BEA WebLogic Integration™

Tutorial: Building a Worklist Application

Version 8.1
July 2003
Part Number:

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

About This Document

This document explains how to build an application that can be used to orchestrate the integration of people in a business process. For an overview of the scenario used in this tutorial and an introduction to what you learn by completing the steps in the tutorial, see "Tutorial: Building a Worklist Application" on page 1-1.

What You Need to Know

This document is intended for anyone who wants to learn how to integrate people in business processes using the Worklist system.

This document assumes a familiarity with building business processes in WebLogic Workshop. To do so, you can complete the following tutorials in the WebLogic Workshop online help system:

- Tutorