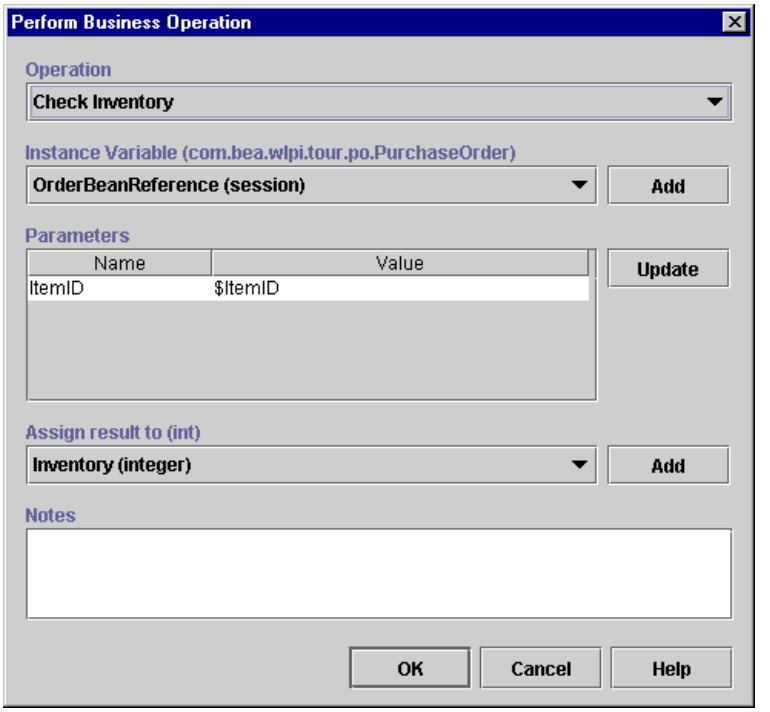
1. From the Integration Actions folder in the Add Action dialog box, select Perform Business Operation and click OK to display the Perform Business Operation dialog box.
2. From the Operation drop-down list, select the business operation that creates the Java class or EJB instance.
3. From the Assign Result drop-down list, select the instance variable for this business operation, or click Add to invoke the Create Variable dialog box and create the variable. The variable type must correspond to the type of component being created.
4. Click OK to add the Perform Business Operation action.

Calling Other Business Operations

Once the Perform Business Operation for the `create()` or constructor method of an EJB or Java class has been added to a node, you can add other business operations that invoke methods on the same class or EJB.

For methods that take parameters and return results, such as, for example, a calculated total price, you also need to create a variable that will store the value returned by the business operation. Be sure that this variable is of the same type as that specified by the method.

Figure 6-42 Perform Business Operation Dialog Box



To call other business operations:

1. From the Integration Actions folder in the Add Action dialog box, select Perform Business Operation and click OK to display the Perform Business Operation dialog box.

2. From the Operation drop-down list, select the business operation representing the business logic you want to execute. For business operations invoking methods on EJBs, or non-static Java class methods, the Instance Variable drop-down list is populated with Java Object, Session EJB or Entity EJB variables.

3. From the Instance Variable drop-down list, select the variable that you have designated to store the reference to the EJB or Java class, as specified in "Calling the Business Operation to Create an EJB or Java Class Instance" on page 6-79.

4. If the business operation displays parameters in the Parameters list, select a parameter from the list and click Update, and use the invoked Expression Builder dialog box to define this value. Typically this value will be provided by a workflow variable you have already defined.
5. If the business operation returns a result, from the Assign Result To drop-down list, select the variable to which the result will be assigned, or click Add to invoke the Create Variable dialog box and define the variable. The variable type must match that specified by the method.
6. Click OK to add the Perform Business Operation action.

Posting an XML Message to a JMS Topic or Queue

Use the Post XML Event action to send an XML message to a specified destination to trigger an event. This action can either create a new XML document, or use the content of an existing XML-type variable in the workflow; either way it embeds the XML content inside a JMS message that can be posted to an external JMS queue or topic for processing by an external application, or to an internal queue for processing by another workflow.

Note: If you are running WebLogic Integration in a clustered environment, posting an XML event to initiate another workflow is preferred over calling the workflow directly with a Start Workflow action, as this method provides better load balancing control.

You can compose the XML document to be sent, or import an existing XML document from the XML repository, a file on disk, or a URL. You can also specify the document to be sent as a variable in which the XML content can be specified at run-time.

In addition to XML message content, the Post XML Event action also inserts JMS headers and values into the message according to options that you specify in the action's properties. For information on standard JMS header fields in WebLogic Server, see "WebLogic JMS Fundamentals" in *Programming WebLogic JMS* at the following URL: http://edocs.bea.com/wls/docs70/jms/fund.html

JMS messaging options are described in the following sections.

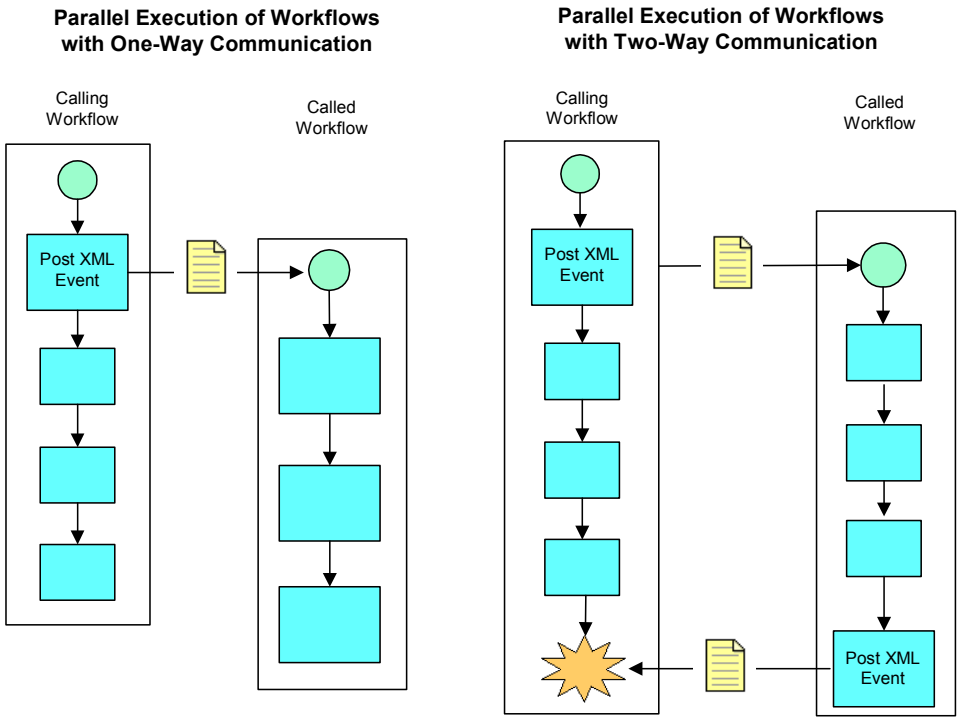
Posting an Event Asynchronously or Synchronously

You can set up your XML event to function in an asynchronous or synchronous manner in relation to workflows or other components that are configured to consume outgoing messages. You do this using Event nodes in the workflow initiating the communication to receive confirmation messages back from the recipient workflows.

By default, the Post XML Event action is asynchronous in that it simply posts a message in a "fire and forget" fashion, while the workflow proceeds to the next action. Thus, if you would like to trigger another workflow or application to be executed in parallel, and the calling workflow does not need to receive any communication back from the called workflow or application, simply use the action as is.

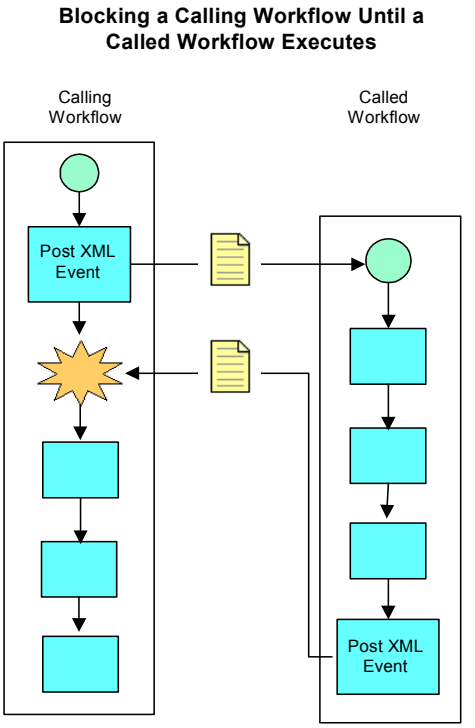
If you would like the calling workflow and the called workflow or application to execute in parallel, but the calling workflow *does* expect to receive a message back from the called workflow or application, you can set up an Event node in the calling workflow that receives the message in a "just in time" fashion. That is, you can set up an Event node only at the point in the workflow where the data returned by the called workflow or application is required. These two scenarios are illustrated in the following figure.

Figure 6-43 Posting an XML Event Asynchronously



If you want to post the XML event in a purely synchronous fashion, that is, to force the calling workflow to wait until the called workflow or application has finished executing before it proceeds, you must set up the called workflow or application to send a message when it finishes executing, and place an Event node that listens for this message immediately after the Post XML Event action in the calling workflow. This design is illustrated in the following figure.

Figure 6-44 Posting an Event Synchronously



Understanding JMS Messaging Options

The following sections discuss the various JMS messaging options that are available from within the Post XML Event dialog box.

Destination

You can specify an internal JMS queue to trigger an Event node in the current workflow or in another one, or to start another workflow defined with an event-triggered Start. (For details about event triggers in Start and Event nodes, see “Defining Event And Event-Triggered Start Properties” on page 5-38.) The JNDI name of the internal JMS queue, to which messages are sent by default, is

`com.bea.wlpiEventQueue`. If you have configured other queues in WebLogic Server, you can also specify an alternate queue name. (For more information on setting up alternate message queues for WebLogic Integration, see "Configuring a Custom Java Message Service Queue" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.)

You can also send the XML message to an external JMS topic or queue to communicate with an external application that subscribes to the JMS topic you specify, or with a specific application that is the JMS queue receiver.

Headers

JMS messages contain a standard set of header fields that are always transmitted with the message. In addition to the `JMSDeliveryMode`, `JMSDestination`, `JMSPriority`, and `JMSExpiration` (time to live) headers that are automatically inserted by the options available in the Post XML Event dialog box, you can also add property fields and values to your outgoing messages for application-specific information to specify information in the message that is not appropriate for the body of the message. For example, if you are using XML messaging to trigger another workflow, you may wish to use a property field to specify the name of the organization in which the workflow should start.

JMS message properties are name-value pairs. The name can be almost any string that is a valid identifier in the Java language. The value can be any one of the following types: Boolean, Byte, Short, Int, Long, Float, Double, or String.

For more information about JMS header and property fields, see "WebLogic JMS Fundamentals" in *Programming WebLogic JMS* at the following URL: http://edocs.bea.com/wls/docs70/jms/fund.html

If you use ordered or addressed messages, WebLogic Integration inserts property fields for an order key you specify, for ordered messages, and workflow instance IDs you specify, for addressed messages, based on the values you enter on the Addressing tab of the Post XML Event dialog box options. However, you can also manually insert two additional supported properties:

- WLPIInstanceIDs—a single variable, or a comma-separated list of variables containing workflow instance IDs determined at run time. For example, either of the following:
  - `$ParentID`
  - `$Child1ID + "," + $Child2ID + "," + $Child3ID`

- WLPITemplateNames—a single template name, or a comma-separated list of template names, or variables containing template names determined at run time

This feature can be useful when you need to consolidate a list of workflow instances or template names that have entered into a conversation with the current workflow and which you want to pass via a single message to an external application.

If you intend the message to be received by another workflow, the receiving workflow can use the `EventAttribute()` function in an event key expression or in a variable initialization in a Start or Event node to retrieve the information you specified in the property field. If multiple instance IDs or template names are specified, you must assign the result of the function to a variable defined as a Java object data type.

For more information on event key expressions, see "Configuring Event Keys" on page 4-18. For more information on initializing variables from event data, see "Initializing Variables from Event Data" on page 5-45. For details about the `EventAttribute()` function, see "Extracting Run-Time Event Data" on page 8-7.

Delivery Mode

You can specify whether a message is to be persistent or non-persistent. Persistent messages are written to a database table and are not lost even if the JMS server fails. The message is delivered again after the server recovers. Non-persistent messages can be lost if the JMS server fails. The message is not delivered again after the server recovers. The default delivery mode is persistent.

Time to live

Whether a message is persistent or non-persistent, you can specify an expiry time for your message. If the message is not delivered before the expiry time, it is discarded. This option is useful for messages that should not be delivered after a certain time, such as a stock bid. The time to live is expressed in milliseconds. For example, if you want the message to be available for 1 hour, you specify a value of 3600000 (1000 milliseconds x 60 seconds x 60 minutes). The default value of 0 (zero) indicates that the message will not expire.

If you specify an expiry time for an addressed message, the message is persisted until the message is successfully delivered to all specified recipients, or until it expires, whichever comes first.

Priority

You can assign a priority level from 0 to 9. Levels 0-4 are normal priority. Levels 5-9 are expedited priority. Messages with an expedited priority are delivered ahead of messages with a normal priority. You would typically use an expedited priority for an alarm or shutdown message. The default priority level is 4.

Note that priority overrides ordered messaging, so that you must specify the same priority level for all messages with the same order key. The recommended setting is to keep the default of 4.

To use other priority levels, you must first configure a destination key in WebLogic Server. For more information, see "Managing JMS" in the *WebLogic Server Administration Guide* at the following URL: http://edocs.bea.com/wls/docs70/adminguide/jms.html.

Transaction Mode

You can specify whether you want the message to be sent immediately, or when the current transaction containing the Post XML Event action commits. Sending the message immediately sends the message whether or not the transaction completes. Sending the message on commit ensures that the message is sent only if the transaction completes successfully and a commit is issued. If the transaction is unsuccessful and is rolled back, the message is not sent. The default is when the transaction commits. For more information on workflow transaction boundaries, see "Understanding the BPM Transaction Model" in *Programming BPM Client Applications*.

Addressed Messaging

You can use addressed messaging to guarantee that a response message is delivered to a particular workflow instance that has begun a conversation with the current workflow (either by calling it via the Start Workflow action, or by triggering a Start or Event node contained within it via a previously sent XML message) — even if the receiving Event node in the instantiated workflow has not yet been activated in the flow. (For information on node activation, see "Understanding the BPM Transaction Model" in *Programming BPM Client Applications*.)

When you use addressed messaging, you typically provide a list of workflow instance IDs to which the message should be delivered. These will have been sent from the originating workflows via a `WorkflowAttribute("InstanceID")` expression embedded in an XML message or as a parameter passed to the workflow via a Start

Workflow action, and then extracted and stored in a variable by a previous node in the current workflow. The list of instance IDs, then, is typically a comma-separated list of variables containing the appropriate IDs. The message will only be delivered to the instances specified in this list.

Note: Workflow instance IDs are stored as strings, so if you want to create variables to hold instance ID values, be sure that these variables are created as string types. For more information on the workflow attribute functions, see "Obtaining Run-time Workflow Data" on page 8-13.

Note that if you specify a time to live, the message will persist until the message has been delivered to all specified recipients, or until the message expires, whichever comes first.

Ordered Messaging

You can specify an order key that guarantees that messages are processed sequentially by the same event listener in the order in which they are received. As an example, if an order processing system receives requests to create an order and to update or cancel an order, you may want to guarantee that create request messages are processed first.

An order key must be an integer value, and the value must be the same for each event that you want processed in the order in which it is received. For example, if two events are posted at the same time, and you want them processed in the order in which they are received, you would enter the same order key value, such as the integer value of 8, for each event. Ordered messages must also be sent to the same JMS queue.

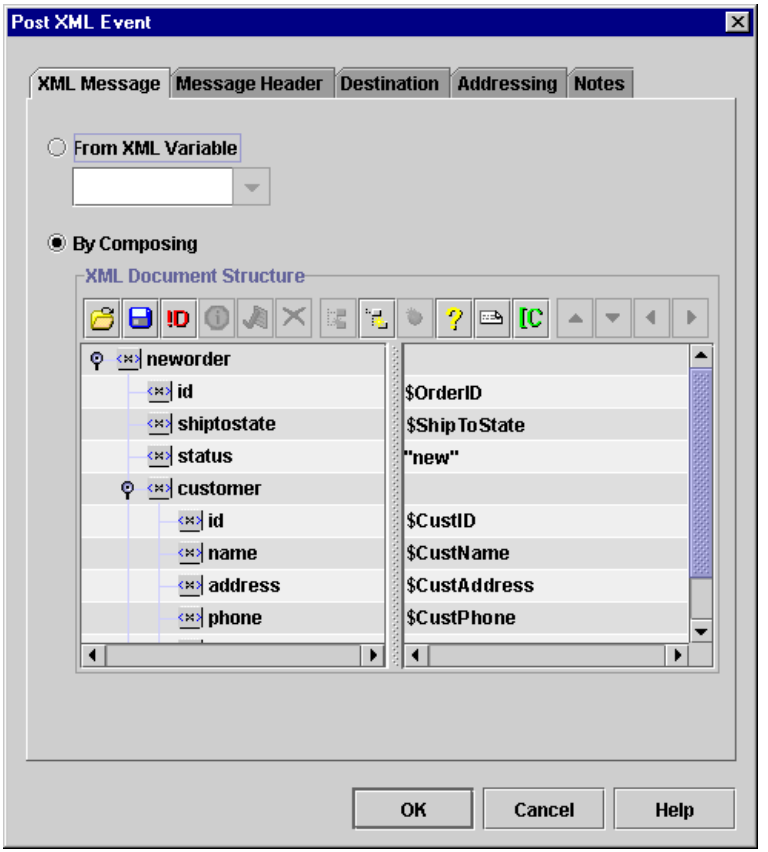
Note that ordered messaging is incompatible with message priority, so that if you use an order key, you must set the same priority level for all messages with the same order key. The recommended setting is to keep the default of 4.

Defining the Post XML Event Action

To define the XML event:

1. From the Integration Actions folder in the Add Action dialog box, select Post XML Event and click OK to display the Post XML Event dialog box.

Figure 6-45 Post XML Event Dialog Box: XML Message Tab



2. Select the XML Message tab to define the XML message you want this action to send, by doing one of the following:

   - If the XML document is stored in an XML type variable, select the variable from the From XML Variable drop-down list. For details about XML type variables, see "Working with Variables" on page 5-28.
   - If you want to compose or load an existing XML document, select the By Composing option. To create a new free-form document, click the Add Child button to begin adding nodes. To specify an existing XML document, click the Import button to load the document and edit as necessary. To create a new type-specified XML document, click the Set Content Type button to

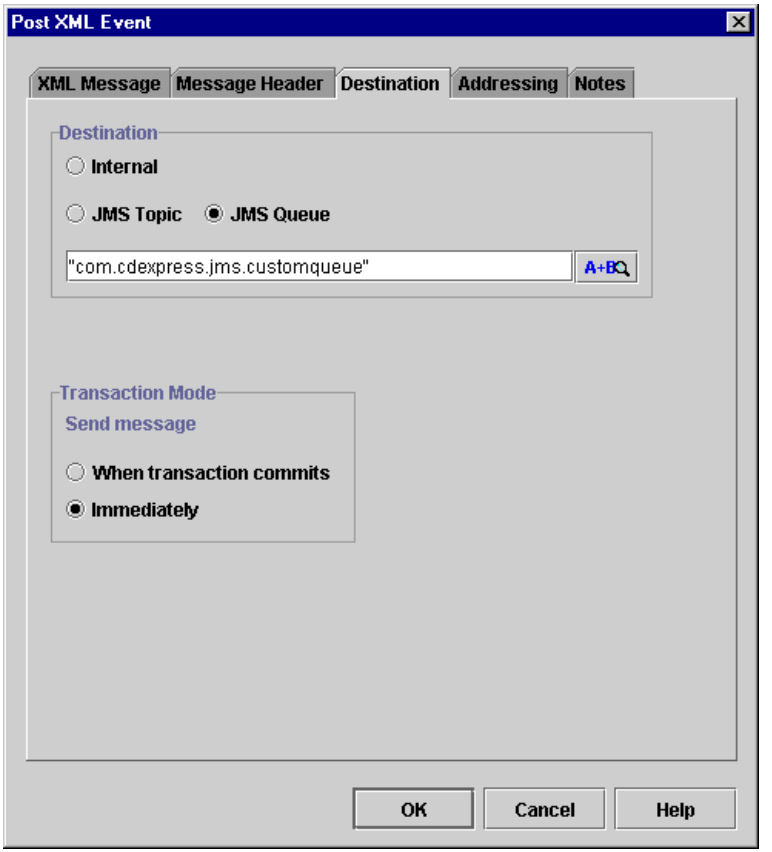
load a Schema document. For detailed procedures for all these options, see "Composing and Editing XML Documents" on page 7-2.

The XML document you define is stored in the workflow template definition.

3. Under the Destination tab, in the Destination options, select one of the following:
   - Internal—sends the message to the default internal queue, `wlpiEventQueue`.
   - JMS Topic—posts the message to an external topic. In the field, enter the JNDI name of the topic surrounded in quotation marks, or enter an expression that will determine the topic name at run time.
   - JMS Queue—posts the message to an external queue, or an alternate internal queue that you have configured in WebLogic Server. In the field, enter the JNDI name of the queue surrounded in quotation marks, or enter an expression that will determine the queue name at run time.

   Note: For more information on constructing expressions, see Chapter 8, "Using Workflow Expressions."

Figure 6-46 Post XML Event Dialog Box: Destination Tab



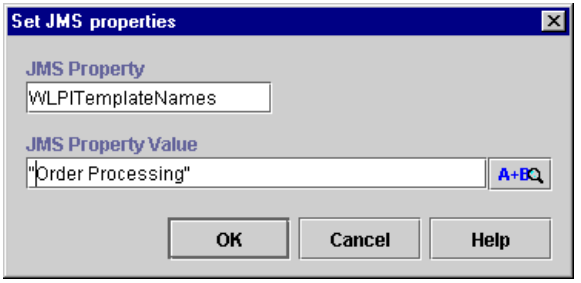
4. Optionally, from the Transaction Mode options, select one of the following:
   - When transaction commits — ensures that the message is sent only if the transaction completes successfully and a commit is issued. If the transaction is unsuccessful and is rolled back, the message is not sent.
   - Immediately — sends the message immediately without waiting for the transaction to complete.
5. Optionally, select the Message Header tab to specify any JMS message properties you want to add to the message. (For information on workflow-specific properties you can use, see “Headers” on page 6-86.) Click Add to display the Set JMS Properties dialog box. In the dialog box, specify the following:

a. In the JMS Property field, specify the property name.

b. In the JMS Property Value field, enter the property value. Enter the value directly, surrounded by quotation marks, or enter an expression that is evaluated at run time to produce the value.

c. Click OK.

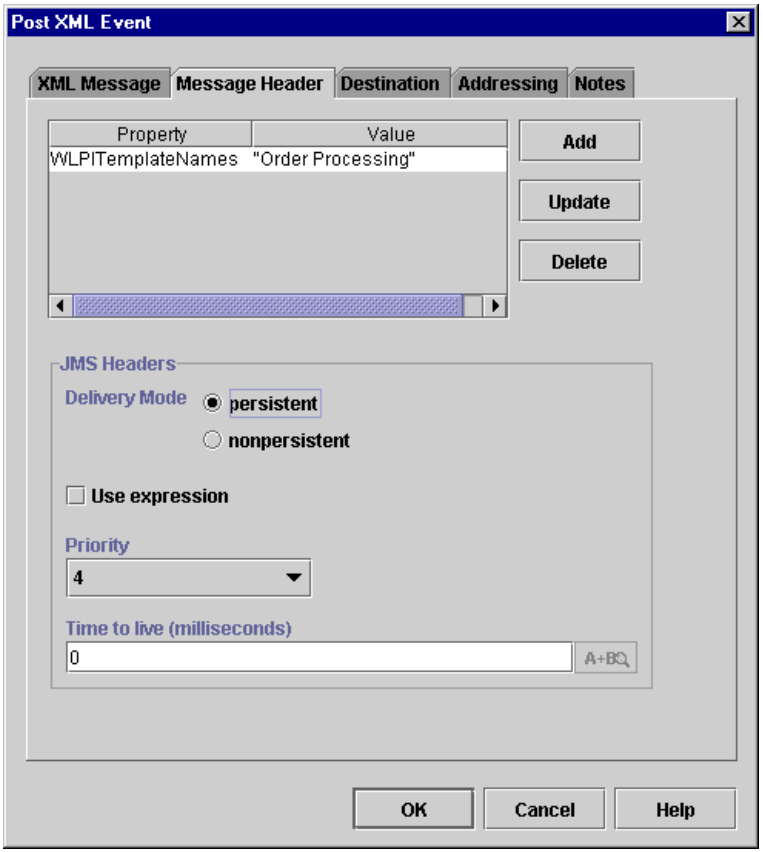
Figure 6-47 Set JMS Properties Dialog Box



6. Optionally, on the Message Header tab, specify a delivery mode, by selecting from the following options:

   - Persistent—writes the message to a database table and ensures that the message will persist even in the event of a JMS server failure.
   - Nonpersistent—does not persist the message. The message may be lost if the JMS server fails.

Figure 6-48 Post XML Event Dialog Box: Message Header Tab



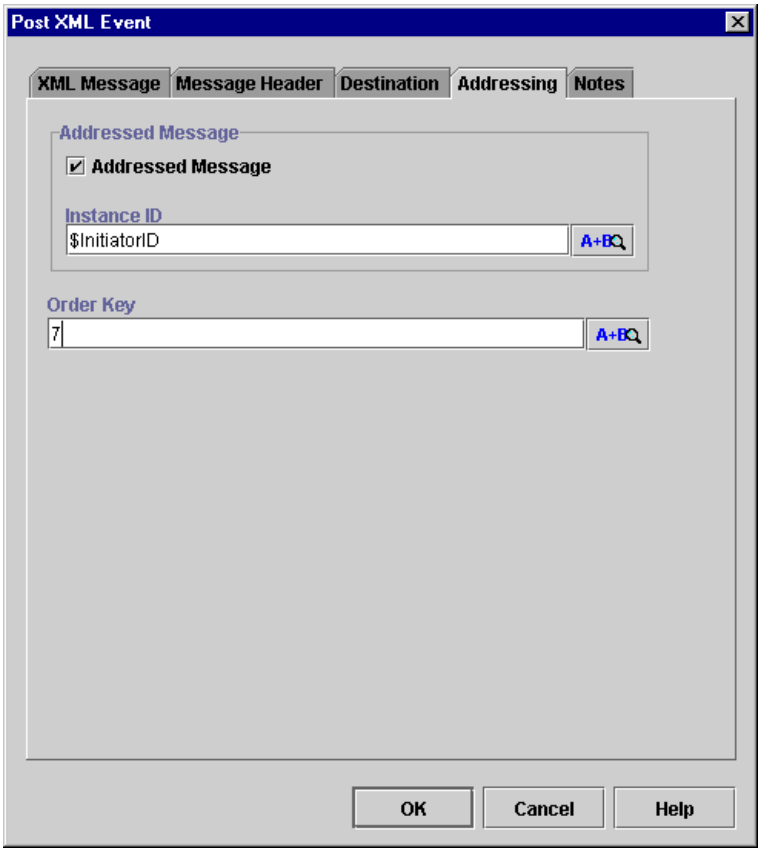
7. Optionally, on the Message Header tab, in the Time to live field, specify an expiry time, in milliseconds, or select the Use expression check box and enter an expression in the field that will be evaluated at run time to produce the value. A value of 0 indicates that the message never expires.

8. Optionally, specify a message priority by selecting a value from 0 (lowest priority) to 9 (highest priority) from the drop-down list, or select the Use expression check box and enter an expression in the field that will be evaluated at run time to produce the value.

   Note: If you specify an order key for ordered messaging, leave the default value of 4.

9. Optionally, to send an *addressed* message, select the Addressing tab to indicate that you want to persist an XML message for one or more workflow instances. Select the Addressed Message check box, and in the Instance ID field, enter a single variable name, or a comma-separated list of variables, in which you have stored workflow instance IDs previously in the current workflow. (For more information, see "Addressed Messaging" on page 6-88.)

Figure 6-49 Post XML Event Dialog Box: Addressing Tab



Note: If you specify a Time to live in the Message Header area, the message persists only for the time you specified.

10. Optionally, to send an *ordered* message, on the Addressing tab, enter an order key for the message that is the same as all other messages which you would like to have processed sequentially. This value must be an integer, or an expression that will determine the integer value at run time.

    Note: If you specify an order key, leave the default value message priority value of 4.

11. Click OK to add the Post XML Event action.

Transforming XML Documents

Extensible Stylesheet Language Transformations (XSLT) define rules for translating an XML document into another XML or non-XML document. An XSL template document specifies which elements in the input XML document are to be transformed, and how they are to be transformed.

The XSL Transform action provides a way for you to specify an input XML document that is to be transformed, an XSL template document that specifies the details of the transformation, and an output variable that will contain the transformed document. The actual transformation occurs at run time, and is performed by the XSL transformation engine bundled with WebLogic Server.

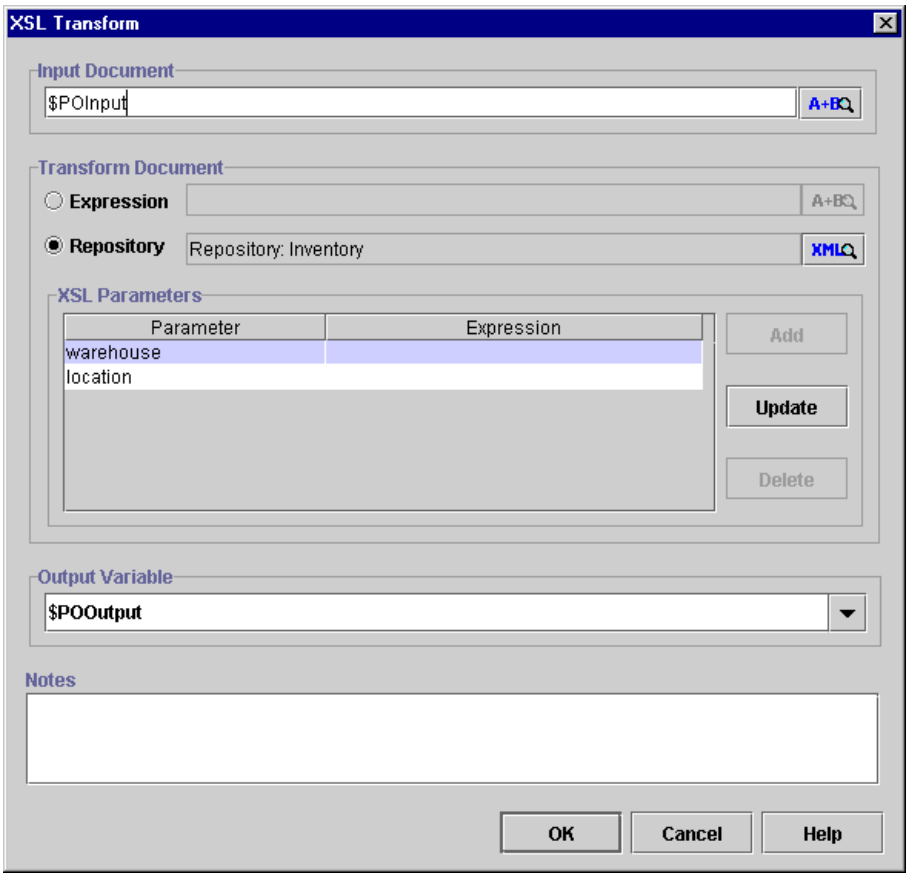
The input document can contain workflow expressions. If it does, the process engine resolves the expressions, and replaces the expression with the result before the transformation occurs. Similarly, the XSL template document can contain references to workflow variables. If it does, the process engine resolves the references, and replaces the reference with the appropriate value before the transformation occurs.

Note that the input document is specified as an expression that identifies the location of the document at run time. This expression could include the name of a workflow variable in which you have stored an XML document. In this case, you will need to have created an XML-type variable (for information, see "Working with Variables" on page 5-28), and assigned an existing or incoming XML document to it previously in the workflow—one way of doing this is by using the Set Workflow Variable action; see "Setting a Variable Value" on page 6-21 for procedures.

The XSL template document, or transform document, can be an entity stored in the repository (for more information, see “Managing Entities in the Repository” on page 4-23), or you can use an expression that locates the document at run time.If the XSL entity in the repository or the XSL document to be located at run time takes parameters, you can also specify expressions that will supply values for those parameters at run time.

The output document must be stored in an XML or string variable which you can create ahead of time. See “Creating a Variable” on page 5-30 for procedures.

Figure 6-50 XSL Transform Dialog Box

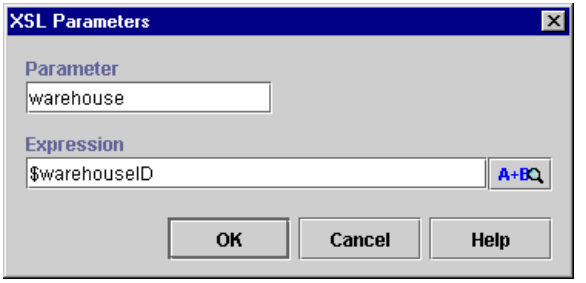


To transform an XML document:

1. From the Integration Actions folder in the Add Action dialog box, select XSL Transform and click OK to display the XSL Transform dialog box.
2. In the Input Document field, enter an expression that represents the XML document you want to transform. The expression is evaluated at run time to obtain the XML document. The expression could contain the name of a variable in which you have previously stored the input document.

   Note: For more information on constructing expressions, see Chapter 8, "Using Workflow Expressions."
3. In one of the Transform Document fields, specify the XSL template document that is used to transform the input document. Select one of the following options:
   - Expression — Enter an expression to be evaluated at run time to obtain the XSL template document.
   - Repository — Enter the name of an XSL template document stored in the XML repository, or click the XML button to display the XML Finder dialog box, to select the desired XSL file from the repository or a location specified by a URL. For procedures on using the XML Finder to retrieve XML entities, see "Retrieving XML Entities" on page 7-18.
4. If you selected to enter an expression for the XSL document, or specified an XSL document that takes parameters, optionally, add parameters to be passed to the transform document at run time by clicking Add to display the XSL Parameters dialog box.

Figure 6-51 XSL Parameters Dialog box



5. In the Parameter field, enter the name of the parameter.

6. In the Expression field, enter an expression that is evaluated at run time to produce the value of the parameter.
7. Click OK to add the parameter. The new parameter appears in the XSL Parameters list.
8. Repeat steps 4 to 7 to add more parameters, or click Update or Delete to update or delete existing parameters.
9. From the Output Variable drop-down list, select the variable that will contain the transformed XML document. The variable must be an XML type variable. You can also type the name of a variable. If it does not exist, you will be prompted to create it. For details about defining variables, see "Working with Variables" on page 5-28.
10. Click OK to add the action.

Handling Exceptions

All actions pertaining to exception handling are discussed in Chapter 9, "Handling Workflow Exceptions."

CHAPTER

7 Working with XML Entities

The following sections explain how to use the Studio to retrieve, compose and save XML entities:

- Overview of XML Document Management Tasks
- Composing and Editing XML Documents
- Using the XML Finder to Retrieve and Export XML Entities

Overview of XML Document Management Tasks

Several workflow actions contain a built-in XML editor that you can use to create, edit, and export free-form and type-specified XML documents that are embedded in workflows. The XML Finder helps you locate XML entities from various sources and save them to different types of storage. Tasks related to managing the content and storage of XML documents are described below.

- Optionally, import existing XML entities from the file system into the repository. These include XML document templates or instances, and XML Schema documents. Procedures are provided in "Managing Entities in the Repository" on page 4-23.

Alternatively, import previously exported XML entities from existing workflow packages. For details, see the procedures in "Importing Workflow Packages" on page 11-5.

- Add the following actions to existing nodes or actions: Set Workflow Variable, Post XML Event, Send XML to Client, XSL Transform, and Invoke Exception Handler. For information on the first three actions, see Chapter 6, "Defining Actions." For information on the Invoke Exception Handler action, see "Invoking an Exception Handler" on page 9-13.
- Compose, import, and edit XML documents from within action dialog boxes. Procedures are given in "Composing and Editing XML Documents" on page 7-2.
- Optionally, use the XML Finder to retrieve XML entities from the file system or the XML repository database for import into workflows. Procedures are given in "Retrieving XML Entities" on page 7-18.
- Optionally, use the XML Finder to export XML documents embedded in workflows to the file system or the XML repository database. Procedures are given in "Exporting XML Entities" on page 7-24.

Note: You may also want to familiarize yourself with the workflow expression language and the Studio's Expression Builder tool before beginning to define XML documents, since the dialog boxes in which you can create XML documents require that you enter XML element values as expressions. Complete information on workflow expressions is available in Chapter 8, "Using Workflow Expressions."

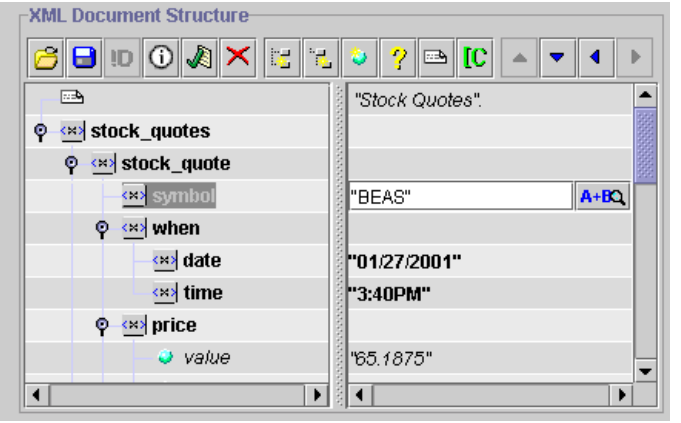
Composing and Editing XML Documents

The following action dialog boxes contain a built-in XML editor that you can use to compose, import, and edit well-formed XML documents which are saved within the workflow template definition:

- Set Workflow Variable
- Post XML Event

- Send XML to Client
- Invoke Exception Handler

Figure 7-1 XML Editor in Action Dialog Boxes



The XML editor presents XML documents in a tree structure consisting of nodes comprising the following standard XML markup:

- Elements
- Attributes
- Comments
- CDATA sections
- Processing instructions

The left pane displays metadata, such as element and attribute tags, and the right pane displays the actual data values for each tag.

The documents you create and edit in Studio dialog boxes are actually XML document *templates* that are used to generate XML document *instances* at run time. This means that you can use workflow expressions for elements and attributes to generate values at run time, and you can invoke the Expression Builder from one of these nodes to help you construct your expression. Element and attribute data must be formulated in workflow expression syntax, so all strings must be enclosed in quotation marks. If the value you enter is not a valid workflow expression, for example, a string not surrounded by quotation marks, or a backslash not preceded by an escape character,

the  icon appears in front of it. For information on the workflow expression language, and on using the Expression Builder, see Chapter 8, "Using Workflow Expressions."

You can create new documents and edit existing ones in two modes: free-form mode and content-type mode. In free-form mode, you simply compose and edit the document with no validation against a content type. Use free-form mode when you want to generate well-formed XML documents that do not need to conform to a specific content type. More information about creating free-form XML documents is provided in "Creating Free-Form Documents" on page 7-6.

In content-type mode, you specify the content type for a new or existing document by loading an existing external Schema document against which the document is validated. You can check the validity of the document as often as you like during the composing or editing process. Complete information about working with content-type-specified documents is provided in "Working with Type-Specified Documents" on page 7-11.

You can also import existing free-form and type-specified documents from the repository or a file on disk (see "Importing Existing Documents" on page 7-7), and add a content type to an existing document (see "Setting a New Content Type for Existing Documents" on page 7-14).

Finally, you can export document templates created or edited in the Studio to the repository or a file on disk. For more information on exporting XML documents from action dialog boxes, see "Exporting XML Entities" on page 7-24.

Dialog boxes that allow you to work with XML documents include a toolbar described in the following table. Specific procedures for creating, importing, editing, and setting the content type for XML documents are provided in the following sections.

Table 7-1 XML Editor Toolbar Buttons and Keyboard Shortcuts

Button	Keyboard Shortcut	Purpose
	Ctrl+q	Retrieves an existing XML document for editing from the repository, a file on disk, or a URL location. For details, see "Retrieving XML Entities" on page 7-18.

Table 7-1 XML Editor Toolbar Buttons and Keyboard Shortcuts

Button	Keyboard Shortcut	Purpose
	Ctrl+w	Saves the XML message from the workflow to the repository or a file on disk. For details, see "Exporting XML Entities" on page 7-24.
	Ctrl+t	Retrieves a Schema document to set as the content type for an XML document. For more information, see "Working with Type-Specified Documents" on page 7-11.
	Ctrl+k	Displays the contents of the Schema file currently set as the content type. For more information, see "Validating Type-Specified Documents" on page 7-16.
	Ctrl+l	Checks whether the document is valid XML and conforms to the current content type. For more information, see "Validating Type-Specified Documents" on page 7-16.
	Delete	Deletes the selected node, or removes a content type definition from the current document if the document type declaration is selected.
	Ctrl+Insert	Adds an element node at the same level as a selected element.
	Insert	Adds a root element node for the document, or an element below the level of a selected element.
	Ctrl+a	Adds an attribute node to a selected element.
	Ctrl+p	Adds a processing instruction node.
	Ctrl+m	Adds a comment node.
	Ctrl+n	Adds a CDATA section node.

Table 7-1 XML Editor Toolbar Buttons and Keyboard Shortcuts

Button	Keyboard Shortcut	Purpose
	Ctrl+Up arrow	Moves a selected node up within a level.
	Ctrl+Down arrow	Moves a selected node down within a level.
	Ctrl+Left arrow	Moves a selected node down a level.
	Ctrl+Right arrow	Moves a selected node up a level.

Creating Free-Form Documents

To create a free-form document:

1. In the action dialog box containing the XML editor, click the Add Element button  to add a root element to the document. The default element name is selected, ready to be edited.
2. To rename the element, type over the default text `element` to enter a name.
3. To add a value to the element, in the right pane, double-click the field next to the element, and type in a value, using workflow expression syntax.
4. Continue to add nodes and values to the document by using the toolbar buttons listed in Table 7-1 or by following the procedures listed in "Editing XML Documents" on page 7-9.
5. When you have finished composing your document, do any of the following:
   - Optionally, export the document to the repository or to a file. Follow the procedures in "Exporting XML Entities" on page 7-24.

- Save the document in the workflow by clicking OK in the action dialog box. If any elements or attributes do not contain valid workflow expressions as values, you are prompted to correct them before the document can be saved.
- To discard the document from the workflow, click Cancel in the action dialog box.

Importing Existing Documents

You can import both free-form and content-type-specified XML documents that you have created and exported from the Studio's XML editor or created by any other method. For information about importing type-specified documents, see "About Importing Type-Specified Documents" on page 7-12.

When you import an existing XML document into a workflow, the XML editor distinguishes between two kinds of XML documents:

- XML documents that were previously exported from the Studio's XML editor—that is, workflow XML document *templates* containing element and attribute values formulated in workflow expression syntax
- XML documents that were created by another method—that is, XML document *instances* containing standard XML values

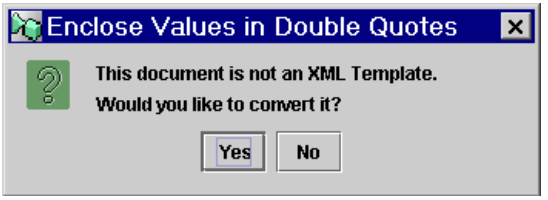
Workflow XML document templates created and exported from the Studio contain element and attribute string values enclosed in quotation marks, while other document instances usually do not. Thus, for XML documents not created in a Studio action dialog box, you will be prompted to convert the document to an XML template. The conversion process inserts quotation marks around all element and attribute values where necessary, so that you do not have to do so.

Note: The Studio does not recognize XML documents exported from previous versions of WebLogic Process Integrator as document templates, so you will still be prompted to convert the these XML documents. To prevent the XML editor from inserting an additional set of quotation marks, do not convert the imported document.

To import an existing free-form or type-specified XML document:

1. From an action dialog box, click the Import button  to display the XML Finder.
2. Follow the procedures in "Retrieving XML Entities" on page 7-18 to use the XML Finder to select a document from various sources. If the document is not detected as a document template, the Enclose Values in Double Quotes message prompt is displayed.

Figure 7-2 Enclose Values in Double Quotes Dialog Box



3. Respond as follows:
   - If the document was created and exported from an earlier version of WebLogic Process Integrator, click No.
   - If the document was created by any other method, click Yes.

   The document is imported into the editor.
4. Add and edit nodes and values as necessary, by using the toolbar buttons listed in Table 7-1 or by following the procedures listed in "Editing XML Documents" on page 7-9.
5. Optionally, for a type-specified document, validate the document by following the procedures in "Validating Type-Specified Documents" on page 7-16.
6. Optionally, to import another document, and replace the current one, click the Import button. When prompted to overwrite the current document, click Yes to continue the import, or No to cancel.
7. When you have finished editing your document, do any of the following:
   - Optionally export it to the repository or a file on disk. Follow the procedures in "Exporting XML Entities" on page 7-24.

- Save the document in the workflow, by clicking OK in the action dialog box. If any elements or attributes do not contain valid workflow expressions as values, you are prompted to correct them before the document can be saved.
- To discard the document from the workflow, click Cancel in the action dialog box.

Editing XML Documents

The following procedures are generalized for new or imported free-form or content-type-specified documents. For additional procedures on type-specified documents, see "Working with Type-Specified Documents" on page 7-11.

Table 7-2 Editing XML Documents

To . . .	Perform This Action
Navigate through a document	In the left pane, expand all nodes. Use the scroll bar on the right-hand side of the right pane, or press Tab or the up and down arrow keys, to move up and down within the document.
Add the first element of a free-form document	Click the Add Child button, or press Insert.
Add a sub-element to an element	In the left pane, select the element, then click the Add Child button or press Insert.
Add an element at the same level as another element	In the left pane, select the element, then click the Add Sibling button or press Ctrl+Insert.
Edit an element name	In the left pane, double-click the element to display an entry field. Change the element name as appropriate.
Add an attribute to an element	In the left pane, select the element, then click the Add Attribute button, or press Ctrl+a.
Edit an attribute name	In the left pane, double-click the attribute to display an entry field. Change the attribute name as appropriate.

Table 7-2 Editing XML Documents

To . . .	Perform This Action
Edit a value for an element or attribute	In the right pane, double-click the line next to the element or attribute for which you want to add or edit the value. An entry field is displayed. Type the value, or click the Expression button to invoke the Expression Builder to construct a valid workflow expression. String literals must be enclosed in double quotes. If the syntax is incorrect, a red X appears next to the entry. For details about defining workflow expressions, see Chapter 8, "Using Workflow Expressions."
Add a comment node	In the left pane, select the element after which the comment will appear, then click the Add Comment button, or press Ctrl+m.
Add a processing instruction	In the left pane, select the element after which the processing instruction will appear, then click the Add Processing Instruction button, or press Ctrl+p.
Change a processing instruction target	In the left pane, double-click the processing instruction to display an entry field. Change the processing instruction target as appropriate.
Add a CDATA section	In the left pane, select the element after which the CDATA section will appear, then click the Add CDATA button, or press Ctrl+n.
Edit a value for a comment, processing instruction, or CDATA section	In the right pane, double-click the line next to the comment, processing instruction, or CDATA icon for which you want to edit the value. An entry field is displayed. Type the value.
Change the order of nodes	Select the element, attribute, comment, processing instruction, or CDATA section you want to move, and click the Move Node Up or Move Node Down arrow, or press Ctrl+the up or down arrow key.
Change the hierarchical level of nodes	Select the element, comment, processing instruction, or CDATA section whose level you want to change, and click the Move Node Right or Move Node Left arrow, or press Ctrl+the right or left arrow key.
Delete a node	In the left or right pane, select the element or attribute you want to delete, and click the Delete button, or press the Delete key.

Working with Type-Specified Documents

You can create an XML document based on an existing external Schema. Internal DTD declarations are not supported.

You can also import an existing document based on a Schema, and you can set the content type for a document already loaded into the XML editor.

About Storing Referenced Schemas

Because theWebLogic Integration server needs to be able to access a Schema document associated with the XML document template at run time to create an XML document instance, Schemas to be referenced by Studio XML documents must exist as XSD files in a location that can be accessed by a URL, or as entities in the repository.

Warning: Keep in mind that if the URL points to a location not on the same machine as the WebLogic Integration server, there is always a risk that the system referenced by the URL may not be available at run time when the process engine generates the XML document instance. In such an occurrence, server exceptions will result. You should take care to store your XSD files in locations that can be reliably accessed at run time, such as on the file system of the WebLogic Integration server. Also ensure that the URLs you use remain valid at run time and point to the correct location of the required resources.

You may find that storing Schema resources in the repository is a convenient way to retrieve commonly accessed documents. For procedures on importing resources into the repository, see “Managing Entities in the Repository” on page 4-23.

On the other hand, you should also keep in mind that if you are planning to export newly created documents, or re-export imported documents, references to Schemas held in the repository are only identified by the entity name in the content type declaration, and will not be resolved by third-party XML parsers. Thus, you will want to ensure that your XSDs are placed in an appropriate location depending on whether an exported XML document will be used outside of WebLogic Integration or not. If they are to be used outside of WebLogic Integration, we recommend storing resources on a disk file and accessing them via URLs. If they are not going to be used outside WebLogic Integration, we recommend storing resources in the XML repository.

About Importing Type-Specified Documents

You can import an existing document that contains a document type declaration to an external Schema document, but not to an internal one. However, you must ensure that the referenced Schema is available for the WebLogic Integration server to access at design time, and that the document type declaration uses a valid URL to specify the document location. If the declaration only includes the name of the document, but not the full location, the WebLogic Integration server will not be able to resolve the reference, and the document cannot be imported.

Another option is to store referenced Schema resources in the XML repository. In this case, you may simply specify the name of the repository entity in the document type declaration, and the WebLogic Integration server will resolve the reference. For procedures on importing resources into the repository, see "Managing Entities in the Repository" on page 4-23.

If you are planning to re-export the document you have imported, the run-time considerations mentioned above also apply.

For procedures for importing type-specified documents, see "Importing Existing Documents" on page 7-7.

Creating Type-Specified Documents

When you create a new XML document based on a Schema, the XML editor creates a default document consisting of the following:

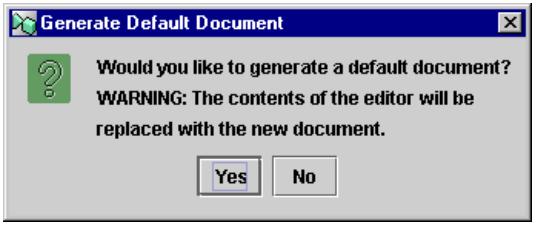
- For Schema-defined documents, an attribute within the root element, that specifies the referenced Schema.
- All required elements and attributes, and one instance of optional elements and attributes. You can add additional optional elements and attributes supported by the content type definition.
- The default value for elements or attributes that specify a default value from a predefined set of acceptable values. You can edit the value to specify another value supported by the content type definition.
- A processing instruction whether the document is a workflow XML template. The processing instruction is not visible, but is generated in an exported document.

To create a new type-specified document:

1. In the action dialog box containing the XML editor, click the Set Content Type button  . The XML Finder is displayed.

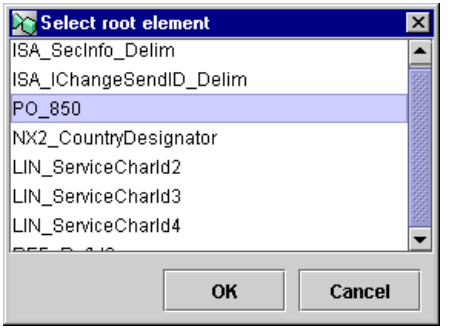
2. Follow the procedures in "Retrieving XML Entities" on page 7-18 to use the XML Finder to select a Schema document from the appropriate sources. When the content-type document is retrieved, the Generate Default Document message prompt is displayed.

Figure 7-3 Generate Default Document Dialog Box



3. Click Yes to generate a default document. The Select Root Element dialog box is displayed, showing a list of all defined elements.

Figure 7-4 Select Root Element Dialog Box



4. Select the element that should be the root element of the document and click OK. A default document is created. For a DTD-based document, a prolog with the document type declaration is inserted above the root element. For a Schema-based document, an attribute referencing the Schema is inserted in the root element.

5. Add and edit nodes and values as necessary by using the toolbar buttons listed in Table 7-1 or by following the procedures listed in "Editing XML Documents" on page 7-9.
6. Validate your document by following the procedures in "Validating Type-Specified Documents" on page 7-16.
7. When you have finished composing and validating your document, do any of the following:
   - Optionally export it to the repository or a file on disk. Follow the procedures in "Exporting XML Entities" on page 7-24.
   - Save the document in the workflow, by clicking OK in the action dialog box. If any elements or attributes do not contain valid workflow expressions as values, you are prompted to correct them before the document can be saved.
   - To discard the document from the workflow, click Cancel in the action dialog box.

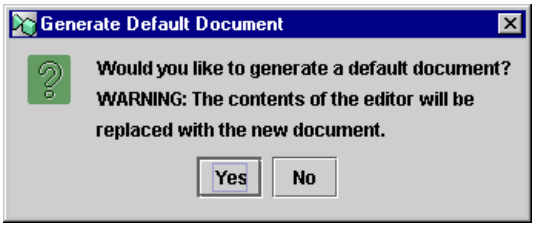
Setting a New Content Type for Existing Documents

You can apply a content type definition to a document after you have created it or to an existing document after import. You can also remove a content type from a type-specified document, or change the content type for a new or existing type-specified document.

To apply a content type definition to a free-form document:

1. Do one of the following:
   - Create a new free-form document, as described in "Creating Free-Form Documents" on page 7-6.
   - Import an existing free-form document by following steps 1 to 3 in "Importing Existing Documents" on page 7-7.
2. Click the Set Content Type button  . The XML Finder is displayed.
3. Follow the procedures in "Retrieving XML Entities" on page 7-18 to use the XML Finder to select a Schema document from the appropriate sources. When the content-type document is retrieved, the Generate Default Document message prompt is displayed.

Figure 7-5 Generate Default Document Dialog Box



4. Click No to preserve the existing document.
5. Continue to define your document, as described in "Editing XML Documents" on page 7-9.
6. Validate your document, as described in "Validating Type-Specified Documents" on page 7-16.

To remove a content type definition from a type-specified document:

1. For a Schema-specified document, select the root element attribute node referencing the Schema.
2. Click the Delete button  . The content type definition is removed, and the document becomes a free-form document.

To change the content type for a type-specified document:

1. For a Schema-specified document, select the root element attribute node referencing the Schema.
2. Select the node containing the content type definition, above the root element.
3. Click the Delete button  . The content type definition is removed.
4. Click the Set Content Type button  . The XML Finder is displayed.
5. Follow the procedures in "Retrieving XML Entities" on page 7-18 to use the XML Finder to select a Schema document from the appropriate sources. When the content-type document is retrieved, the Generate Default Document message prompt is displayed.

6. Click No to preserve the existing document.
7. Continue to define your document, as described in "Editing XML Documents" on page 7-9.
8. Validate your document, as described in "Validating Type-Specified Documents" on page 7-16.

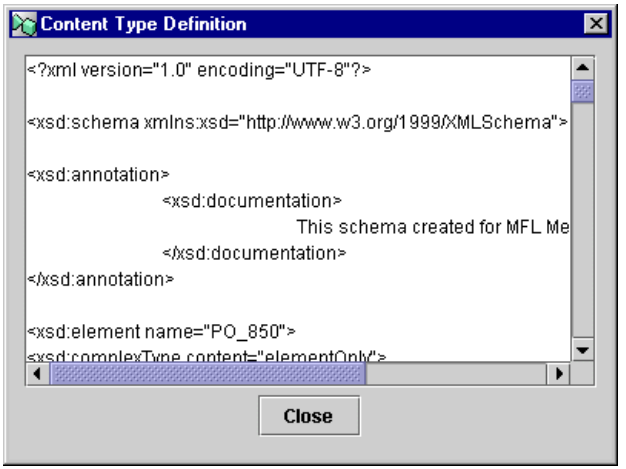
Validating Type-Specified Documents

While composing or editing a new or imported type-specified documents, you can view the content of an Schema document set as the current content type, and you can take advantage of the Studio validation feature that allows you to view the source of errors and edit them simultaneously.

To view the content of the associated Schema:

1. Click the View Content Type Definition button  . The Content Type Definition window is displayed, showing the content of the current content type definition document.

Figure 7-6 Content Type Definition Window

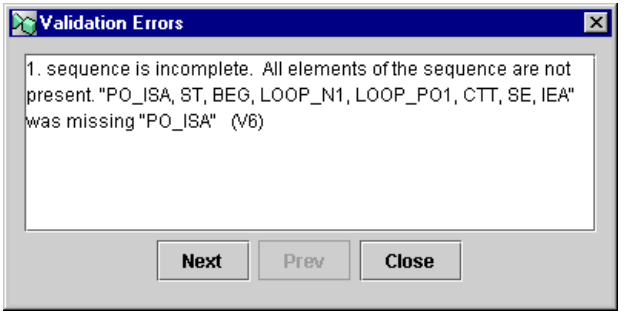


2. Click Close to close the window.

To validate a new or imported type-specified document against the content type definition:

1. Click the Validate Document Structure button  . A message is displayed, informing you whether the document is valid. If there are errors in your document, the Validation Errors dialog box appears, listing the nature and location of the error.

Figure 7-7 Validation Errors Dialog Box



2. Click Next or Prev to scroll through the list of errors, and edit the erroneous node.
3. When done, click Close to close the dialog box.
4. Click the Validate Document Structure button again to re-validate the document, and repeat steps 1 to 3 until you receive the Valid Document message.
5. Click OK to close the Valid Document message box.

Using the XML Finder to Retrieve and Export XML Entities

The XML Finder dialog box allows you to retrieve from and save XML entities to the repository, the local file system, or a URL. The dialog box also maintains a list of the XML entities you used most recently, so you do not have to search for them if you want to use them again.

XML entities can be XML documents, Document Type Definitions (DTD), Schema documents (XSD), Message Language Format (MFL) files, and Extensible Stylesheet Language (XSL) template documents.

You can access the XML Finder several ways. Different operations are available depending on the way you access the XML Finder, and whether you are retrieving or saving a document. These topics are described in the following sections.

Retrieving XML Entities

You can retrieve and load XML entities into the following dialog boxes:

- From the Set Workflow Variable, Send XML to Client, Post XML Event and Invoke Exception Handler action dialog boxes, you can retrieve existing XML documents from the repository, the local file system, or a URL, for editing within the dialog box. You can retrieve existing Schema documents from the repository or a URL, to set the content type for a document you compose or edit.
- From the XSL Transform dialog box, you can retrieve the name of an XSL document to use as a transform document from the repository or a URL.
- From the XPath Wizard you can open any XML entity from the repository, the local file system, or a URL, to generate and test XPath expressions.
- By choosing Tools→Show XML Finder, you can import XML entities from the local file system or a URL into the repository. For details, see "Managing Entities in the Repository" on page 4-23.

Retrieving the Most Recently Used XML Entities

The XML Finder maintains a list of the XML entities you used most recently on your workstation. You can re-use the entities in the list without having to search for them again.

To retrieve an entity from the most recently used list:

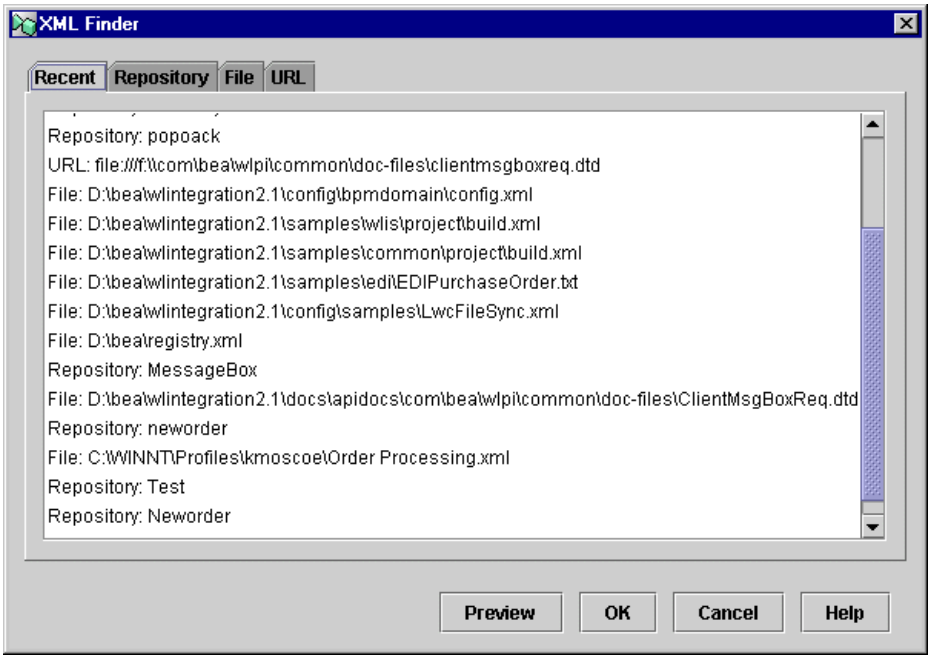
1. Open the XML Finder by doing one of the following:

- From the Set Workflow Variable, Send XML to Client, Post XML Event and Invoke Exception Handler action dialog boxes, click the Import button to retrieve an XML document.
- From the XPath Wizard, click the Open button.

Note: The Recent tab is unavailable from the XSL Transform dialog box, and from dialog boxes where you can open a Schema document, because these actions create an XML document instance at run time. Specifying a recent entity is not appropriate in these cases because the process engine cannot access the file at run time.

2. In the XML Finder dialog box, select the Recent tab. Disk files are identified by filename and location. repository entities are identified by entity name.

Figure 7-8 XML Finder: Recent Tab



3. Select the entity you want to use.

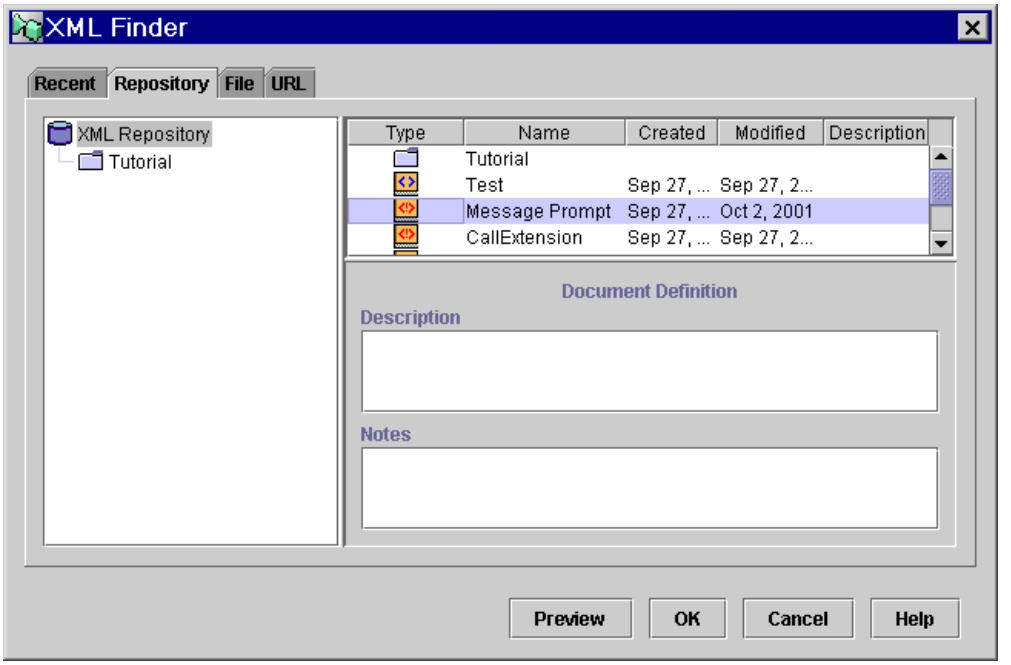
4. Optionally, to see the contents of the entity, click Preview to display the Preview Document window. Click OK to close the window.
5. To return the entity to the original dialog box, click OK.

Retrieving from the Repository

To retrieve an entity from the repository:

1. Open the XML Finder by doing one of the following:
   - From the Set Workflow Variable, Send XML to Client, Post XML Event and Invoke Exception Handler action dialog boxes, click the Import  button to retrieve an XML document, or click the Set Content Type  button to retrieve or Schema document.
   - From the XSL Transform dialog box, click the XML button.
   - From the XPath Wizard, click the Open  button.
2. In the XML Finder dialog box, select the Repository tab.
3. Select the folder from which you want to retrieve an XML entity.

Figure 7-9 XML Finder: Repository Tab



4. In the top-most panel on the right, select the entity you want to retrieve.
5. Optionally, to see the contents of the entity, click Preview to display the Preview Document window. Click OK to close the window.
6. To return the entity to the original dialog box, click OK.

Retrieving from the File System

You can retrieve content from a file on a local disk drive or any network drive mapped to the local machine.

To retrieve an entity from the local file system:

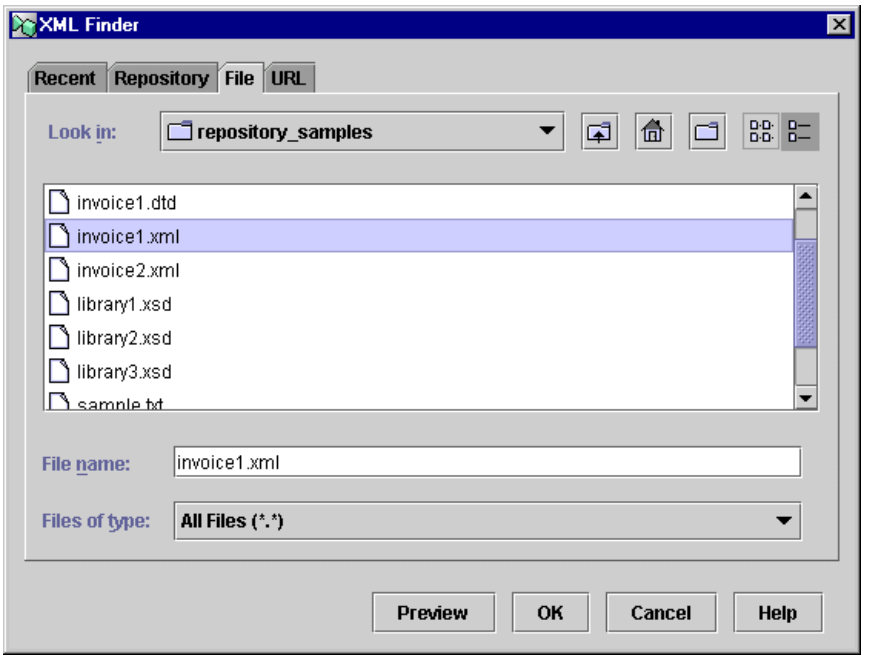
1. Open the XML Finder by doing one of the following:

- From the Set Workflow Variable, Send XML to Client, Post XML Event and Invoke Exception Handler action dialog boxes, click the Import button to retrieve an XML document.
- From the XPath Wizard, click the Open button.

Note: The File tab is unavailable from the XSL Transform dialog box, and from dialog boxes where you can open a Schema document, because these actions create an XML document instance at run time. Specifying a file on the local file system is not appropriate in these cases because the process engine cannot access the local file system at run time.

2. In the XML Finder dialog box, select the File tab.

Figure 7-10 XML Finder: File Tab



3. In the Look in field, specify the drive and folder from which you want to obtain the XML entity. If necessary, use the buttons to the right of the Look in field.
4. From the file list, select the entity you want to retrieve.

5. Optionally, to see the contents of the entity, click Preview to display the Preview Document window. Click OK to close the window.
6. To return the entity to the original dialog box, click OK.

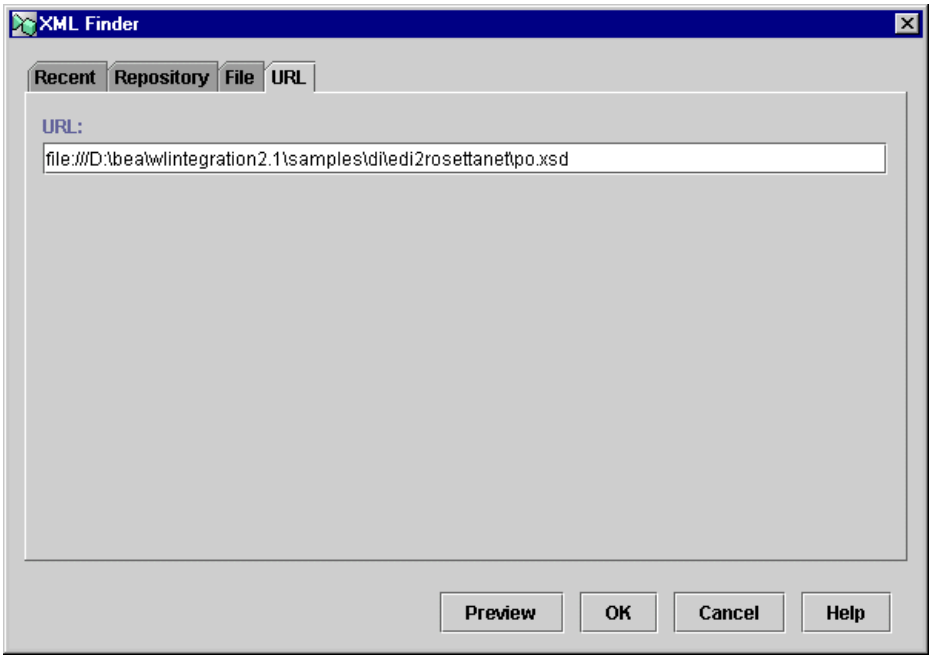
Retrieving from a URL

You can use a URL to specify a location on a local machine, remote machine on a network, or other external systems. Remote locations must be mapped to a drive on the local machine.

To retrieve an entity from a URL:

1. Open the XML Finder by doing one of the following:
   - From the Set Workflow Variable, Send XML to Client, Post XML Event and Invoke Exception Handler action dialog boxes, click the Import  button to retrieve an XML document, or click the Set Content Type  button to retrieve a Schema document.
   - From the XSL Transform dialog box, click the XML button.
   - From the XPath Wizard, click the Open  button.
2. In the XML Finder dialog box, select the URL tab.

Figure 7-11 XML Finder: URL Tab



3. In the URL field, enter the complete URL, including the protocol, server, path and filename, of the XML entity you want to retrieve.
4. Optionally, to see the contents of the entity, click Preview to display the Preview Document window. Click OK to close the window.
5. To return the entity to the original dialog box, click OK.

Exporting XML Entities

You can save XML documents created within the Studio from the following dialog boxes:

- From the Set Workflow Variable, Send XML to Client, Post XML Event and Invoke Exception Handler action dialog boxes, you can save XML documents

you compose within the action to the repository or the local file system for reuse in other workflow actions, nodes, or even template definitions.

- By choosing Tools—Show XML Finder, you can export XML entities stored in the XML repository to the file system. For details, see "Managing Entities in the Repository" on page 4-23.

Documents exported from action dialog boxes are formatted with the standard XML escape sequences (`&quot;`) representing quotation marks around element and attribute values. For content-type-specified documents, a prolog including the DOCTYPE and an epilog indicating the validity are also inserted.

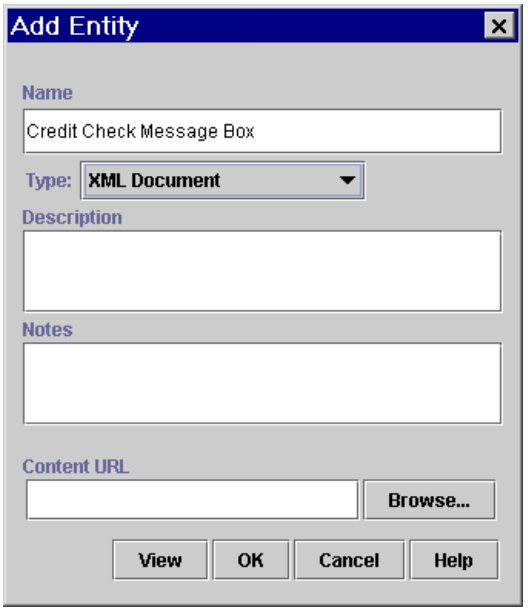
Exporting to the Repository

When you export an XML document template from an action dialog box to the repository, the entry is created in two steps. First, an empty entity is created. Then, the entity is populated with the content of the document template. For this reason, it is not possible to preview the document until after it has actually been populated with content, which occurs when you exit the XML Finder.

To save an entity from an Action dialog box to the repository:

1. From the Set Workflow Variable, Send XML to Client, Post XML Event and Invoke Exception Handler action dialog boxes, click the Export button  . The XML Finder appears.
2. Select the Repository tab.
3. In the left pane, right-click the folder in which you want to save the entity, and from the pop-up menu, select Add Entity. The Add Entity dialog box appears.
4. In the Name field, enter a unique name for the entity you are adding.

Figure 7-12 Add Entity Dialog Box



5. Using the Type drop-down list, select the type of entity you are adding.
6. Optionally, enter a description and notes about the entity in the Description and Notes fields, respectively.
7. Click OK. The entity appears in the top right window of the XML Finder.
8. Click OK to exit the XML Finder. The entity has now been created in the repository.

Exporting to the File System

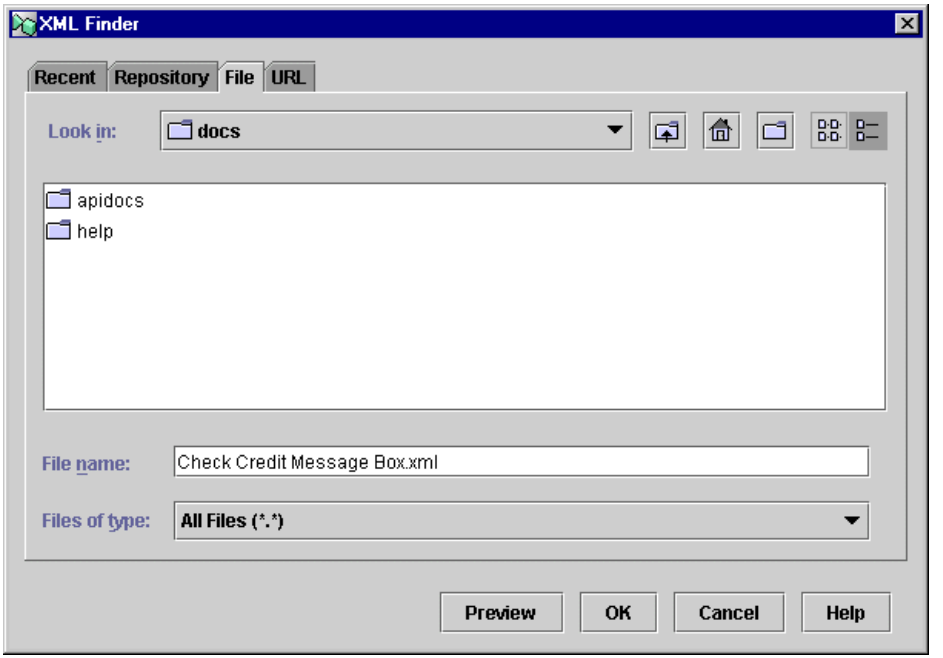
From action dialog boxes, you can save the XML documents you create or edit as `.xml` files on a local disk drive or any remote drive mapped on the local machine.

To save an entity to a file on disk:

1. From the Set Workflow Variable, Send XML to Client, Post XML Event and Invoke Exception Handler action dialog boxes, click the Export button  . The XML Finder appears.

2. In the XML Finder, select the File tab.

Figure 7-13 XML Finder: File Tab



3. In the Look in field, specify the drive and folder in which you want to save the XML entity. If necessary, use the buttons to the right of the Look in field.
4. In the File name field, enter the name of the file you want to create, and add the `.xml` extension. If the file already exists, you are prompted with a warning message. Click Yes to overwrite the file, No to enter a new filename, or Cancel to cancel the export.
5. Click OK to save the file and exit the XML Finder.

Exporting to a Recently Accessed File

You can only use the Recent tab to export to and overwrite an already existing file.

To save an entity to a recently accessed file:

1. From the Set Workflow Variable, Send XML to Client, Post XML Event and Invoke Exception Handler action dialog boxes, click the Export button  . The XML Finder appears.
2. In the XML Finder, select the Recent tab.
3. In the list of entities, select the XML file or repository entity you want to overwrite.
4. Click OK to save the file and exit the XML Finder.

Exporting to a File Located by a URL

You can only use the URL tab to export to and overwrite an already existing file that can be located by a URL.

To save an entity to an existing file located by a URL:

1. From the Set Workflow Variable, Send XML to Client, Post XML Event and Invoke Exception Handler action dialog boxes, click the Export button  . The XML Finder appears.
2. In the XML Finder, select the URL tab.
3. In the URL field, enter the complete URL, including the protocol, server, path and filename, of the XML file you want to overwrite.
4. Click OK to save the file and exit the XML Finder.

CHAPTER

8 Using Workflow Expressions

The following sections explain the workflow expression language, and how to use the Expression Builder and XPath Wizard to generate workflow expressions:

- About Workflow Expressions
- Using Literals
- Using Variables
- Using Operators
- Using Functions
- Data Type Conversions for Variable Assignment
- Using the Expression Builder
- Creating XPath Expressions Using the XPath Wizard

About Workflow Expressions

Wherever you see the Expression button  next to a field in a Studio dialog box, this indicates that the field requires an entry formulated in the workflow expression language. Expressions are used throughout the Studio to create template definition labels, define conditions in decision nodes and events, configure event keys, and to specify information that will be provided at run time.

A workflow expression is an algebraic expression that defines a calculation that the system performs at run time, and is made up of literals, such as strings, integers and other constants, workflow variables, operators, and workflow functions. Workflow expression syntax allows you to manipulate strings, test for relationships and conditions, perform arithmetic calculations, use functions that obtain run-time information from workflows or XML messages, and so on.

The result of an expression may be a string, integer, double, date/time value, or Boolean values. Expressions that yield a Boolean result are referred to as conditional expressions or conditions.

The Studio also provides two tools to help you construct expressions: the Expression Builder, which provides syntax checking and error information (described in "Using the Expression Builder" on page 8-28), and the XPath Wizard, which you use to generate XPath expressions from sample XML documents (described in "Creating XPath Expressions Using the XPath Wizard" on page 8-31).

The following sections provide the syntax for literals, operators, and workflow variables, and introduce the use and syntax of built-in functions.

Using Literals

Expressions can contain literal values or constants. The following table describes the available literals.

Table 8-1 Literal Usage

Literal Type	Format	Description	Example
String	"*string*" or '*string*'	A character string enclosed in single or double quotes. Use the following escape sequences to embed special characters in the string: \r carriage return character \n line feed character \' single quote character \" double quote character \f form-feed character \t tab character \\ back slash character \0 ASCII null character	"cancelled"
Integer	[+|-] *digits*	A 32-bit signed integer in the range -2,147,483,647 to +2,147,483,648 (approximately 9 digits or precision)	5000
Double	[+|-]*digits*[.[*digits*]]	A 64-bit IEEE double in the range $-2^{53} \times 10^{104}$ to $+ 2^{53} \times 10^{104}$ (approximately 15 digits of precision)	5000.00
Date	"*MM/dd/yyyy hh:mm:ss* AM|PM GMT[+|-]*hh:mm:ss*"		"10/01/2001 12:11:11 AM GMT-04:00"

Note: Date literals can only be used to set date-type variables. They cannot be used in functions or expressions used in any other contexts.

Using Variables

An expression can contain references to any workflow variable defined in a current workflow.

A workflow variable reference can use any one of the following syntaxes:

- `:variable`
- `:'variable'`
- `:"variable"`
- `$variable`
- `$'variable'`
- `$"variable"`

Using Operators

The following table describes the available operator values.

Table 8-2 Operator Usage

Operator	Symbol	Syntax	Operand(s)	Result
AND	AND	*expr1* AND *expr2*	Logical	True if expr1 and expr2 are both True; otherwise False
OR	OR	*expr1* OR *expr2*	Logical	True if either expr1 or both expr1 and expr2 are True; otherwise False
XOR	XOR	*expr1* XOR *expr2*	Logical	True if either but not both expr1 or expr2 are True; otherwise False
NOT	NOT	NOT *expr*	Logical	True if expr is False; otherwise False

Table 8-2 Operator Usage

Operator	Symbol	Syntax	Operand(s)	Result
(parentheses)	`()`	`(expr)`	Any expression	Evaluates expr first
Multiply	`*`	`expr1 * expr2`	Numeric	Numeric product
Divide	`/`	`expr1 / expr2`	Numeric	Numeric quotient
Modulo	`%`	`expr1 % expr2`	Numeric	Numeric modulus (remainder when expr1 divided by expr2)
Plus	`+`	`+ expr` `expr1 + expr2`	Numeric String or numeric	Unary plus String concatenation or numeric addition
Minus	`-`	`- expr` `expr1 – expr2`	Numeric Numeric	Unary minus (negates expr) Subtraction
Less than	`<`	`expr1 < expr2`	String or numeric	Logical True if expr1 is less than expr2; otherwise False
Less than or equal to	`<=`	`expr1 <= expr2`	String or numeric	Logical True if expr1 is less than or equal to expr2; otherwise False
Equal to	`=`	`expr1 = expr2`	String or numeric	Logical True if expr1 is equal to expr2; otherwise False
Not equal to	`<>`	`expr1 <> expr2`	String or numeric	Logical True if expr1 is not equal to expr2; otherwise False
Greater than	`>`	`expr1 > expr2`	String or numeric	Logical True if expr1 is greater than expr2; otherwise False
Greater than or equal to	`>=`	`expr1 >= expr2`	String or numeric	Logical True if expr1 is greater than or equal to expr2; otherwise False

Using Functions

Functions are built-in expressions you can use to get run-time data. You can use functions for various purposes, such as typecasting variables, identifying workflow information, and performing operations on data. Many functions also allow you to specify built-in attributes that return specific workflow and system data at run time.

The following sections group workflow functions by category. In the listings, typical function attributes are given in parentheses. Note, however, that functions can contain other functions and expressions, and attributes can also be expressed as embedded expressions.

Note: In addition to the default functions listed in this section, there may also be plug-in functions that you can use.

If you want to assign the results of functions to variables or compare the results of two functions, you will need to ensure that the data types match. Thus, return types are also listed in the following function descriptions.

Obtaining Run-time System Data

Use the functions listed in this section to obtain system information at run-time:

- Date()

Date()

Description	Returns the current system date and time at the moment the expression is evaluated, such as the activation of a node, or the execution of an action.
Format	`Date()`
Return type	Java Date object To format the return value as a string, wrap the `Date()` function in a `DateToString()` function and specify the format. For information, see "DateToString()" on page 8-18.

Extracting Run-Time Event Data

You can use the following functions for obtaining properties and extracting content from the header or XML body of JMS messages destined for workflows:

- EventAttribute()
- EventData()
- XPath()

EventAttribute()

Description	Obtains event properties from an incoming JMS message or plug-in-defined event.

Format	`EventAttribute(`*`expression`*`)`
	`expression` is an expression that returns the name of the JMS header or property field. For information about the JMS header and property fields, see "Message" in "WebLogic JMS Fundamentals" in *Programming WebLogic JMS* at the following URL:
	`http://edocs.bea.com/wls/docs70/jms/fund.html`
	Note: `EventAttribute()` should be used only in the following contexts, in which an incoming JMS message is consumed:
	■ Event nodes
	■ Event-triggered Start nodes
	■ Event Key Expression configurations

Example	`EventAttribute("JMSDestination")`
Return type	Java Object

EventData()

Description	Retrieve the actual message content of an incoming JMS message, or plug-in-defined event. The content could be an XML document. **Note:** `EventData()` should only be used in the following contexts, where an incoming JMS message is consumed: ■ Event nodes ■ Event-triggered Start nodes ■ Event Key Expression configurations
Format	`EventData()`
Return type	Java Object

XPath()

Description	Extracts content from XML documents.
Format	`XPath("xpathstring", [,xmldocument])` *`xpathstring`* is the XPath language expression *`xmldocument`* is an expression that yields the XML document against which the XPath expression is evaluated. This may be a string containing the source text of a valid XML document, or more typically, a reference to an XML or string variable containing the XML text. This parameter is optional. If it is not specified, the XML document is assumed to be the incoming XML event. **Note:** If you do not specify a variable containing an XML document, the `XPath()` function should only be used in the following contexts, where the identity of the incoming XML document is known: ■ Event nodes ■ Event-triggered Start nodes ■ Event Key Expression configurations ■ Send XML to Client action ■ Exception Handler receiving an XML document from the Invoke Exception Handler action
Example	see below
Return type	DOM object Typically you will use the XPath language's `text()` function inside a workflow XPath function expression (see below for more information), which returns a Node List. Other XPath functions, however, can return Double, Boolean, String, and Integer types.

XPath is a language for addressing parts of an XML document. It provides basic facilities for the manipulation of strings, numbers, and Booleans. XPath gets its name from its use of a path notation as in URLs for navigating through the hierarchical structure of an XML document. To obtain the official specifications of the XPath language, refer to the following Internet site: http://www.w3.org/TR/xpath.html

To obtain a quick reference guide to XPath notations and functions, refer to the following Internet site: http://www.mulberrytech.com/quickref/XSLTquickref.pdf

The following are examples of the most common uses of XPath for retrieving text values from nodes, attribute values, selecting sub-trees, selecting nodes by attribute value and so on from an XML document. Consider the XML document in the following listing:

Listing 8-1 Sample XML Document

```
<?xml version="1.0"?>
<a>
  <b name="bill">This is the first value</b>
  <c>
    <d id="d1">This is the second value</d>
    <d id="d2">This is the third value</d>
    <d id="d3">This is the fourth value</d>
    <d id="d4">This is the fifth value</d>
  </c>
</a>
```

To select the textual value of the <b> element (i.e., "This is the first value"):

```
XPath("/a/b/text()")
```

To select the textual value of the <d> element whose ID attribute has the value "d3" (i.e., "This is the fourth value"):

```
XPath("/a/c/d[@id=\"d3\"]/text()")
```

To select the textual value of the second <d> element (i.e., "This is the third value"):

```
XPath("/a/c/d[2]/text()")
```

To select the value of the name attribute of the <b> element (i.e., "bill"):

```
XPath("/a/b/@name")
```

To select the entire <c> subtree:

```
<c>
    <d id="d1">This is the second value</d>
    <d id="d2">This is the third value</d>
    <d id="d3">This is the fourth value</d>
    <d id="d4">This is the fifth value</d>
</c>)
XPath("/a/c")
```

Note: To create XPath expressions, you can either type them yourself, or you can use the XPath Wizard. For details about using the wizard, see "Creating XPath Expressions Using the XPath Wizard" on page 8-31.

XML Element Dot Notation

For variable initialization purposes, in Start or Event Properties, Exception Handler Properties, and Send XML to Client, you can also use a simpler dot notation to obtain element data from an XML document, with the following syntax:

root_element.subelement1.subelement2.subelement3 . . .

Consider the following sample XML document:

Listing 8-2 Sample XML Document

```
<account>
       <number>847365</number>
       <customer>John Doe</customer>
       <balance>
              <status>past due</status>
              <date_due>7-11-2001</date_due>
              <amount_due>5670.85</amount_due>
       </balance>
       <credit_limit>7500.00</credit_limit>
</account>
```

In this example, to retrieve the value past due, you would use the following expression:

```
account.balance.status
```

Note that you cannot use this notation to obtain attribute values, and that the root element must begin the expression. You cannot begin the expression with a sub-element.

Note also that this notation always returns a string, and not be used with typecasting functions to return any other data type.

Obtaining Run-time Workflow Data

Use the functions listed in this section to obtain run-time data from workflows:

- CurrentUser()
- TaskAttribute()
- WorkflowAttribute()
- WorkflowVariable()

CurrentUser()

Description	Returns the ID (user name) of the user currently executing a task.
Format	`CurrentUser()`
Return type	String

TaskAttribute()

Description	Provides information about a workflow task.	
Format	`TaskAttribute("attribute" [,"taskname"])` *attribute* is the task attribute. The following attributes may be used.	
Function Attribute	**Information**	**Return Type**
`TaskId`	System-defined task instance ID	String
`Assignee`	User or role ID of assignee	String
`Priority`	Priority set by the user	Integer
`Due`	Task overdue date	Date
`Name`	Name of task	String
`Started`	Task start date/time	Date
`Completed`	Completed date/time	Date
`Comment`	Task comment set by the user	String
	taskname may be used to specify a task other than the current one in which the expression is defined. **Note:** Only tasks in the same workflow may be specified.	

Note: If you try to use an atrribute other than the ones listed in the above table, the server will throw an exception.

WorkflowAttribute()

Description	Provides information about the current workflow only.	
Format	`WorkflowAttribute("attribute" )` *`attribute`* is the workflow attribute. The following attributes may be used.	
Function Attribute	**Information**	**Return Type**
`InstanceId`	System-defined workflow instance ID	String
`TemplateId`	System-define workflow template ID	String
`TemplateDefinitionId`	System-defined workflow template definition ID	String
`Initiator`	Workflow initiator	String
`ParentId`	System-defined workflow instance ID of a parent workflow instance for a called workflow instance	String
`Name`	Template name set by the user	String
`Started`	Start date/time	Date
`Completed`	Completed date/time	Date
`Label` `Id`	Template definition label set by the user	String
`Comment`	Workflow comment set by the user	String
`ExceptionType` `ErrorType`	Name of Java exception class that raised the error	String
`ExceptionNumber` `ErrorNumber`	Message number of the error being handled by the exception handler. For a list of workflow error messages by number and text, see "System Error Messages" on page 9-15.	Integer

`ExceptionSeverity` `ErrorSeverity`	Five integer values: ■ `0`: Unknown error type, internal use only ■ `1`: A fatal exception occurred while processing a user request ■ `2`: A fatal, illegal condition such as inconsistent workflow state ■ `3`: A non-fatal workflow condition that the user can rectify manually ■ `4`: A custom error raised either by an application calling `WorkflowProcessor.invokeWorkfowErrorHandler`, or by a workflow executing the Invoke Error handler action **Note:** Value is set to 4 when a workflow invokes an exception handler via the Invoke Error Handler action or via the API.	Integer
`ExceptionText` `ErrorText`	Message text of the error being handled by the exception handler For a list of workflow error messages by number and text, see "System Error Messages" on page 9-15.	String
`ExceptionObject` `ErrorObject`	The exception object being handled by the exception handler.	Exception Object

Note: If you try to use an atrribute other than the ones listed in the above table, the server will throw an exception.

WorkflowVariable()

Description	Returns the value of a workflow variable for a particular workflow instance.

Format	`workflowVariable(`*`instanceid`*`, `*`variable`*`)` *`instanceid`* is the workflow instance ID, which would normally be represented by a string-type variable which has been populated with a value provided by the `WorkflowAttribute("InstanceID")` function sent from another workflow. *`variable`* is the workflow variable. For formatting information, see "Using Variables" on page 8-4.
Example	`workflowVariable($InstanceID), $ItemQuantity)`
Return type	Object

Converting Data Types

You can use several functions to convert the return type of one expression to another. For information on workflow type conversion rules, see "Data Type Conversions for Variable Assignment" on page 8-25.

- DateToString()
- StringToDate()
- ToInteger()
- ToString()

DateToString()

Description

Converts a date to a string.

Format

dateToString("date", "format")

date is the date to be converted to the string value. The date must be expressed as a Java date object, so you must embed the following functions to specify the date:

- To specify the current run-time date and time at the moment the expression is evaluated, use `Date()`; see "Date()" on page 8-7.
- To specify the run-time date and time when the current workflow is started or completed, use `WorkflowAttribute("Started")` or `WorkflowAttribute("Completed")`.
- To specify the run-time date and time the current task is started or completed, use `TaskAttribute("Started")` or `TaskAttribute("Completed")`.

format is a string specifying the format of the string. Possible values are listed below.

Note: Format is case sensitive.

Format	Description	Example
`yyyy`	Year	`2000`
`MM`	Numeric month of the year	January = `01`, February = `02`,
`dd`	Numeric day of the month	`02` or `28`
`DD`	3- digit day of the year based on the total number of days (365) in a year	October 18, 2000 = `292` (the 292nd day of the year)
`hh`	Hours in non-military time	1 pm = `01`, 2 pm = `02`
`HH`	Hours in military time	1 pm = `13`, 2 pm = `14`
`mm`	Minutes	`02`
`ss`	Seconds	`35`
`SSS`	Milliseconds	`370`

The following separator characters are valid: `-` (dash); `/` (slash); `:` (colon); `.` (period); spaces

Example	`DateToString(Date(), "yyyy-MM-dd HH:mm:ss.SSS")` resulting in `2000-10-18 14:30:35.370`
Return type	String

StringToDate()

Description	Converts a string to a date.
Format	`StringToDate("`*`string`*`", "`*`format`*`")` *`string`* is the string to be converted to the date value. The string must be formatted according to the format you specify in the second argument. *`format`* is a string specifying the format of the date. For available formats, see "Date Function Formats" on page 8-23.
Example	`StringToDate("2001.09.10", "yyyy.MM.dd")`
Return type	Date

ToInteger()

Description	Converts a string value to an integer. **Note:** The string must represent a valid integer value.
Format	`ToInteger(`*`expression`*`)` *`expression`* is the expression (or string enclosed in double quotes) to be converted to integer.
Example	`ToInteger(ToString(XPath("/item/quantity/text()")))`
Return type	Integer

ToString()

Description	Converts any data type value to a string.
Format	`ToString(`*`expression`*`)` *`expression`* is the expression to be converted to string.
Example	`ToString($TotalPrice)`
Return type	String

Manipulating Data

You can use functions listed in this section to perform various operations on data:

- Abs()
- DateAdd()
- StringLen()
- SubString()

Abs()

Description	Returns the absolute value of an expression.
Format	`Abs(`*`expression`*`)` *`expression`* is a workflow expression for which the absolute value is to be calculated.
Example	`Abs($ItemPrice * $ItemQuantity)`
Return type	Integer, Double, or String, depending on input

DateAdd()

Description

Performs date arithmetic.

Format

`DateAdd(`*`date, "interval", number`*`[,`*`business calendar name`*`])`

`date` is the base date. The date must return a Java date object, so you must embed the following functions to specify the date:

- To specify the current run-time date and time at the moment the expression is evaluated, use `Date()`; see "Date()" on page 8-7.
- To specify the run-time date and time when the current workflow is started or completed, use `WorkflowAttribute("Started")` or `WorkflowAttribute("Completed")`.
- To specify the run-time date and time the current task is started or completed, use `TaskAttribute("Started")` or `TaskAttribute("Completed")`.
- To specify a constant base date, use `StringToDate()` function in the appropriate format; see "StringToDate()" on page 8-19. For formatting information, see "Date Function Formats" on page 8-23.

`interval` is the unit of time to use. Possible values are listed below.

Note: The interval is not case sensitive except for `m` and `M`.

Interval	Description
`S`	Seconds
`m`	Minutes
`H`	Hours
`D`	Days
`W`	Weeks
`M`	Months
`BH`	Business hours
`BD`	Business days

`number` is an integer (optionally signed) indicating the number of units to add or subtract.

`business calendar name` is the name of the business calendar to be used for the business hours and business days calculation. If no calendar is specified, the default calendar is used.

Example	`DateAdd(Date(), D, 7, mycalendar)`
Return type	Date

StringLen()

Description	Returns the length of a string
Format	`stringlen(`*`expression`*`)` *`string`* is the string or expression for which a length will be provided.
Example	`stringlen(TaskAttribute("Comment"))`
Return type	Integer

SubString()

Description	Extracts a substring from a string.
Format	`SubString(`*`expression, start`* `[,`*`length`*`])` *`string`* is the string from which the substring is to be extracted. *`start`* is the starting position in the string. (The first position is 0.) *`length`* is the length of the substring. This is an optional parameter. If it is not included, the substring extracted will be the rest of the string.
Example	`SubString(TaskAttribute("Comment"), 0, 50)`
Return type	String

Date Function Formats

To specify a date format for any of the date functions, use a *time pattern* string to describe dates and times. For example, the following time pattern string used in a `DateToString()` function:

```
"yyyy.MM.dd G 'at' hh:mm:ss z"
```

results in the following formatting:

```
2000.07.31 AD at 13:10:35 PDT
```

Similarly, if you use the `StringToDate()` function, you must format a date literal according to the format you specify.

This section describes pattern letters, provides formatting guidelines, and shows examples.

Table 8-3 Pattern Letter Definitions

Symbol	Description	Format	Example
G	era designator	*text*	AD
y	year	*number*	1996
M	month in year	*month name & 01 - 12*	July & 07
d	day in month	*01 - 31*	10
h	hour in am/pm	*1 - 12*	12
H	hour in day	*0 - 23*	0
m	minute in hour	*number*	30
s	second in minute	number	55
S	millisecond	*number*	978
E	day in week	*day name*	Tuesday
D	day in year	*number*	189
F	day of week in month	*number*	2 (2nd Wed in July)

Table 8-3 Pattern Letter Definitions

Symbol	Description	Format	Example
w	week in year	*number*	27
W	week in month	*number*	2
a	am/pm marker	AM or PM	PM
k	hour in day	*1 - 24*	24
K	hour in am/pm	*0 - 11*	0
z	time zone	*abbreviated text*	PST
'	escape for text		'at'
''	single quote	'	'Wednesday' 's'

Format Guidelines

The count of pattern letters determine the format.

- for `text`: for 4 or more pattern letters, use the full form. For less than 4, use the short or abbreviated form, if one exists.
- for `number`: the minimum number of digits. Shorter numbers are zero-padded to this amount. Year is handled specially; that is, if the count of 'y' is 2, the year will be truncated to 2 digits.
- for *`text or number`*: for 3 or more pattern letters, use text; otherwise, use a number.

Any characters in the pattern that are not in the ranges of `['a'..'z']` and `['A'..'Z']` will be treated as quoted text. For instance, characters like `':', '.', ' ', '#'` and `'@'` will appear in the resulting time text even though they are not within single quotes.

A pattern containing any invalid pattern letter will result in a thrown exception during formatting or parsing.

Examples of Time Patterns

Table 8-4 Examples Using the U.S. Locale

Format Pattern	Result
`"yyyy.MM.dd G 'at' hh:mm:ss z"`	1996.07.10 AD at 15:08:56 PDT
`"EEE, MMM d, ''yy"`	Wed, July 10, '96
`"h:mm a"`	12:08 PM
`"hh 'o''''clock' a, zzzz"`	12 o'clock PM, Pacific Daylight Time
`"K:mm a, z"`	0:00 PM, PST
`"yyyyy.MMMMM.dd GGG hh:mm aaa"`	1996.July.10 AD 12:08 PM

Data Type Conversions for Variable Assignment

Often you will want to assign the resulting value of an expression to a workflow variable. For example, you may want use the data returned by an XPath expression from an XML document to initialize variables when a workflow starts. Or you may wish to assign the value of one variable type to another.

Where the return type of an expression is different from the type of variable to which you want to assign the expression's resulting value, the server automatically performs a type conversion. In some cases, however, a conversion cannot be performed, and an exception will occur. The following table shows how conversions are handled. Numbers in the table correspond to notes following the table.

Note: The conversion rules presented in the following table apply to the default settings for data types. For conversion rules used for null value support, see "Configuring BPM to Support Null Variables" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

Table 8-5 Data Type Conversion Rules

Source Expression Evaluates to:	Target Variable Type:					
	String	**Integer**	**Double**	**Date**	**Boolean**	**XML**
String	String	Integer*	Double*	Date*	Boolean**	See Note *****.
Integer	String	Integer	Integer's Double value	NS	False if Integer value is 0, true if any other value	NS
Double	String	Double's Integer value	Double	NS	False if Double value is 0.0, true if any other value	NS
Date	String	NS	NS	Date	NS	NS
Boolean	"True" or "False"	1 if true, 0 if false	1.0 if true, 0.0 if false	NS	Boolean true or false	NS
XML	DOM serialized to String	Integer***	Double***	Date***	Boolean****	XML
Node List	DOM serialized to String	Integer***	Double***	Date***	Boolean****	XML node***

NS = Not supported. If the conversion is attempted, an error message is displayed indicating that the conversion is illegal.

Node List = the object returned by an XPath text() function.

* If the source value is not a valid input for the target data type, an error message is displayed indicating that the conversion is illegal.

** True if string value is “true”, “t”, or “1”; false if string value is “false”, “f”, “0”. Any other string value will cause an error message to be displayed indicating that the conversion is illegal.

*** DOM object serialized to string, and then converted to the target data type. If the source value is not a valid input for the target data type, an error message is displayed indicating that the conversion is illegal.

**** DOM object serialized to string. True if string value is “true”, “t”, or “1”; false if string value is “false”, “f”, “0”. Any other string value will cause an error message to be displayed indicating that the conversion is illegal.

***** DOM parsing to an XML document element, not simply a node. This means that if it is reconverted to a String, the string will contain an XML header similar to the following: `<?xml version="1.0" encoding="UTF-8"?>`. If the source value is not valid for DOM parsing, an error message is displayed indicating that the conversion is illegal.

Note that these conversion rules are not fully supported for calculations in expressions. For example, assume you have the following expression:

```
XPath("your_expression") + 2
```

The result of this statement cannot be calculated because the data type of *your_expression* is not known. To make this statement valid, you need to use a typecasting function to convert the result to the data type you want.

For example, assuming *your_expression* evaluates to a value of 3, and you want the result to be an string data type. You need to use the following statement:

```
ToString(XPath("your_expression") + 2
```

In this case, the expression will evaluate to a string value of `52`.

Now let us assume that *your_expression* evaluates to a value of 3, and you want the result to be an integer. You need to use the following statement:

```
ToInteger(ToString(XPath("your_expression")) + 2
```

In this case, the expression will evaluate to an integer value of `7`.

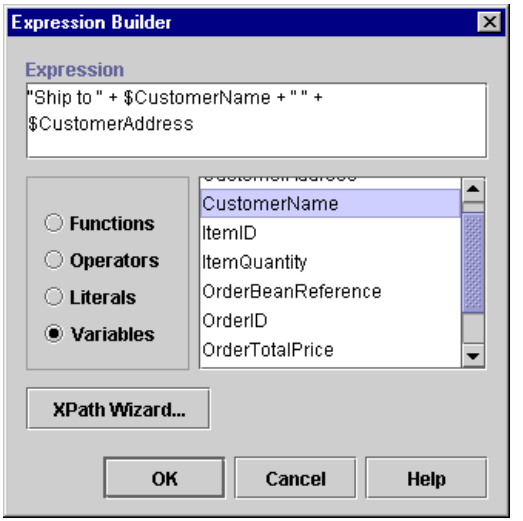
Using the Expression Builder

The Expression Builder helps you to build expressions, that consist of functions, operators, literals, and variables. Throughout the Studio, dialog box fields have an Expression Builder button, shown in the following figure.

Figure 8-1 Expression Builder Button

You click the button to display the Expression Builder dialog box.

Figure 8-2 Expression Builder Dialog Box



Note: If a plug-in is defined for functions that are part of expressions, this dialog box contains the new function.

To build an expression:

1. Do any of the following:
   - Type the expression or an expression component directly in the Expression field.
   - Select from the Functions, Operators, Literals, and Variables options, and double-click a component from the scrollable list to the right to place the selected expression text in the Expression field. In some cases, the inserted item contains placeholders. You can either replace the placeholders by typing the correct information or by inserting additional components from the component lists.
   - For the XPath function, optionally click XPath Wizard to open the XPath Wizard to automatically generate XPath function expressions from sample XML documents. For details about using the wizard, see "Creating XPath Expressions Using the XPath Wizard" on page 8-31.
2. When the expression in the Expression field is complete, click OK to return the expression to the source dialog box field.

If you enter an invalid expression, the system displays a message that attempts to explain why the expression is invalid. The following table lists the messages that may be encountered, together with possible causes.

Table 8-6 Invalid Expression Messages

Message	Possible Cause(s)
Invalid *operator* operand	One or both of the operand(s) to an AND, OR, XOR, NOT,*,/,%,+,-,<,<=,=,<>,>=,> operator was not valid. For example, attempted arithmetic on a non-numeric string: `"Name: " * 25.4`
Mismatched operand types in *operator* comparison	Attempting to compare values of different types. For example, comparing a string to a number: `"mystring" <= 56.9`.
NumberFormatException	A string representation of a number did not have the required format for the type of number. For example: `1.23ZX`
Invalid function name	The expression calls an undefined function. For example: `"Name: " + somefunc()` is not a valid function name.
Incomplete escape sequence	The last character in a quoted string was a backslash character that was not followed with one of the following characters: r, n, f, t, \, ', ",0.
Invalid escape sequence	A quoted string contained a backslash character that was not followed with one of the following characters: r, n, f, t, \, ', ",0.
Fatal error	An internal error occurred in the expression evaluator. The expression probably contains a syntax error.
Unclosed string	A string literal did not have a matching quote at the end. In the following example, a double quote is not matched by a single quote: `"This is an unclosed string'`
Illegal character	The expression contains a character that violates the syntax of the expression language. In the following example, ! is the invalid character: `7 * $wks + " days" !`
Error: unmatched input	The evaluator could not interpret the expression. The expression probably contains a syntax error.

Table 8-6 Invalid Expression Messages

Message	Possible Cause(s)
Unrecoverable syntax error	The expression contains an invalid token. In this example, name is invalid because the + operator was omitted: `"Name: " $name` The Expression should have been: `"Name: " + $name`

The Expression Builder does not check the validity of the following:

- The data type of a workflow variable referenced by the expression.
- The correct number of parameters being passed to a function.
- Type matching. If there is a type mismatch in the expression (for example, performing arithmetic on a non-numeric string), a run-time error results when the expression is evaluated.
- Function attributes. If you use an attribute that is not valid for a function, a run-time exception results.

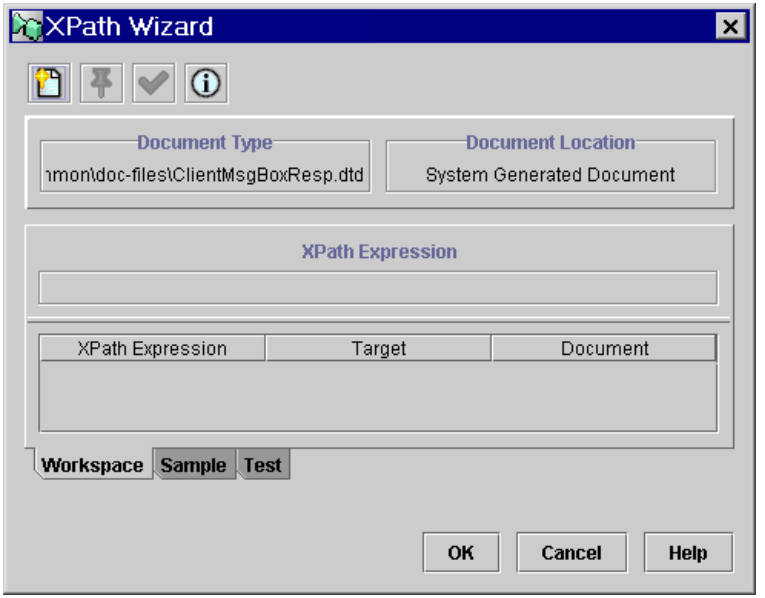
Creating XPath Expressions Using the XPath Wizard

The XPath wizard provides a graphical interface that you can use to generate XPath expressions automatically from actual XML entities, or define and test XPath expressions that define and validate against XML entities. Once you have generated and tested an XPath expression, you can return it to the Expression Builder which appends any additional arguments required to conform to workflow syntax.

The XPath Wizard can sample XML documents, or Document Type Definition (DTD), XML Schema document (XSD), Extensible Stylesheet Language (XSL) or Message Format Language (MFL) files. If you load a DTD, Schema, XSL, or MFL file into the wizard, it automatically generates a sample XML document based on the specifications in the file you selected. You use the generated XML document to create the XPath expression.

To open the XPath Wizard, from the Expression Builder, click XPath Wizard.

Figure 8-3 XPath Wizard



The XPath Wizard contains three tabs: Workspace, Sample, and Test, which you use to do the following:

- Workspace tab—View all the XPath location expressions and functions you create during a Studio session and select them one by one to test on the Test tab, or to return them to the Expression Builder. See "Viewing XPath Expressions" on page 8-36.
- Sample tab—Generate XPath expressions for sample XML, DTD, MFL, and Schema documents. Procedures are given in "Generating XPath Location Expressions from XML Entities" on page 8-33.
- Test tab—Test XPath expressions by applying them to XML documents. See "Testing XPath Expressions" on page 8-38.

All three tabs display a toolbar and two common read-only fields:

- Document Type—indicates whether the currently loaded document is associated with a type definition or not. If the loaded file is a DTD or XSD, the path and filename are given. For other documents, no type or identification is given.
- Document Location—indicates whether the currently loaded document is a system-generated sample document or not. If the loaded document is an XML, MFL or XSL document, the path and filename are given.

Toolbar buttons are described in the following table.

Table 8-7 XPath Wizard Toolbar Buttons

Button	Description
	Load an XML entity into the XPath Wizard. For more information, see "Generating XPath Location Expressions from XML Entities" on page 8-33.
	Pin an XML expression to the Workspace. For more information, see "Generating XPath Location Expressions from XML Entities" on page 8-33.
	Test a selected XML expression against an XML entity. For more information, see "Testing XPath Expressions" on page 8-38.
	Display information about the XPath wizard.

Generating XPath Location Expressions from XML Entities

Using the XPath wizard, you can generate XPath location expressions that target:

- Element content
- Attribute content
- Nodes

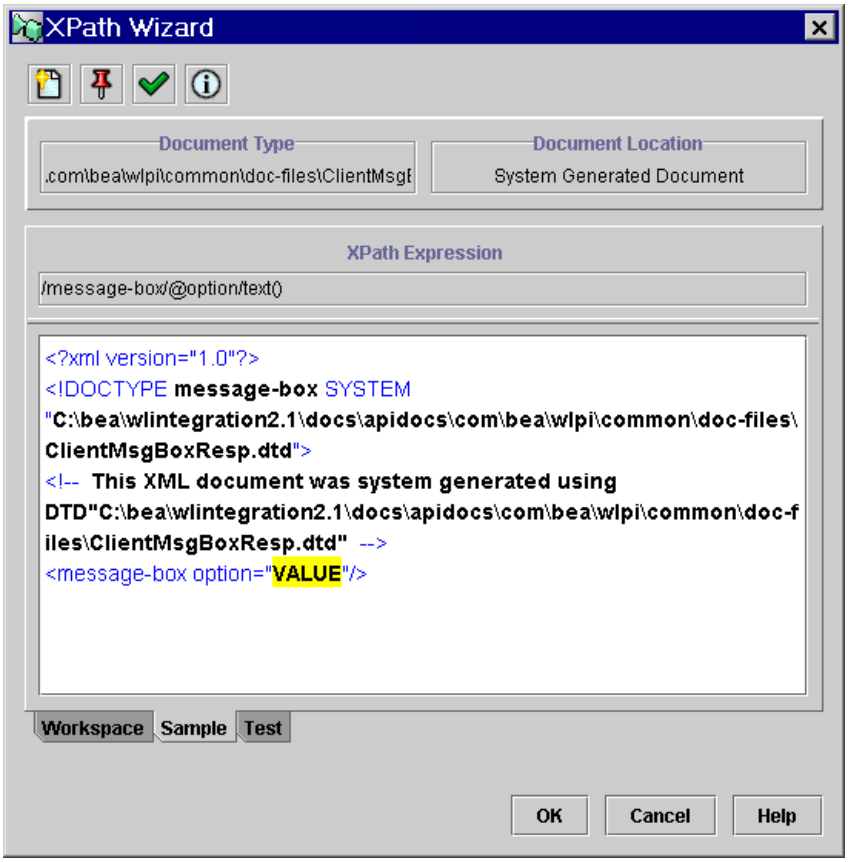
You can load XML, MFL, XSL, DTD or XSD documents for sampling, If you open a DTD or XSD document, the wizard generates a sample XML document from the DTD or Schema you selected. You use the generated XML document to create the XPath expression.

To generate an XPath location expression:

1. From the Expression Builder dialog box, click XPath Wizard to display the XPath Wizard.
2. In the XPath Wizard, select the Sample tab to open the Sample area where you can create XPath expressions.
3. Click the  button to open the XML Finder dialog box. Use the XML Finder dialog box to retrieve an XML, DTD, XSD, MFL, or XSL document that you can use the create the XPath expression. For details, see "Retrieving XML Entities" on page 7-18.

   After you retrieve the document using the XML Finder, the content of the document appears in the XPath wizard. If you open a DTD or XSD document, the wizard generates a sample XML document from the DTD or Schema you selected.

Figure 8-4 XPath Wizard: Sample Tab with Sample XML Document



4. Select the markup or content for an element, attribute, or node that you want to be the target of the XPath expression. The wizard generates an XPath expression based on your selection, and displays it in the XPath Expression field.

5. Put the XPath expression you created in the Workspace by clicking the  button.

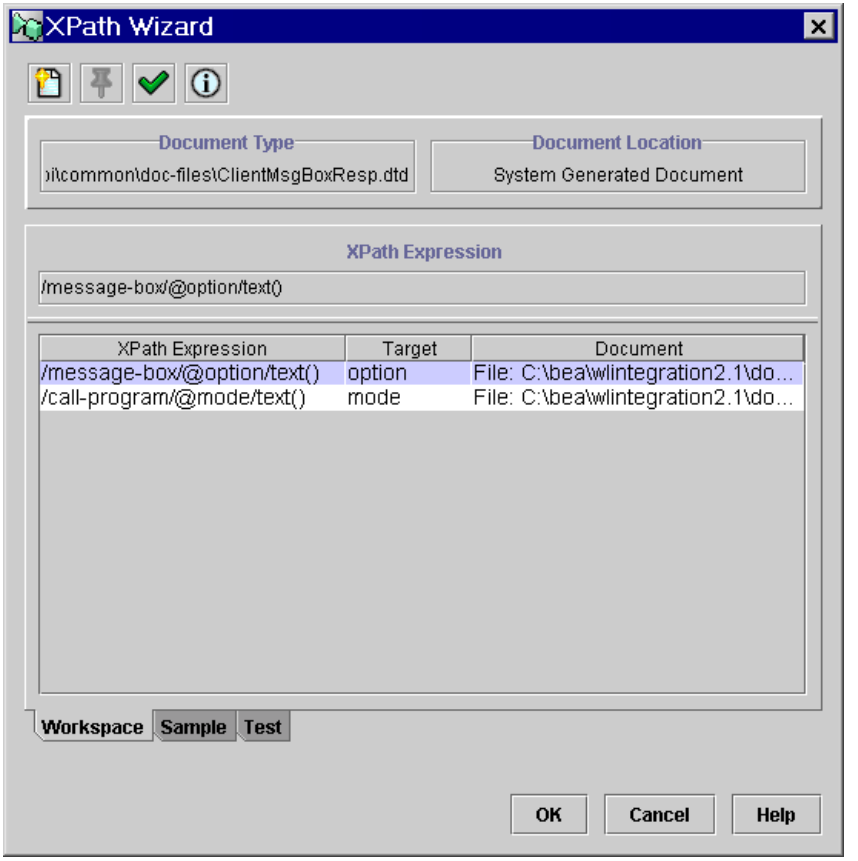
6. To view the expressions you create, select the Workspace tab. For details, see "Viewing XPath Expressions" on page 8-36.

7. To test the expressions you create, select the Test tab. For details, see "Testing XPath Expressions" on page 8-38.
8. Click OK to return the expression to the Expression Builder.

Viewing XPath Expressions

The Workspace is the area in the XPath Wizard where you can hold all the XPath expressions you create during a session of the Studio. Every time you create an XPath expression, you can pin it to the Workspace. This way, you can create multiple expressions for a single XML document, and then test them later to ensure they return the correct result.

Figure 8-5 XPath Wizard: Workspace Tab



The Workspace shows you the following information for each XPath Expression:

XPath Expression	The XPath location expression generated on the Sample tab or entered on the Test tab.
Target	The element or attribute in the XML document that is the target of the XPath expression.
Document	The source document from which the XPath expression was generated.

To test an expression, select the expression in the list, click the Test tab, and follow the procedures in "Testing XPath Expressions" on page 8-38.

To return an expression to the Expression Builder, select the expression in the list, and click OK. This closes the XPath Wizard window and places the selected expression in the Expression Builder.

Testing XPath Expressions

You can use the testing feature of the XPath Wizard for several purposes:

- To test a location expression you have built in the Expression Builder or that you have generated from a sample document. After you have generated XPath location expressions from one document, you may want to test them against other sample documents to be sure that they still return the content you expect.
- To validate other types of XPath expressions, including XPath language functions, that you create yourself, against a sample document. You can pass expressions that you have built in the Expression Builder directly to the Wizard, you can append an XPath function to a location previously generated by the Sample feature, or you can create the expression from scratch in the Wizard. All of the functions of the XPath language are supported by the testing feature of the XPath Wizard, including:
  - String functions that perform operations on the string contents of nodes and attributes, such as accessing, extracting, and combining
  - Boolean functions that perform Boolean tests or operations on their arguments, and that return a value based on the test or operation
  - Number functions that return numeric results, such as a count of nodes in an document, or the sum of numeric content
  - Node set functions that return information about individual nodes or node sets

  For a full list of functions specified by the XPath language, see http://www.w3.org/TR/xpath.html.

  After you have built your expression, the XPath Wizard returns the value that is calculated.

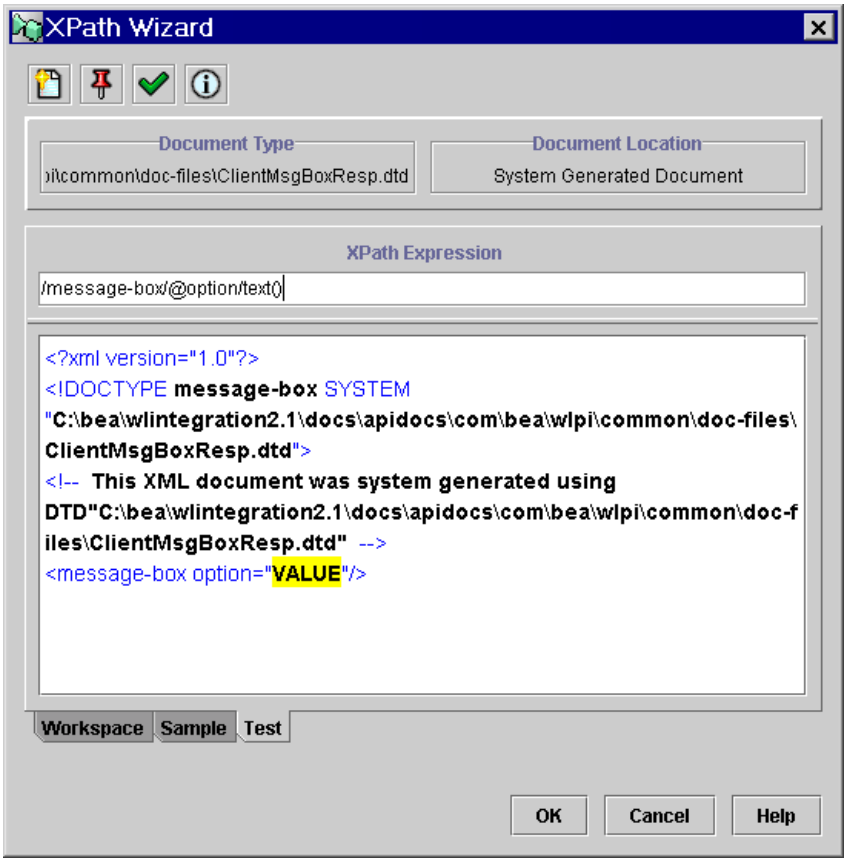
Testing Location Expressions

To test an XPath location expression:

1. Optionally, from the Expression Builder, enter the XPath expression you want to test, and click XPath Wizard. The XPath expression appears as the first item in the XPath Expression list in the Workspace.
2. Select the Sample tab, and click the Open button to open the XML Finder dialog box. Use the XML Finder dialog box to retrieve an XML, DTD, XSL, XSD, or MFL document that you will use to test the XPath expression. For details, see "Retrieving XML Entities" on page 7-18.

   After you retrieve the document using the XML Finder, the content of the document appears in the XPath wizard. If you open a DTD or XSD document, the wizard generates a sample XML document from the DTD or Schema you selected.
3. If you did not pass an expression from the Expression Builder as in step 1, generate an XPath location expression from the sample document, as described in "Generating XPath Location Expressions from XML Entities" on page 8-33.
4. In the Workspace, select the expression you want to test.
5. Select the Test tab. The Test area shows you the XML document with the target of the XPath expression highlighted.

Figure 8-6 XPath Wizard: Test Tab



6. If desired, modify the XPath expression by editing it in the XPath Expression field.
7. To test the expression again, so that the value is highlighted in the document, click the  button. The new result of the expression is highlighted in the XML document.
8. To return the expression to the Expression Builder, click OK.

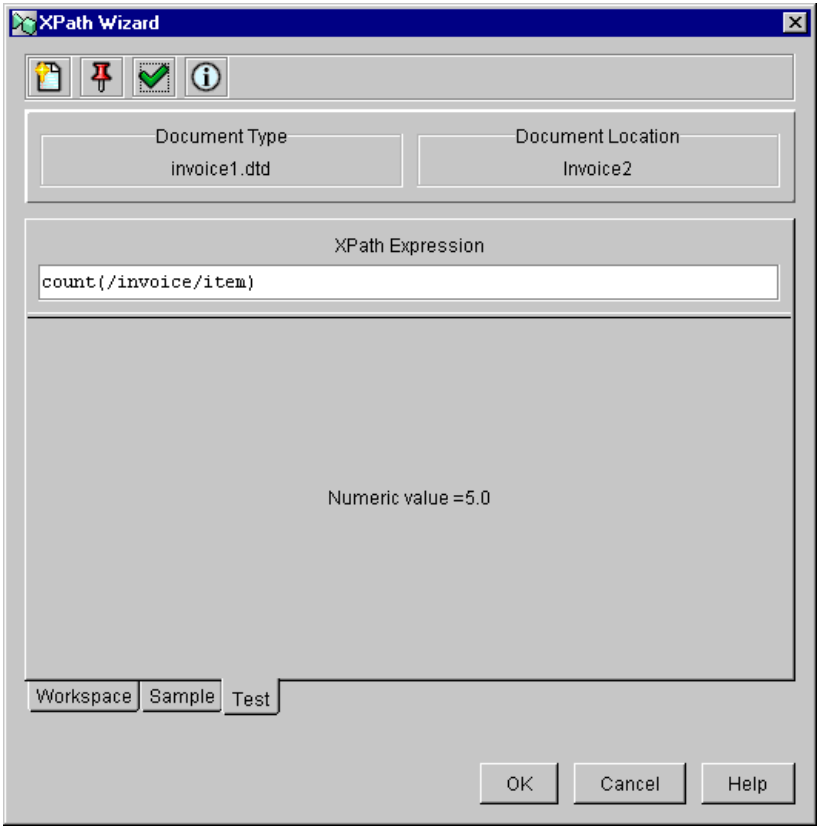
Testing Expressions That Contain Functions

To test an XPath expression containing a function:

1. Optionally, from the Expression Builder, enter the XPath expression you want to test, and click XPath Wizard. The XPath expression appears as the first item in the XPath Expression list in the Workspace.
2. Select the Sample tab, and click the Open button to open the XML Finder dialog box. Use the XML Finder dialog box to retrieve an XML, DTD, or Schema (XSD) document that you will use to test the XPath expression. For details, see "Retrieving XML Entities" on page 7-18.

   If you open a DTD or XSD document, the wizard generates a sample XML document from the DTD or Schema you selected. You use the generated XML document to test your XPath expression against.
3. Optionally, generate an XPath location expression from the sample document, as described in "Generating XPath Location Expressions from XML Entities" on page 8-33.
4. Select the Test tab.
5. In the XPath Expression field, enter your expression, or edit or append a function to an expression you have placed there from the workspace.
6. Click the  button. The return value of the function is displayed in the Test area.

Figure 8-7 Result of an XPath Function



7. If necessary, modify the XPath function by editing it in the XPath Expression field.

8. To test the function again, click the  button. The return value of the function is displayed in the Test area.

9. To return the function to the Expression Builder, click OK.

CHAPTER

9 Handling Workflow Exceptions

The following sections explain how to handle internally and externally generated workflow exception conditions:

- About Workflow Exception Handling
- Overview of Exception Handler Definition Tasks
- Defining Exception Handlers
- Invoking an Exception Handler from a Workflow
- System Error Messages

About Workflow Exception Handling

The workflow exception handling facility enables you to define, trap, and respond to internally and externally generated exception conditions at run time. Exceptions may be specific abnormal conditions that you can target within the workflow, or they may be typical run-time server exceptions that you trap and respond to accordingly.

Exception handling within WebLogic Integration is performed at the workflow level, rather than at the task level. All workflow template definitions have at least one exception handler, the system exception handler. The system exception handler, is by default, the initial exception handler and is invoked whenever an exception occurs. The concept of the *initial* exception handler is required, since exceptions can occur prior to a specific exception handler being invoked from within the workflow. For example,

exceptions can occur during the initialization of variables in a Start node. The system exception handler responds to exceptions by marking the active transaction for rollback only and rethrowing the exception to the client.

The workflow exception handling facility allows you to define customized exception handlers that specify actions to be executed in response to exceptions. Exception handlers have commit and rollback processing paths. You specify certain actions within a workflow to be executed in each of these paths. When an exception occurs, if the currently active transaction is marked for rollback only, the exception handler executes the rollback processing path for the transaction. If the transaction is not marked for rollback only, the exception handler executes its commit path. For information on the workflow transaction model, see "Understanding the BPM Transaction Model" in *Programming BPM Client Applications*.

You can also use workflow functions in conditional expressions to identify particular exceptions that you want to catch, by type, severity, text, and other criteria. For more information, see "Obtaining Run-time Workflow Data" on page 8-13.

Once you have defined the custom exception handler and its actions, you can invoke it in three ways:

- By setting it to act as the initial exception handler. This causes the specified custom exception handler to catch exceptions as soon as the workflow is instantiated.
- By using the Set Workflow Exception Handler action in any workflow node. This causes the custom exception handler to be the active one from that point until the next Set Workflow Exception Handler action is reached within that workflow or until the workflow completes, whichever occurs first.
- By using the Invoke Exception Handler action in any workflow node.This causes the actions defined within the specified exception handler to be executed at that point, regardless of whether an exception has occurred. It also, optionally, sends an XML document to the exception processor, allowing workflow variables to be populated by values from this XML document. This action is only typically used in a conditional situation to catch an exception specifically thrown by the workflow definition.

Overview of Exception Handler Definition Tasks

Exception Handlers are like sub-workflows within a workflow in which you can specify various actions on commit and/or rollback paths of the transaction in which an exception has occurred. The exception handler can respond to general exception occurrences, or the handler can use a condition to specify a specific exception to trap. To use a custom exception handler in your workflows, you need to do the following:

Note: You may also want to familiarize yourself with the workflow expression language and the Studio's Expression Builder and XPath Wizard tools before beginning to define exception handlers and actions that reference them. Many of the tasks described in this section, such composing an XML document in the Invoke Exception Handler action, or initializing variables in an exception handlers, require entering expressions into dialog box fields. For complete information on workflow expressions, see Chapter 8, "Using Workflow Expressions."

1. Create an exception handler and, optionally, set it to be the initial exception handler. Procedures are given in "Creating a Custom Exception Handler" on page 9-5.
2. Add actions to the exception handler. For complete information on actions, see Chapter 6, "Defining Actions."
3. Add an exit method to the exception handler. Procedures are given in "Exiting an Exception Handler" on page 9-7.
4. Add an action to a workflow node that invokes the exception handler. Procedures are given in "Invoking an Exception Handler from a Workflow" on page 9-11.

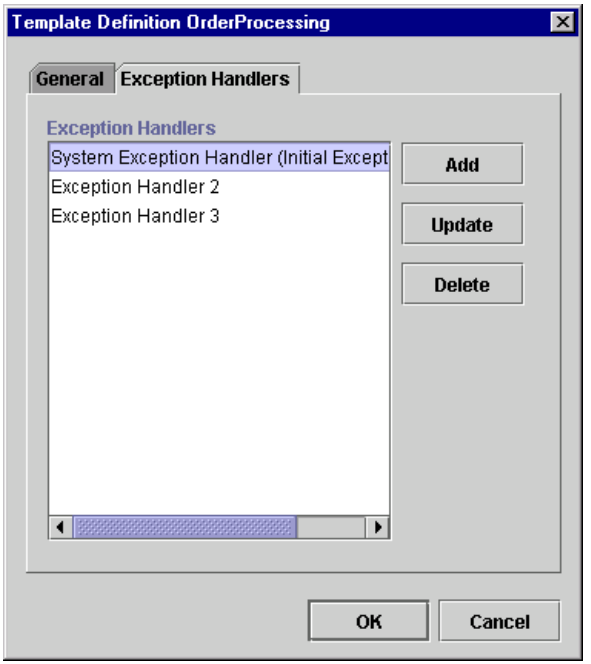
Defining Exception Handlers

You use the exception handler definition facility to define custom exception handlers. If you do not define any exception handlers, the system exception handler is, by default, used.

You can set the custom exception handler to be the initial exception handler, that is, the one that is active as soon as a workflow is instantiated. The template definition can change its active exception handler throughout the course of the workflow by using the Set Exception Handler action. (See "Setting the Workflow Exception Handler" on page 9-12 for information.)

Custom exception handlers defined for the current template definition are displayed in the folder tree under the Exception Handlers folder, and in the properties dialog box for a Template Definition.

Figure 9-1 Template Definition Properties Dialog Box: Exception Handlers Tab



When you define a custom exception handler, you can specify actions to be performed for both commit and rollback exception situations. For a commit paths, all workflow actions are available to be performed. For a rollback path, only the following actions are available: Post XML Event, Call Program, Perform Business Operation, Exit Exception Handler, No Operation, and Make Audit Entry. For more information about actions, see Chapter 6, "Defining Actions."

Note: Certain Enterprise Java Beans (EJBs) can either mark a transaction for rollback only by calling a UserTransaction method, or they can throw an unchecked exception across a container boundary. In either of these two cases, WebLogic Server rolls back the transaction.

You can also define workflow variables to be populated by values from the XML message defined within the Invoke Exception Handler action. For more information, see "Invoking an Exception Handler" on page 9-13.

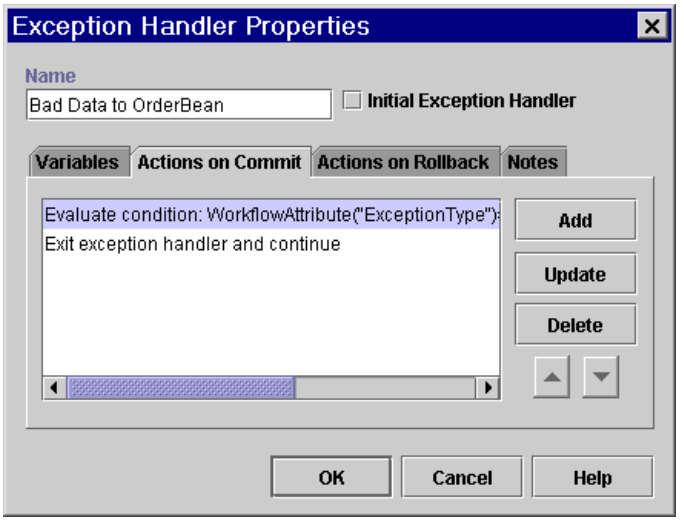
Creating a Custom Exception Handler

To create an exception handler for a workflow template definition:

1. Do one of the following:
   - In the folder tree, expand the folder for the template definition, right-click the Exception Handlers folder and select Create Exception Handler.
   - From the Exception Handlers tab of the template definition's properties dialog box, click Add.

   The Exception Handler Properties dialog box is displayed.

Figure 9-2 Exception Handler Properties Dialog Box



2. In the Name field, enter a meaningful and easily identifiable name for the new exception handler.
3. Optionally, select the Initial Exception Handler check box to mark the exception handler as the default initial exception handler in the workflow. When a workflow encounters an exception, this is the first exception handler that is called.
4. Optionally, on the Variables tab, click Add to display the Workflow Variable Assignment dialog box, which you can use to initialize any variables you have created in the workflow, by specifying an expression that is evaluated at run time to become the value of the variable.
5. From the Variable drop-down list, select a variable to store incoming data.
6. In the Expression field, enter the expression that is evaluated at run time to produce the value for the variable, by doing one of the following:
   - To specify a constant, use the syntax provided in “Using Literals” on page 8-2.
   - If you will be using the Invoke Exception Handler action (see “Invoking an Exception Handler” on page 9-13) to send an XML message containing values to populate variables, use an `XPath()` function (for information, see

"XPath()" on page 8-10), or the dot notation for XML elements (for information, see "XML Element Dot Notation" on page 8-12). You can also use the Expression button  to invoke the XPath Wizard, from which you can generate XPath expressions automatically from a sample incoming document. For information, see "Creating XPath Expressions Using the XPath Wizard" on page 8-31.

7. Click OK. The variable initialization appears in the list on the Variables tab of the Exception Handler Properties dialog box.
8. Repeat steps 4 to 7 for all variables you want to initialize.
9. In the Actions on Commit tab, use the Add, Update or Delete buttons to specify actions to be performed if the current active transaction is not marked for rollback only. For information on working with actions, see Chapter 6, "Defining Actions."
10. In the Actions on Rollback tab, you specify the add, update, or delete actions to be performed if the currently active transaction is programmatically marked for rollback only. The workflow is rolled back to the beginning of the currently active transaction.
11. On the Actions on Commit or Actions on Rollback tab, add the Exit Exception handler action, to return program control to the main workflow. For more information, see "Exiting an Exception Handler" on page 9-7.
12. Click OK to add the custom exception handler. The new exception handler appears on the Exception Handlers tab of the template definition's properties dialog box, and under the Exception Handlers folder in the folder tree.

Exiting an Exception Handler

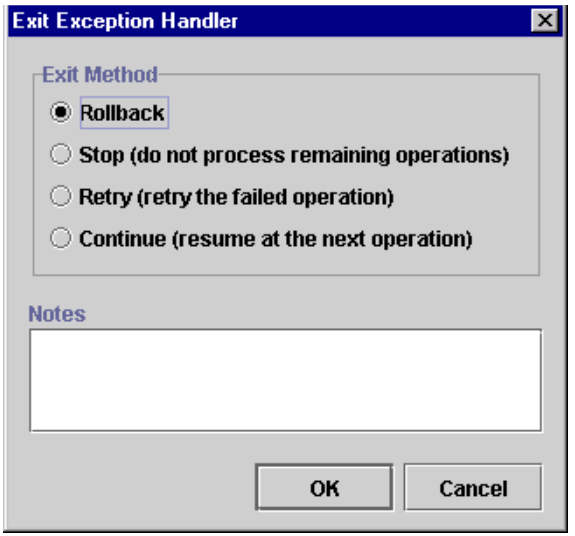
You use the Exit Exception Handler action, which is only available from within the Exception Handler Properties dialog box, to exit an exception handler. (For information, see "Defining Exception Handlers" on page 9-4.) Be sure to add this action to the Actions on Commit or Actions on Rollback tab of the Exception Handler to return control to the main flow.

In a Rollback path, the only exit option available is Rollback. In a Commit path, you can choose from four options:

- Rollback — Marks the user transaction for rollback only, returning the workflow to the state in which it existed prior to the start of the transaction. The exception is propagated to the client (if any) as a WorkflowException.
- Stop — Exits the exception handler and does not execute any follow-on actions.
- Retry — Exits the exception handler and attempts to retry the failed operation. (The failed operation is an action or the setting of a variable.)
- Continue — Stops the execution of the exception handler and attempts to continue the execution of the workflow at the next operation. (The next operation is the next action or the next variable to be set.)

Note: Certain Enterprise Java Beans (EJBs) can either mark a transaction for rollback only by calling a UserTransaction method, or they can throw an unchecked exception across a container boundary. In either of these two cases, WebLogic Server rolls back the transaction.

Figure 9-3 Exit Exception Handler



To exit an exception handler:

1. From within the Exception Handler Properties dialog box, invoke the Add Action dialog box, expand the Exception Handling actions folder, select Exit Exception Handler, and click OK to display the Exit Exception Handler dialog box.

If you are adding the action to the Actions on Rollback tab, the only available option is Rollback. If you are adding the action to the Actions on Commit tab, select one of the Exit Method options.

2. Click OK to add the action.

Updating a Custom Exception Handler

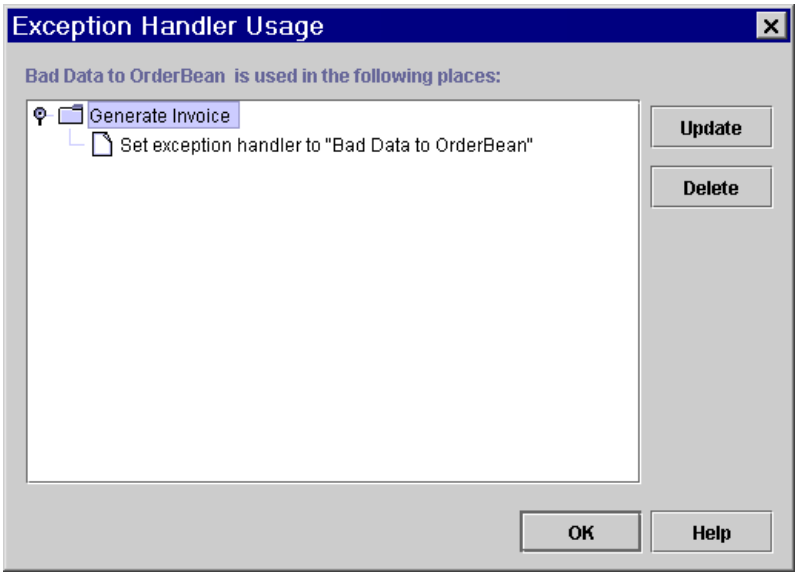
1. Do one of the following to display the Exception Handler Properties dialog box:
   - In the folder tree, expand the Exception Handlers folder, right-click the desired exception handler folder, and select Properties from the pop-up menu.
   - From the Exception Handlers tab of the template definition's properties dialog box, select the desired exception handler and click Update.
2. Make any necessary changes to the exception handler.
3. Click OK to save your changes.

Viewing Exception Handler Usage

To see where an exception handler is referenced within the workflow:

1. In the folder tree, right-click the exception handler folder and select Usage from the pop-up menu to display the Exception Handler Usage dialog box, which lists the places within the workflow where the selected exception handler is referenced.

Figure 9-4 Exception Handler Usage Dialog Box



2. Optionally, use the following buttons in the dialog box to do the following:
   - Update — select to open the dialog box for the selected object.
   - Delete — select to delete the selected object.
3. Click OK to close the Exception Handler Usage dialog box.

Deleting a Custom Exception Handler

Once defined, an exception handler can only be deleted if it is not referenced by either the Invoke Exception Handler action or the Set Workflow Exception Handler action (for more information, see “Invoking an Exception Handler” on page 9-13). To see a list of places where an exception handler is referenced, follow the procedure in “Viewing Exception Handler Usage” on page 9-9.

To delete an exception handler:

1. Do one of the following:

- In the folder tree, expand the Exception Handlers folder, right-click the desired exception handler folder, and select Delete from the pop-up menu.
- From the Exception Handlers tab of the template definition's properties dialog box, select the desired exception handler and click Delete.

2. When prompted, confirm the deletion.

Invoking an Exception Handler from a Workflow

To invoke an exception handler from a workflow, you use the following exception handling actions:

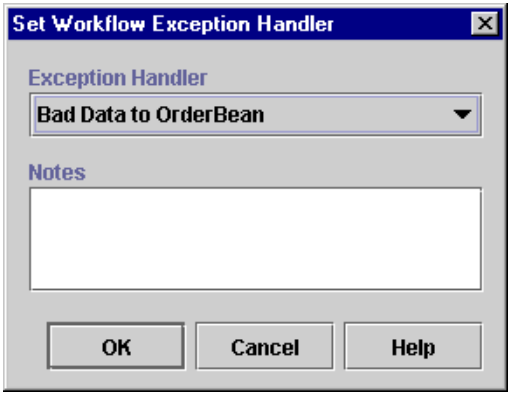
- Set Workflow Exception Handler—this action sets the exception handler for the workflow to the one you specify. All exceptions from this point, until this action is specified again, will be caught by the exception handler you specify. For more information, see "Setting the Workflow Exception Handler" on page 9-12.
- Invoke Exception Handler—this action invokes the actions defined in a custom exception handler, and optionally sends an XML message to the exception processor, at a certain point in the workflow. It is typically used in a conditional situation to perform alternate actions for an exception defined specifically in the workflow. For more information, see "Invoking an Exception Handler" on page 9-13.
- Exit Exception Handler—this action is used from within an exception handler to return control to the main workflow. For more information, see "Exiting an Exception Handler" on page 9-7.

Setting the Workflow Exception Handler

You use the Set Workflow Exception Handler action to make a specified exception handler the active exception handler for a workflow template definition. If this action is not used in the workflow template definition and no other exception handler is defined and marked as the initial exception handler, the system exception handler, by default, is used and invoked whenever an exception occurs.

The exception handler you set for the workflow persists throughout the instance of the workflow until you use a subsequent Set Workflow Exception Handler action to reset the exception handler to the system handler or to another custom-defined one.

Figure 9-5 Set Workflow Exception Handler



To set the workflow exception handler to a custom handler:

1. From the Add Action dialog box, expand the Exception Handling actions folder, select Set Workflow Exception Handler, and click OK to display the Set Workflow Exception Handler dialog box.
2. From the Exception Handler drop-down list, do one of the following:
   - To set the workflow exception handler to a custom one you have defined, select the exception handler from the Exception Handler drop-down list.
   - To reset the workflow exception handler to the default system exception handler, select (system exception handler) from the Exception Handler drop-down list.
3. Click OK to add the action.

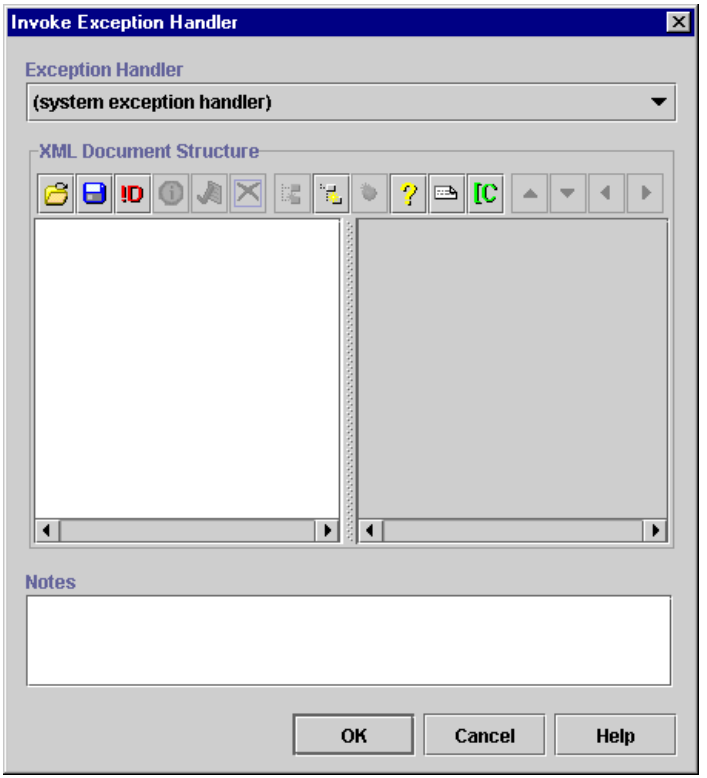
Invoking an Exception Handler

You use the Invoke Exception Handler action to invoke a specific exception handler within the workflow and, optionally, send an XML document that you define to the exception handler. You can use the XML document to send values that will be used to populate variables when the exception handler is invoked. (For information, see "Defining Exception Handlers" on page 9-4.)

Note that this action does not override the exception handler that is set as the active one. Any exceptions that occur are still handled by that exception handler. Furthermore, upon execution of the Invoke Exception Handler action, actions defined in the exception handler are executed, regardless of whether an exception occurs. Thus, you typically use this action in a conditional situation where you want to invoke the actions defined in the exception handler as a result of an exception being thrown from within the workflow itself. This action will not handle exceptions thrown by business operations.

You can also use this action to simply send an XML document to any exception handler you select.

Figure 9-6 Invoke Exception Handler Dialog Box



To invoke a custom exception handler:

1. From the Add Action dialog box, expand the Exception Handling actions folder, select Invoke Exception Handler, and click OK to display the Invoke Exception Handler dialog box.
2. From the Exception Handler drop-down list, do one of the following:
   - Leave the Exception Handler field set to the default system exception handler to simply send an XML document to the event processor at this point in the workflow.
   - Select a custom exception handler you have defined. The actions in the custom-defined exception handler specified here will be invoked when this action is executed, regardless of whether an exception has actually occurred.

Note that any exceptions that occur at this point are still handled by the exception handler set as the workflow exception handler.

3. Optionally, define an XML document to send to the event processor when the exception handler is invoked. To specify the XML Document Structure, do one of the following:
   - To create a new free-form document, begin composing the document by clicking the Add Child button to begin adding nodes.
   - To specify an existing XML document, load the document by clicking the Import button.
   - To create a new type-specified XML document, load the appropriateSchema document by clicking the Set Content Type button.

   For detailed procedures for all these options, see "Composing and Editing XML Documents" on page 7-2.

   The XML document you define is stored in the workflow template definition.
4. Click OK to add the action.

System Error Messages

As described in "Obtaining Run-time Workflow Data" on page 8-13, the `WorkflowAttribute()` function may be used with four attributes that allow the exception handler to interrogate the following information:

- The Java exception class name
- The error number
- The error message text
- The error severity code (which also tells you if the exception handler was called—via action or API—or was raised as a result of a caught exception):
  - `0`: Unknown error type, internal use only
  - `1`: A fatal exception occurred while processing a user request
  - `2`: A fatal, illegal condition such as inconsistent workflow state

- `3`: A non-fatal workflow condition that the user can rectify manually
- `4`: A custom error raised either by an application calling `WorkflowProcessor.invokeWorkfowErrorHandler`, or by a workflow executing the Invoke Error handler action

The following table lists workflow error messages by number and text.

Table 9-1 Workflow Error Messages

ErrorNumber	ErrorText
0	Unknown error
1	System error
2	Workflow error
3	Workflow warning
4	Nested exception is:
5	The server was unable to complete your request.
6	The server was unable to complete your request.
7	This workflow cannot be modified, because it is complete.
8	This task's properties do not allow it to be marked done.
9	This task's properties do not allow it to be unmarked done.
10	This task's properties do not allow it to be reassigned.
11	This task's properties do not allow it to be modified.
12	Cannot take this task, because it is already done.
13	Cannot assign this task, because it is already done.
14	Cannot execute this task, because it is already done.
15	Cannot execute this task, because it is inactive.
16	Cannot execute this task, because the workflow is complete.
17	Cannot execute this task, because it is not assigned to you.

Table 9-1 Workflow Error Messages

ErrorNumber	ErrorText
18	Source and destination of reroute must be different.
19	The effective date must be on or before the expiry date.
20	Source user is already rerouted during the period specified.
21	Specified reroute would create a circular reference.
22	Task names must be unique within a workflow.
23	"{0}" is already in use as an organization name.
24	Role names must be unique within an organization.
25	The specified user "{0}" is already defined.
26	Report names must be unique.
27	Variable names must be unique.
28	Workflow template names must be unique.
29	Workload graph names must be unique.
30	"{0}" is already used as business calendar name.
31	User "{0}" is currently logged on and cannot be deleted.
32	Cannot delete user "{0}", because there are tasks assigned to the user. Please reassign all tasks first.
33	You do not have permission to maintain users.
34	You do not have permission to maintain roles.
35	You do not have permission to maintain organizations.
36	You do not have permission to define workflows.
37	You do not have permission for workflow monitoring.
38	You do not have permission to do task rerouting.
39	You do not have permission to modify the server configuration.

Table 9-1 Workflow Error Messages

ErrorNumber	ErrorText
40	The variable "{0}" cannot be deleted, because it is referenced by the ID expression or the trigger definition.
41	The variable "{0}" cannot be deleted, because it is referenced by one or more actions.
42	The variable "{0}" cannot be deleted, because it is referenced by one or more decisions.
43	Variable names cannot be blank.
44	Cannot delete role "{0}", because there are tasks assigned to the role. Please reassign all tasks first.
45	The workflow "{0}" cannot be deleted, because it is referenced by one or more actions.
46	The task "{0}" cannot be deleted, because it is referenced by one or more actions.
47	The role "{0}" cannot be deleted, because it is referenced by one or more actions.
48	The user "{0}" cannot be deleted, because it is referenced by one or more actions.
49	This workflow cannot be modified, because it is suspended.
50	Cannot delete a business calendar's template rules.
51	No business calendar ID specified in expression.
52	No XML defined for calendar.
53	The system could not find the specified business calendar "{0}".
54	Business calendar "{0}" has no rules defined for the year {1}.
55	The system could not find the processor for business calendar "{0}".
56	{0}, {1}: unexpected token "{2}" in business calendar rule.
57	{0}, {1}: unexpected character "{2}" in business calendar rule.
58	The timezone identifier "{0}" is not recognized.
59	EJB home environment not set.
61	Unable to get initial context: environment may be invalid; {0}.
62	Unable to load class object for home interface of type "{0}"; {1}.

Table 9-1 Workflow Error Messages

ErrorNumber	ErrorText
63	Unable to load class object for remote interface of type "{0}"; {1}.
64	No object is bound to "{0}" in specified context; {1}.
65	The object bound to "{0}" does not implement home interface "{1}".
66	No home method was found on "{0}" matching "{1}"; {2}.
67	No remote method was found on "{0}" matching "{1}"; {2}.
68	Unable to load class for method return value of type "{0}"; {1}.
69	No remote object(s) were found.
70	EJB "{0}" method invocation failed; {1}.
71	"{0}" object returned from method invocation not of expected type "{1}".
72	Wrong number of parameters for method: expected {1}, found {2}.
73	Unable to load class (wrapped if primitive) for parameter {0} of type "{1}"; {2}.
74	Unable to load class (unwrapped if primitive) for parameter {0} of type "{1}"; {2}.
75	Value for parameter {0} cannot be cast to required type "{1}".
76	The home method did not return a session bean.
77	Workflow template (ID={0}) not found.
78	Workflow template definition (ID={0}) not found.
79	The system could not find an active, effective template definition to start.
80	XML syntax error at line {0}, column {1}.
81	Unable to load class object for "{0}"; {1}.
82	No matching constructor was found on "{0}"; {1}.
83	Creation of new "{0}" failed; {1}.
84	No method was found on "{0}" matching "{1}"; {2}.
85	"{0}" method invocation failed; {1}.

Table 9-1 Workflow Error Messages

ErrorNumber	ErrorText
86	The workflow template is currently locked by {0}.
87	The interval “from” date is invalid.
88	The interval “to” date is invalid.
89	The date "{0}" is invalid.
90	Invalid month name: "{0}".
91	Invalid day name: "{0}".
92	The system could not find the specified task instance: "{0}".
93	Mandatory input variable "{0}" not set.
94	The system could not find the specified join instance: "{0}".
95	The system could not find the specified variable instance: "{0}".
96	The system could not find the specified organization: "{0}".
97	The system could not find the specified role: "{0}".
98	The system could not find the specified user: "{0}".
99	User "{0}" does not belong to organization "{1}".
100	The system could not add user "{0}" to organization "{1}".
101	The system could not add user "{0}" to role "{1}".
102	The system could not remove user "{0}" from organization "{1}".
103	The system could not remove user "{0}" from role "{1}".
104	The specified user "{0}" does not belong to the organization within which the role "{1}" is defined.
105	No security realm has been installed.
106	The installed security realm "{0}" is not listable.
107	The security realm "{0}" is not manageable.

Table 9-1 Workflow Error Messages

ErrorNumber	ErrorText
108	Unable to connect to database.
109	The role "{0}" is empty.
110	Fixup error: missing reference "{0}".
111	Invalid interval unit: "{0}".
112	Illegal type conversion: from "{0}" to "{1}".
113	Cannot instantiate a workflow, because the template definition is inactive.
114	The system could not find the target task "{0}".
115	The system could not find the target event "{0}".
116	The system could not find the target action "{0}".
117	The system could not find the specified business operation "{0}".
118	Error calling program: "{0}" with arguments "{1}".
119	User "{0}" does not have an e-mail address.
120	User "{0}" in role "{1}" does not have an e-mail address.
121	The routing table failed to identify a suitable assignee.
122	No workflow organization defined for user "{0}".
123	Wrong start date expression: "{0}".\n"(1)".
124	The system could not find the error handler "{0}".
125	An exception occurred during error handler processing.
126	An error handler exceeded the maximum number of retries allowed.
127	The application or workflow instance invoked an error handler.
128	The system could not find the specified workflow instance: {0}.
129	An error occurred when parsing an XML document.

Table 9-1 Workflow Error Messages

ErrorNumber	ErrorText
130	The definition was created with a later of version of the product. Upgrade your WebLogic Process Integrator Studio to version {0} or later.

CHAPTER

10 Monitoring Workflows

The following sections discuss workflow monitoring:

- Overview of Workflow Monitoring Tasks
- Working with Workflow Instances
- Viewing User and Role Worklists
- Changing Task Permissions and Priority
- Changing Task Status and Assignment
- Using Workload Reports
- Using Statistics Reports

Overview of Workflow Monitoring Tasks

The Studio workflow monitoring features allow workflow designers to perform run-time monitoring to help debug and troubleshoot workflow designs in a design environment, and system administrators to monitor and intervene in the real-time execution of workflows in a production environment.

Additionally, in a production environment, administrators can monitor user and role workloads for manually-assigned tasks.

Finally, business analysts can perform post-run-time data collection on manually assigned tasks to gather and compile historical raw and statistical workload and performance data to determine bottlenecks and inefficiencies in business processes. This section describes the following tasks you can perform:

- Display the status of running or completed workflow instances. Procedures are given in "Viewing Workflow Instance Status" on page 10-5.
- Display and modify the current value of variables in running instances. Procedures are given in "Viewing and Updating Workflow Instance Variables" on page 10-8.
- Modify the tasks associated with a workflow instance, for example, reassigning tasks or forcing work to be redone. Procedures are given in "Changing Task Status and Assignment" on page 10-16.
- Modify the permissions defined for tasks to allow Worklist users or other administrators to modify tasks. Procedures are given in "Changing Task Permissions and Priority" on page 10-14.
- View the task lists for users or roles and modify task assignment, status or permissions. Procedures are given in "Viewing User and Role Worklists" on page 10-12.
- Display the workload status of the system, showing the number of tasks categorized by status (pending, inactive, done, overdue) and user or role. Procedures are given in "Using Workload Reports" on page 10-18.
- Compile raw data and perform statistical reports on workloads and execution times, according to users and roles. Procedures are given in "Using Statistics Reports" on page 10-22.

Note: In order to perform any of the monitoring tasks described in this section, with the exception of viewing user and role worklists, you must have Monitor Instance permission. For more information on permissions, see "Assigning Permissions to Users and Roles" on page 3-26.

Working with Workflow Instances

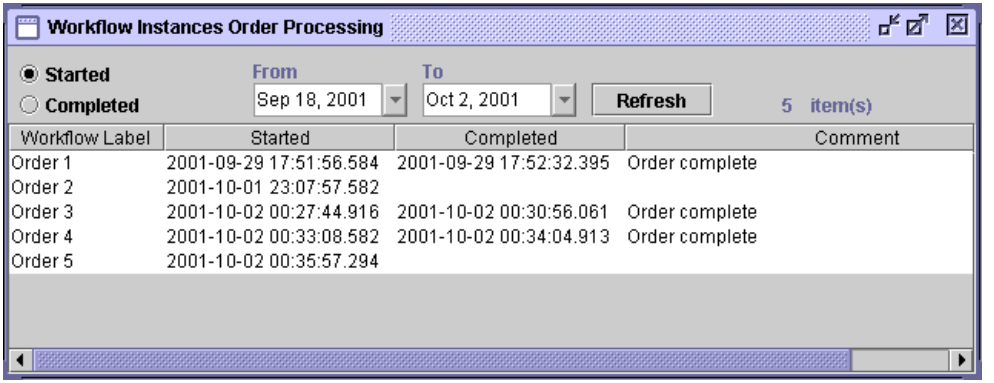
A workflow instance is a session of a workflow template definition that has been placed into run time. You can view the status of workflow instances and the current value of variables. In a design environment, you can view instances to help you debug

and troubleshoot problems with a workflow design. In a production environment, you can view the status of a running workflow and intervene to update variables or modify tasks.

To view a list of workflow instances:

1. In the folder tree, right-click the workflow template or template definition for which you would like to view instances, and from the pop-up menu, select Instances to display the Workflow Instances dialog box.
2. Select one of the following options:
   - Started—select to display all workflows (still in-progress or completed) that were started in a time period you specify.
   - Completed—select to display all workflows that were completed in a time period you specify.
3. From the From and To drop-down boxes, select a start and end date for the period of time for which you would like to view instances for the selected workflow template definition.

Figure 10-1 Workflow Instances Dialog Box



For each workflow instance, the following information is displayed:

Workflow Label	The label generated from the expression specified in the Workflow Label field of the template definition's properties dialog box. For more information, see "Updating, Labeling, and Activating a Template Definition" on page 5-12.
Started	The date the workflow was instantiated.
Completed	The date the workflow was completed. If it is not completed, this column will be blank.
Comment	The comment generated from an expression specified in the Set Workflow Comment action in the workflow. For information, see "Setting Up a Workflow Comment" on page 6-43. If this action was not defined, this column will be blank.

For performance reasons, the process engine only returns 100 items at a time. If there are additional items, a button (+) appears which, when clicked, will retrieve the next 100 items. When no additional items remain, the button is no longer shown.

To update the instances list, click Refresh.

From the Workflow Instances dialog box, you can do the following:

- View the status of the workflow instance in a graphical or list representation. For details, see "Viewing Workflow Instance Status" on page 10-5.
- View and update the current value of workflow instance variables. For details, see "Viewing and Updating Workflow Instance Variables" on page 10-8.
- Modify task assignment and properties. For details, see "Changing Task Permissions and Priority" on page 10-14.
- Delete workflow instances. For details, see "Deleting Workflow Instances" on page 10-11.

Viewing Workflow Instance Status

To view the status of a particular workflow instance, double-click the instance in the list in the Workflow Instances dialog box, or right-click the instance and select Workflow Status from the pop-up menu.

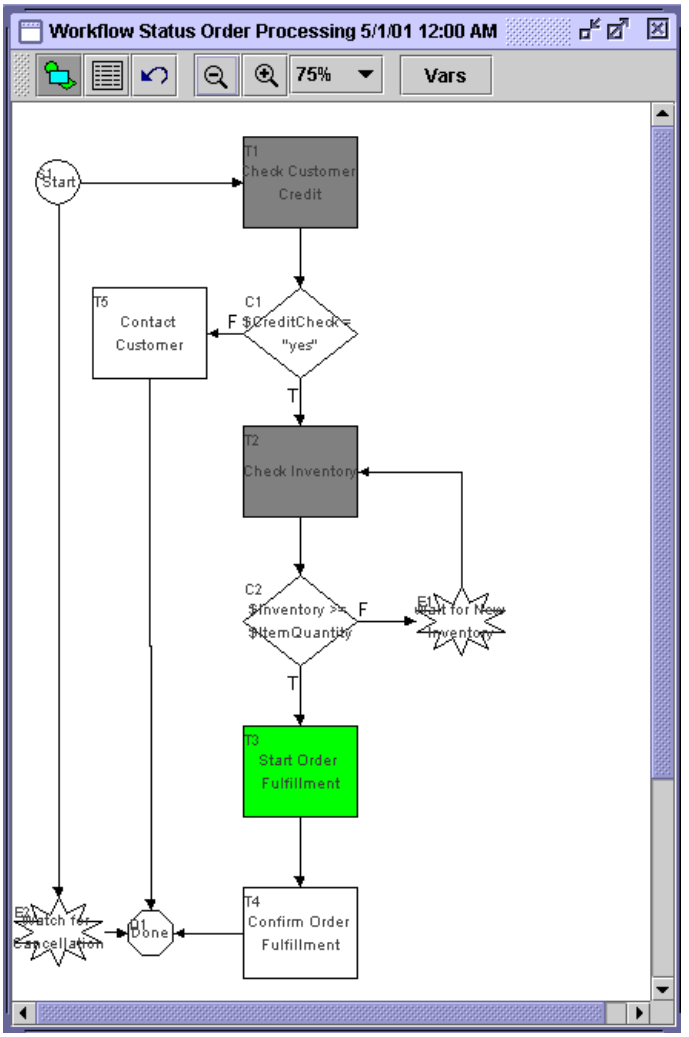
A window is displayed showing either a flowchart or task list representation of the current state of the running workflow. You can use the buttons at the top of the window to do the following:

Table 10-1 Workflow Status Window Buttons

Button	Use to...
	Display a graphical flowchart view of the workflow instance.
	Display a list of all tasks in the workflow instance.
	Refresh the workflow view.

To display a flowchart view of the workflow instance, click the  button.

Figure 10-2 Workflow Status Window: Flowchart View

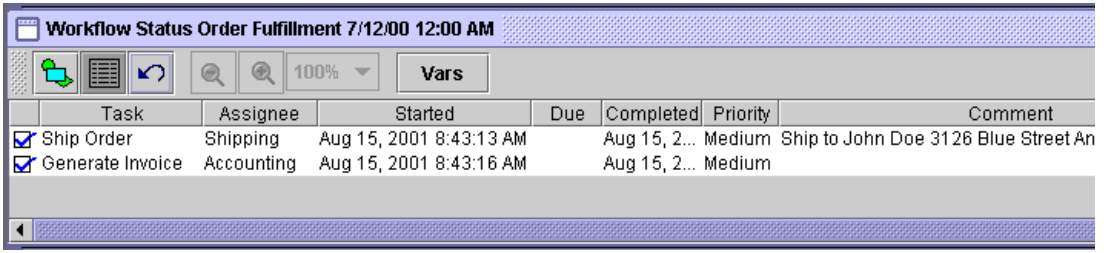


In the graphical representation, active tasks are green, executed tasks are gray, and the inactive tasks or other nodes are white. (For information on task states, see “Understanding Task States” on page 5-54.)

For debugging purposes, you can use this information to identify where a stoppage has occurred in the workflow. If a task that should have completed is still showing as green, that is, as activated but not executed, this is usually an indication that there is an error in the definition of that node.

To display a list of all tasks in the workflow and their status, click the  button.

Figure 10-3 Workflow Status Window: List View



The list view displays the following information for each task:

Task	The name of the task.
Assignee	The user or role to which the task has been assigned. For more information on task assignment, see "Setting Up Manual Tasks" on page 6-44.
Started	The date and time the task was started.
Due	The due date for the task, as specified by Set Task Due Date action in the workflow. For more information, see "Setting a Task Due Date" on page 6-51.
Completed	The date and time the task was completed.
Priority	The priority level assigned to the task. For more information, see "About Task Priority" on page 5-56.
Comment	The comment generated from an expression specified in the Set Task Comment action in the workflow. For information, see "Setting a Task Comment" on page 6-54. If this action was not defined, this column will be blank.

In addition, the following indications appear:

- Tasks shown without a box are non-manual tasks or tasks that have not yet been activated (no started or completed date appears).
- An empty box indicates that the task has been activated but is pending, that is, or waiting to be executed (it shows a started date, but not a completed date).
- A checked box indicates a task that has already been completed (it shows a completed date).
- A red box indicates that the task is now overdue, that is, its due date is before the current date.

From the Workflow Instances dialog box, you can also do the following:

- Change a task's status and assignment. For details, see "Changing Task Permissions and Priority" on page 10-14.
- Change a task's permissions and priority. For details, see "Changing Task Permissions and Priority" on page 10-14.

Viewing and Updating Workflow Instance Variables

You can view and update the current value of all variables defined for the workflow at any point during its execution.

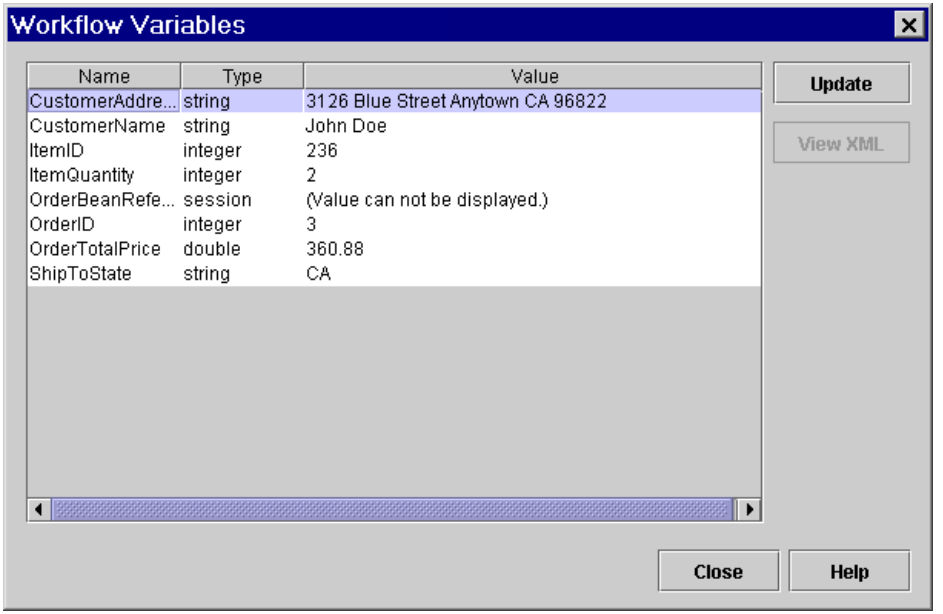
When debugging workflows at design time, it is a good idea to view variable values to help troubleshoot possible design errors. Even if a workflow appears to have executed correctly, you may find incorrect settings of variable values, which may indicate a design bug.

To view workflow instance variables:

1. Do one of the following:
   - From the Workflow Instances dialog box, right-click the desired workflow instance, and from the pop-up menu, select Variables.
   - From the Workflow Status window for an instance, click the Vars button.

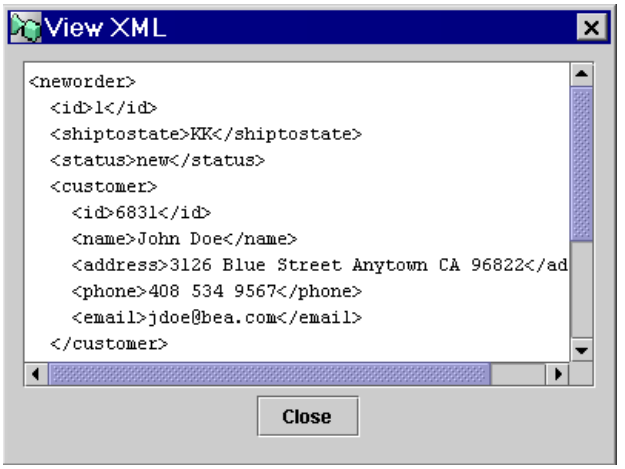
The Workflow Variables dialog box displays each variable in a list, showing its name, type, and its current value. (For information on variable types, see "Working with Variables" on page 5-28.)

Figure 10-4 Workflow Variables Dialog Box



2. To view the content of an XML-type variable only, select an XML variable in the list and click View XML. The View XML window appears with the XML content of the variable displayed.

Figure 10-5 View XML Window

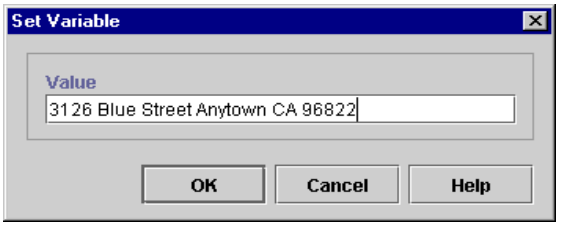


3. Click Close to close the View XML window.

To update a variable's value:

1. From the Workflow Variables dialog box, select the desired variable, and click Update to display the Set Variable dialog box.

Figure 10-6 Set Variable Dialog Box



2. In the Value field, enter a constant that will become the new value from the variable.

Note: Because workflow logic often depends on the value of variables, use caution when manually changing the value of a variable. Also be sure that the value you enter is valid for the date type of the variable.

3. Click OK to save the change and reset the variable.

Deleting Workflow Instances

You can delete a single workflow instance or multiple workflow instances according to date.

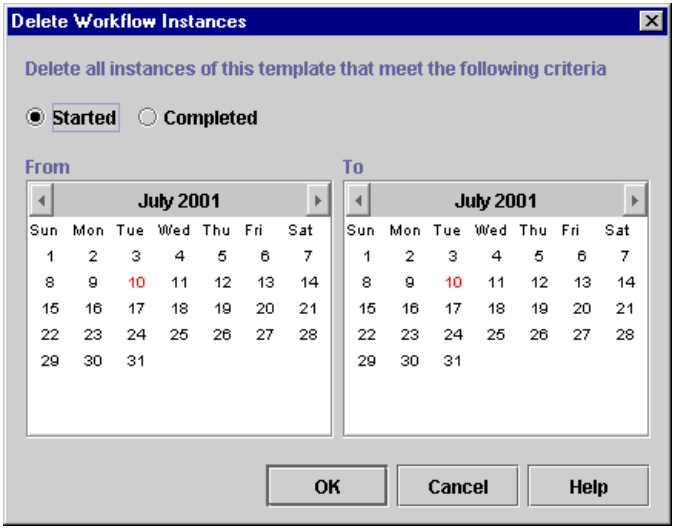
To delete a single workflow instance:

1. Right-click the workflow instance in the list in the Workflow Instances dialog box, and select Delete from the menu.
2. When prompted by the Delete Workflow Instance warning message, click Yes to delete the instance, or No to cancel the delete.

To delete multiple workflow instances for a template:

1. In the folder tree, right-click the template or template definition folder for the workflow instances you want to delete, and from the pop-up menu, select Delete Instances. The Delete Workflow Instances dialog box is displayed.

Figure 10-7 Delete Workflow Instances Dialog Box



2. Select one of the following options:
   - Started — if selected, this option finds all workflow instances—started and completed—with a start date in the specified date range.

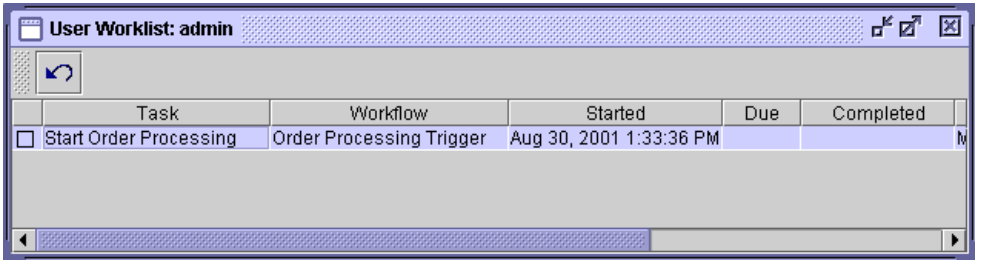
- Completed — if selected, this option finds completed workflow instances with a completed date in the specified date range.

3. In the From and To boxes, select a month and date to specify the period of time for which you would like to delete all the instances for the selected workflow.
4. Click OK to delete the selected workflow instances, or Cancel to cancel the delete.

Viewing User and Role Worklists

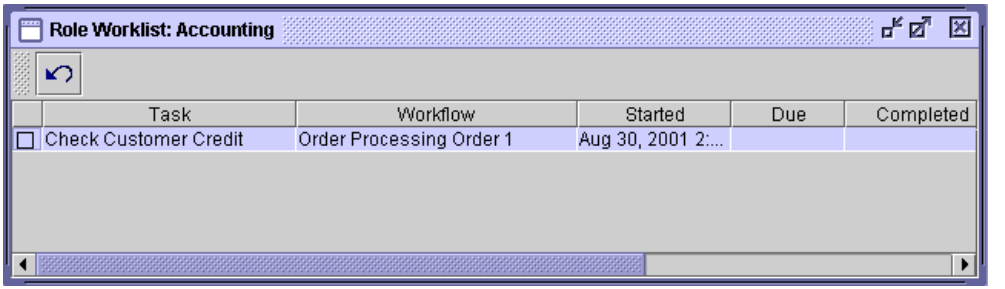
To view a user worklist, in the folder tree for the desired organization, right-click a user from the list and select Open User Worklist from the pop-up menu. A list of tasks assigned to the user is displayed in the User Worklist dialog box.

Figure 10-8 User Worklist Dialog Box



To view a role worklist, in the folder tree for the desired organization, right-click a role from the list and select Open Role Worklist from the pop-up menu. A list of tasks assigned to the role is displayed in the Role Worklist dialog box.

Figure 10-9 Role Worklist Dialog Box



The task list displays the following information for each task:

Task	The name of the task.
Workflow	The name of the workflow in which the task is defined.
Started	The date and time the task was started.
Due	The due date for the task, as specified by Set Task Due Date action in the workflow. For more information, see "Setting a Task Due Date" on page 6-51.
Completed	The date and time the task was completed.
Priority	The priority level assigned to the task. For more information, see "About Task Priority" on page 5-56.
Comment	The comment generated from an expression specified in the Set Task Comment action in the workflow. For information, see "Setting a Task Comment" on page 6-54. If this action was not defined, this column will be blank.

In addition, the following indications appear:

- Tasks shown without a box have not yet been activated (no started or completed date appears).
- An empty box indicates that the task has been activated but is pending, that is, has not yet been executed by the user (it shows a started date, but not a completed date).

- A checked box indicates a task that has already been completed (it shows a completed date).
- A red box indicates that the task is now overdue, that is, its due date is before the current date.

To refresh the task list for a user or role, click the  button.

From the Role Worklist and User Worklist dialog boxes, you can also do the following:

- Change a task's status and assignment. For details, see "Changing Task Permissions and Priority" on page 10-14.
- Change a task's permissions and priority. For details, see "Changing Task Permissions and Priority" on page 10-14.

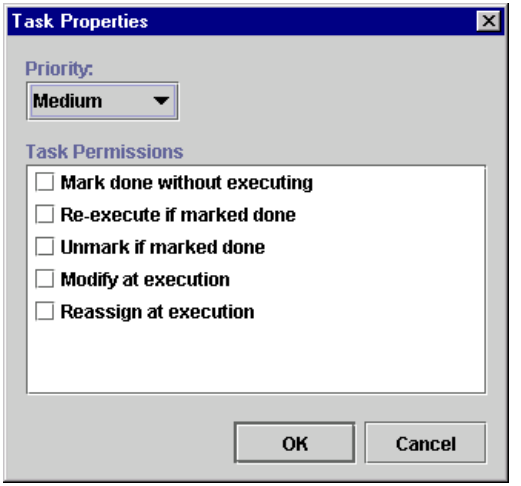
Changing Task Permissions and Priority

If a task has been defined with Modify at Execution permission, you can change the priority and permissions for a task. (For information, see "Defining Task Properties" on page 5-53.) This also enables you or a Worklist user to change the task's status and assignment. For more information, see "Changing Task Status and Assignment" on page 10-16.

To change task permissions and priority:

1. From the Workflow Status, User Worklist or Role Worklist dialog boxes, right-click the task you wish to reassign, and from the pop-up menu, select Properties to display the Task Properties dialog box.

Figure 10-10 Task Properties Dialog Box



2. Optionally, from the Priority field, select a new priority level for the task—Low, Medium, or High.

3. Enable any of the following Task Permissions check boxes:

Permission	Explanation
Mark done without executing	Allows a Worklist user or Studio administrator to mark the task done without it having been executed, and manually set the task's completed date.
Re-execute if marked done	Allows a Worklist user to re-execute a task even if it has already been completed.
Unmark if marked done	Allows a Worklist user or Studio administrator to change the status of the task back to not done when the task is marked as done. (If a user marks a completed task as not done, the task status is set to Active; it does not, however, reverse the effects of any actions that have been executed.)
Modify at execution	Allows a Worklist user or Studio administrator to change the permissions for a task before it has been executed.

Permission	Explanation
Reassign at execution	Allows a Worklist user to take or reassign the task, or a Studio administrator to reassign the task to another user or role before it has been executed.

4. Click OK to save your changes to the task.

Changing Task Status and Assignment

If a task's permissions allow it, you can intervene in a currently running workflow to reassign tasks to different users or roles, and mark and unmark them done.

Note: Although you can reassign tasks, mark or unmark them done, or change their properties, you cannot execute tasks on worklists invoked from within the Studio. To execute a task, you must do so from the Worklist or custom client application. For more information, see *Using the WebLogic Integration Worklist*.

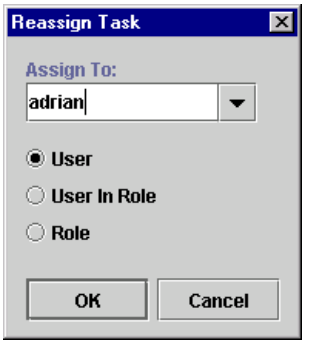
Reassigning a Task

If the task has been defined with Reassign at Execution permission, and the task has not already been marked done, you can reassign a task to a different Worklist or custom client user, role, or user in role. (For an explanation of these distinctions and of task assignment, see "Setting Up Manual Tasks" on page 6-44.)

To reassign a task:

1. From the Workflow Status, User Worklist or Role Worklist dialog boxes, right-click the task you wish to reassign, and from the pop-up menu, select Reassign Task to display the Reassign Task dialog box.

Figure 10-11 Reassign Task Dialog Box



2. Select one of the following options:
   - User—select to reassign the task to a specific user.
   - Role—select to reassign the task to any user in the specified role.
   - User in Role—select to reassign the task to the user in the specified role with the fewest tasks assigned to him or her.
3. In the Assign To drop-down list, select the name of the user or role to which you are reassigning the task.
4. Click OK to reassign the task.

Marking a Task Done

If the task has been defined with Mark Done Without Executing permission, you can mark a task done before it is actually executed by a user.

To mark a task done, from the Workflow Status, User Worklist or Role Worklist dialog boxes, right-click the task you wish to mark done, and from the pop-up menu, select Mark Task as Done. The task's completed date is set to the current date, and the workflow proceeds.

Unmarking a Task Done

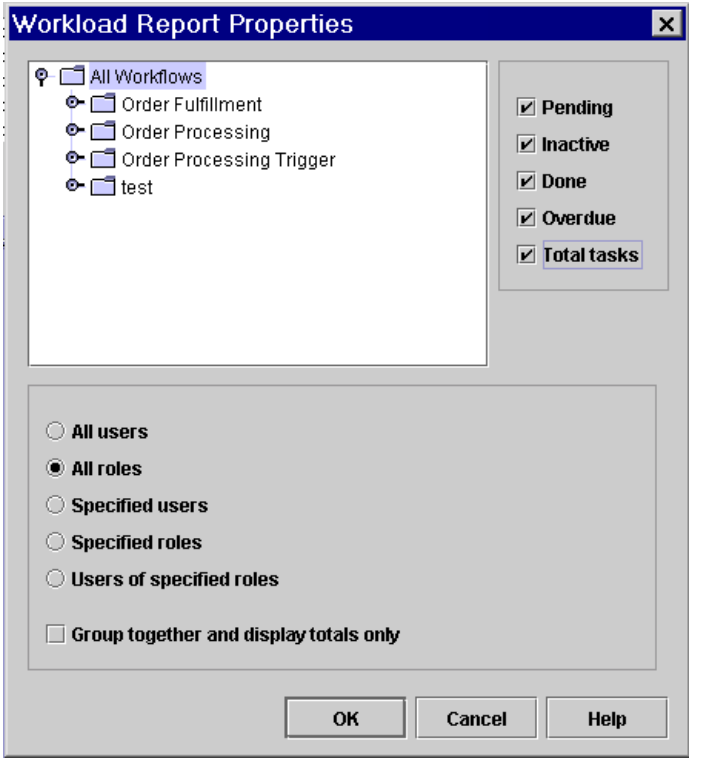
If the task has been defined with Unmark If Marked Done permission, you can unmark a task done that has already been completed.

To unmark a task done, from the Workflow Status, User Worklist or Role Worklist dialog boxes, right-click the task you wish to unmark done, and from the pop-up menu, select Unmark Task as Done. The task's status is reset to pending and the completed date is cleared.

Using Workload Reports

The Studio allows you to view workload reports, showing the number of tasks, by workflow, task, user or role, and task status. In the Studio folder tree, right-click Workload Report, and choose Open from the pop-up menu to display the Workload Report Properties dialog box.

Figure 10-12 Workload Report Properties Dialog Box



Compiling Workload Report Information

The workload report will display information according to the selections you make in the Workload Report Properties dialog box:

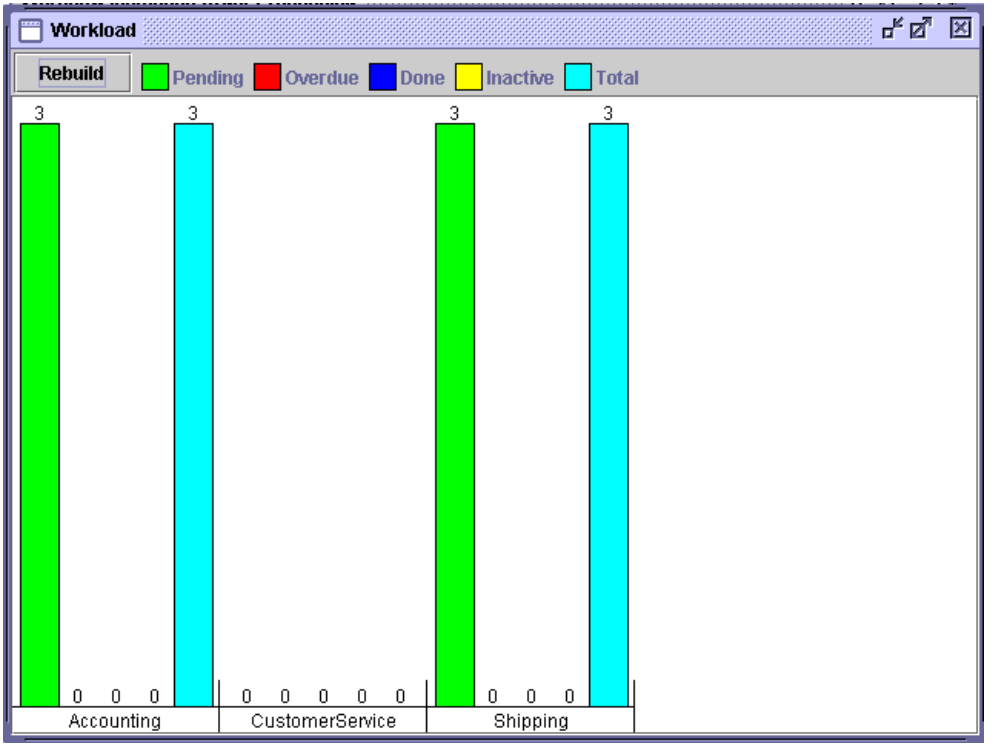
- Workflow Templates folders — select a workflow template to include information pertaining to all workflow template definitions within that workflow template, or expand a workflow template and select only one workflow template definition. Only information within the selected workflow template or workflow template definition will be included in the workload report.
- Pending — displays pending tasks. Pending tasks have been started and not yet completed.

- Inactive — displays inactive tasks. Inactive tasks are those which have not yet been started.
- Done — displays completed tasks.
- Overdue — displays those tasks which are now overdue. Overdue tasks are pending tasks with a due date on or before the current date.
- Total Tasks — displays all tasks, regardless of task state (that is, inactive, pending, and completed tasks are all counted).
- All users — displays workload information for each user.
- All roles — displays workload information for each role.
- Specified users — displays workload information for each of the selected users in the displayed users list.
- Specified roles — displays workload information for each of the selected roles in the displayed roles list.
- Users of specified roles — displays workload information for each user who is a member of one of the selected roles in the displayed roles list.
- Group together and display totals only — displays total counts of tasks only for the group identified by the above selected option. If not selected, each individual user or role is shown in the workload report.

Viewing Workload Reports

After you make your workload report selections, click OK to display a graphical representation of the current workload.

Figure 10-13 Workload Report Dialog Box



The contents of the report are broken down by user, role, or totals, depending on the options selected. This display may scroll left and right if there is a lot of information. A bar chart indicates the total number of tasks of various state counted. The legend at the top of the window shows what each color represents, depending on the options selected at design time. For more precise indicators, the actual number of tasks is shown above each bar.

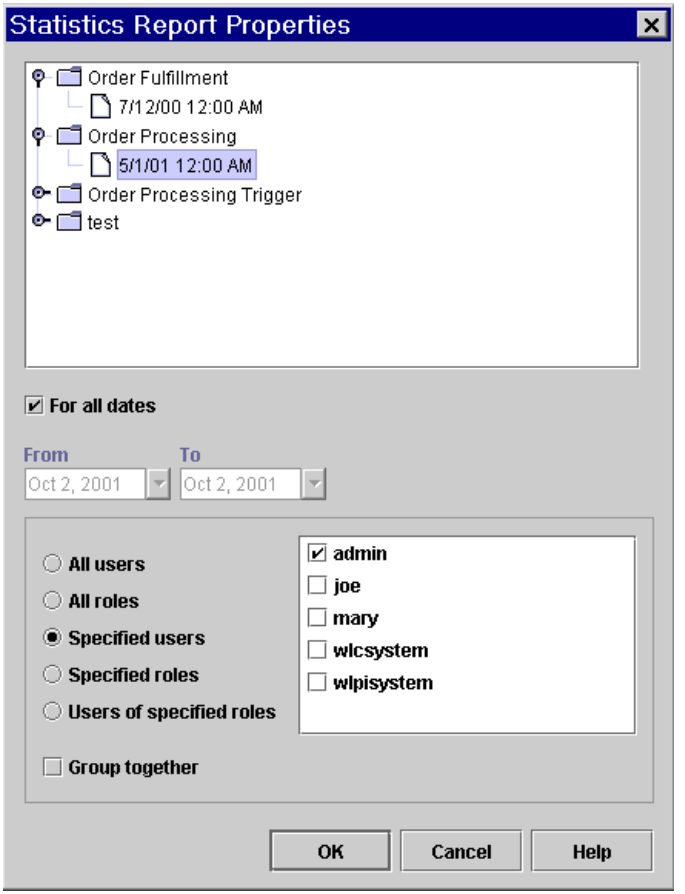
Using Statistics Reports

You can view statistics reports that are based on tasks, users, roles, and so on. A statistics report provides statistical information based on historical data kept for completed workflows. It shows number of tasks completed, total time, average time, and minimum and maximum times spent for each task. Standard deviation from the average time is also shown.

Compiling Statistics Report Information

In the Studio folder tree, right-click Statistics Reports and choose Open from the pop-up menu to display the Statistics Report Properties dialog box. The report includes information according to the selections you make in the Statistics Report Properties dialog box.

Figure 10-14 Statistics Report Properties Dialog Box



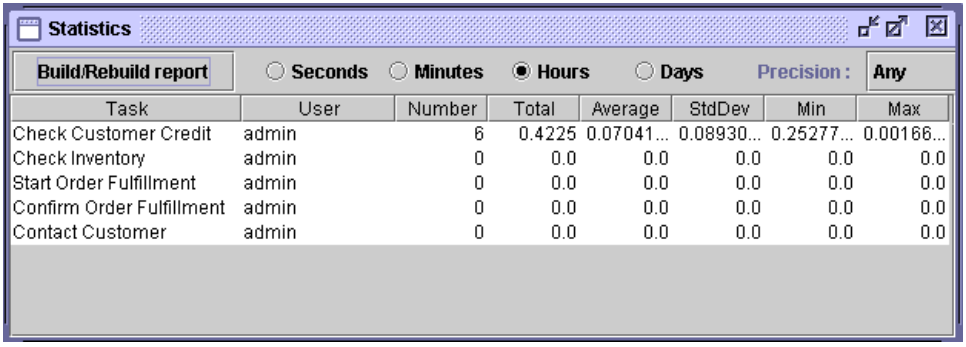
- List of workflow templates — list of available workflow templates. Expand a template to select the workflow template definition to include in the statistics report.
- For all dates — statistical calculations are gathered regardless of date.
- From — all tasks, in completed workflows, completed on or after this date are included in the statistical calculations.
- To — all tasks, in completed workflows, completed on or before this date are included in the statistical calculations.

- All users — displays statistical information for each user.
- All roles — displays statistical information for each role.
- Specified users — displays statistical information for each selected user.
- Specified roles — displays statistical information for each selected role.
- Users of specified roles — displays statistical information for each user who is a member of one of the selected roles in the displayed roles list.
- Group together — the report displays total counts of tasks only for the group identified by the above selected option. If not selected, each individual user or role is shown in the workload report.

Viewing Statistics Reports

When you make your statistics report selections, click OK to display the statistics report.

Figure 10-15 Statistics Report



The top of the dialog box has several options:

- Build/Rebuild — rebuild the report if you change the units of time or the precision level.
- Seconds, Minutes, Hours, Days — units of time used in the report.

- Precision — drop-down list from which you can select how many decimal places you want in the statistical information.

Several columns are displayed in the report, depending on the options selected at report design time:

- Task — name of the task being measured.
- User — user or role assigned to the task. If the Group together option was selected at report design time, this column is not shown, because the results are totaled for all users or roles.
- Number — total number of tasks completed.
- Total — total amount of time spent on doing the task. This is the time interval between the task start date and completed date.
- Average — average amount of time spent performing this task.
- StdDev — standard deviation from the average amount of time spent performing this task.
- Min — minimum amount of time spent performing this task.
- Max — maximum amount of time spent performing this task.

Because the report definition is stored separately from the results, the statistics report can be run at any time.

11 Importing and Exporting Workflow Packages

The following sections show you how to import and export workflow packages from and to Java archive files and how to import and export template definitions from and to XML files:

- About Import/Export
- Exporting Workflow Packages
- Importing Workflow Packages
- Importing and Exporting Workflow Template Definitions from and to XML Files

About Import/Export

WebLogic Integration provides the capability of exporting and importing workflow objects to and from Java Archive (JAR) files. This capability allows you to export and import templates, template definitions, business operations, business calendars, event key tables, and XML repository items. Organizations, users, and roles cannot be exported or imported.

You can mark exported packages as *published* packages, which marks their contents as read-only when they are imported. In published templates, new template definitions cannot be created. However, published templates and template definitions may be deleted.

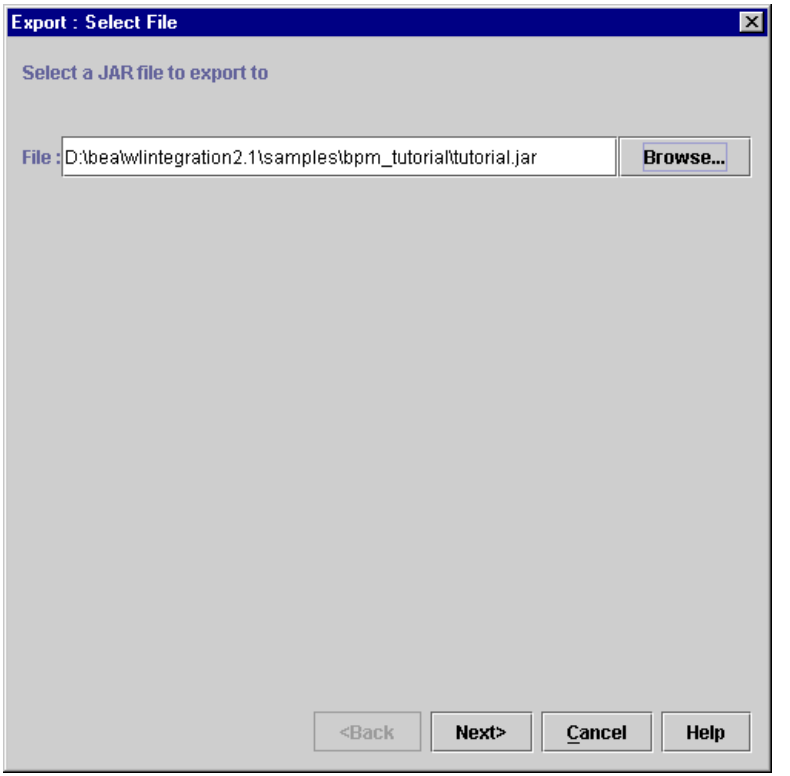
You can also assign password protection to exported packages to prevent them from unauthorized import.

Exporting Workflow Packages

To export a workflow package:

1. Choose Tools—Export Package to display the Export: Select File dialog box.

Figure 11-1 Export: Select File Dialog Box



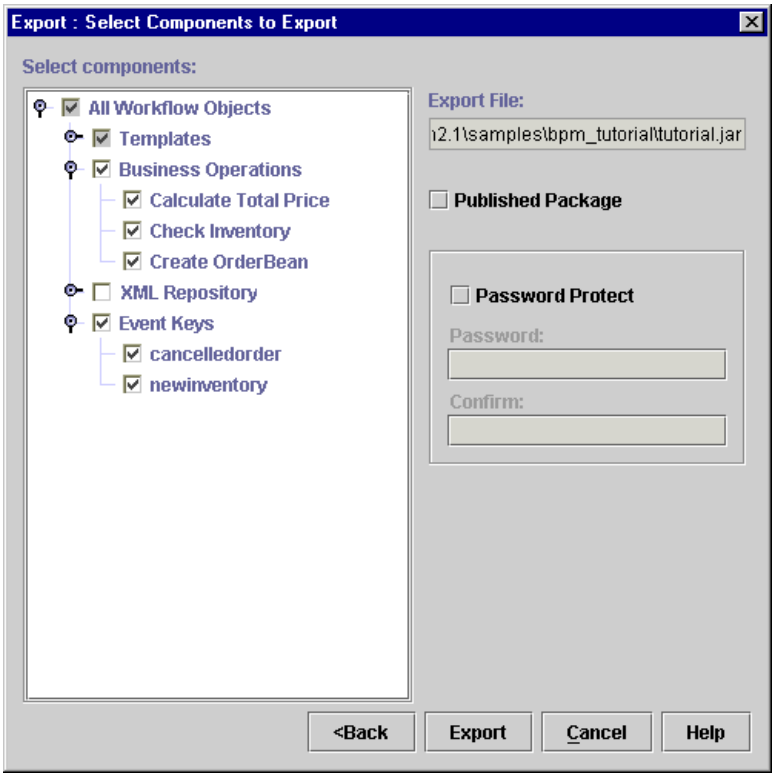
2. Specify the full pathname of the JAR file to which you want to export the package, by doing one of the following:
   - Type the path and filename directly in the File field.
   - Click Browse to display the Save dialog box, select a folder from the Look In field, enter a name for the file in the File name field, and click Save to supply the path and filename.
3. Click Next.

   If you specified an existing JAR file that is not password protected, the Export: Select Components to Export dialog box is displayed.

   If you specified an existing JAR file that is password protected, the Export: Enter Password dialog box is displayed. Enter the necessary password and click

Next to display to display the Export: Select Components to Export dialog box, or click Back to specify an alternate file.

Figure 11-2 Export: Select Components to Export Dialog Box



Note: If, in the Export: Select File dialog box, you selected an existing JAR file on your system in which to export the package, the items in the tree are prechecked, based on the contents of the selected JAR file.

4. Specify the components to be exported by selecting or deselecting the appropriate check boxes. The export feature does not enforce template definition integrity so you can deselect any object to exclude it from the exported package.

   Note: If you select a template definition to be exported, any sub-workflows or business operations referenced by the template definition are automatically selected. However, business calendars and XML entities are not

automatically selected. Therefore, you must manually select these items if you want them to be included in the exported package.

If you export a template definition that specifies a due date for a task, and the due date is set using a business calendar, be sure to export the calendar along with the template definition.

Similarly, if you export a template definition that uses the XSL Transform action, be sure to export the XML entities in the repository required by the XSL Transform action. The XML entities could be the XSLT template document or the XML input document.

5. Optionally, select the Published Package check box to make the objects of a published package read-only after they are imported to the target system.
6. Optionally, select the Password Protect check box to assign a password to the package, and enter the password in the Password and Confirm fields. The user on the target system must specify this password before they can import the package.
7. Click Export to begin the export. After the export operation is complete, a Review Export Summary report is displayed indicating if the export operation was successful.
8. Click Close to close the Review Export Summary dialog box.

Importing Workflow Packages

If you are importing a template definition from a published package (that is, a package that was previously exported and marked as published), and a template does not exist on the target system, a published template is automatically created. If, however, you are importing a published template definition into an existing template, the existing template must also be published. You cannot import a published template definition into an unpublished template.

Similarly, if you are importing a template definition from an unpublished package and a template does not exist on the target system, an unpublished template is automatically created. To import an unpublished template definition into an existing template, the existing template must also be unpublished. You cannot import an unpublished template definition into an published template.

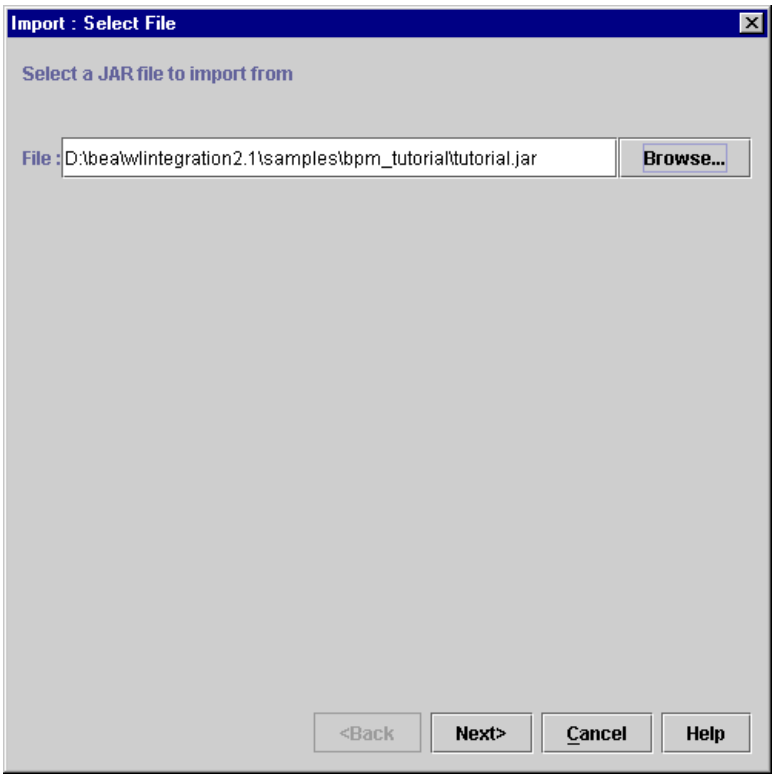
During the import process, a warning is displayed if you are attempting to import an object with the same name as an object that already exists on your system. You can overwrite templates, business operations, and business calendars. You are not permitted to overwrite template definitions, so multiple copies will be created instead.

After import, you can associate templates with additional organizations by using the Template Properties dialog box and checking additional organizations (see "Updating Template Properties" on page 5-6 for procedures). Do not re-import the same template into additional organizations, as this will simply create multiple template definitions.

To import a workflow package:

1. Choose Tools→Import Package to display the Import: Select File dialog box.

Figure 11-3 Import: Select File Dialog Box

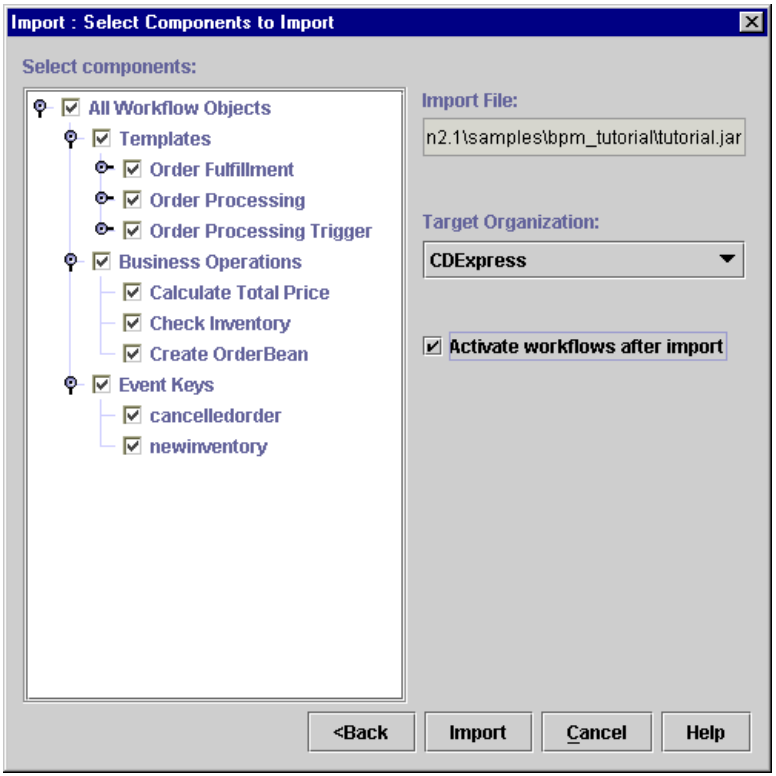


2. Specify the full pathname of the JAR file from which you want to import the package, by doing one of the following:
   - Type the path and filename directly in the File field.
   - Click Browse to display the Open dialog box, select a folder from the Look In field, enter a name for the file in the File name field, and click Open to supply the path and filename.
3. Click Next.

   If the package you want to import is not password-protected, the Import: Select Components to Import dialog box is displayed.

   If the package you want to import is password-protected, the Import: Enter Password dialog box is displayed. Enter the necessary password and click Next to display the Import: Select Components to Import dialog box.

Figure 11-4 Import: Select Components to Import Dialog Box



4. Clear the check box for any object that you do not want to import.

   Note: If you import a template definition that specifies a due date for a task, and the due date is set using a business calendar, be sure to import the calendar along with the template definition. Similarly, if you import a template definition that uses the XSL Transform action, be sure to import the XML entities in the repository required by the XSL Transform action. The XML entities could be the XSLT template document or the XML input document.

5. From the Target Organization drop-down menu, select the organization to which the imported template definition should be initially assigned.

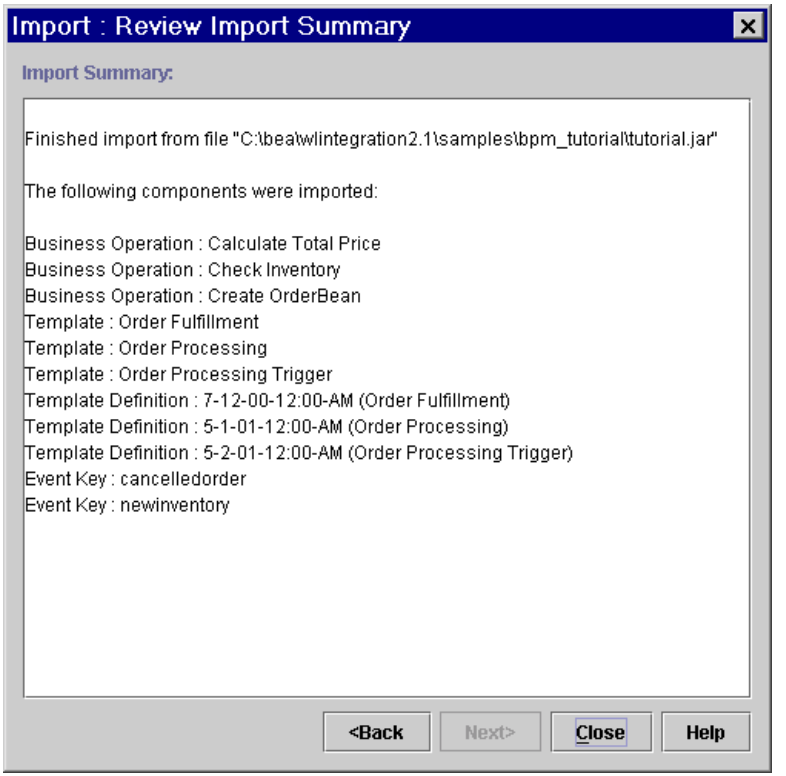
6. Optionally, select the Activate after import check box if you want the imported template definitions to be activated automatically after they are imported. For information on activation, see "Updating, Labeling, and Activating a Template Definition" on page 5-12.

   Note: If a template definition has unresolved references to objects, it cannot be activated.

7. Click Import to begin the import operation, or Cancel to cancel the operation.

   After the import operation is complete, a Review Import Summary report is displayed. The report lists all the objects that were imported and any problems that were encountered during the import operation.

Figure 11-5 Review Import Summary Window



8. Click Close to close the Review Import Summary dialog box, or Back to return to the previous screen.

Importing and Exporting Workflow Template Definitions from and to XML Files

For compatibility with earlier versions of WebLogic Process Integrator, you can export a workflow template definition to an XML file on any drive mapped on your computer, and import a workflow template definition (that was previously exported), in XML format, into the Studio to create a new workflow template definition.

Note: If you do not need to maintain compatibility with older versions of WebLogic Process Integrator, we recommend that you take advantage of the Import/Export Package facility described earlier in this section.

Exporting Workflow Template Definitions to XML

To export a workflow template definition:

1. In the folder tree, right-click the template definition to be exported.
2. Choose Export from the pop-up menu.The Save dialog box is displayed.
3. Select the drive and the directory in which you want to save the exported workflow template definition.
4. Optionally, select or enter a name for the file in the File name field. The default name is the template name.
5. Click Save to export the template definition, or Cancel to cancel the operation.

Importing Workflow Template Definitions from XML

Workflow template definitions imported from an XML file are always marked as *inactive*. Before an imported workflow template definition can be instantiated, you must change the definition to *active* in the Template Definition dialog box. For details, see "Updating, Labeling, and Activating a Template Definition" on page 5-12.

If a template does not exist, you must create one before you can import a template definition. For details about creating a template, see "Creating a Workflow Template" on page 5-4.

Note: In some cases, importing workflows may display warning messages. You must define any business operations or event keys that are referenced by imported workflows, and redefine Perform Business Operation or Start Workflow actions contained in imported workflows.

To import a workflow template definition, proceed as follows:

1. In the Studio folder tree, right-click the workflow template into which you will import the workflow template definition.

   Note: If the template into which you are importing the workflow is named differently from the name of the XML file you are importing, a warning message is displayed.

2. From the pop-up menu, select Import Template Definition to display the Open dialog box.

3. In the Open dialog box, select the current location of the XML file on your hard drive, and click Open.

   Note: You cannot import a workflow template definition using a name that is different from the one that it was assigned when it was exported. If you attempt to do so, you will receive a warning message.

4. After the file is imported, an import confirmation dialog box is displayed that asks you to confirm the import. Click Yes to import the workflow template definition. The template definition is created with the dates and times that were originally specified

Index

D

E

F

U

V

W

X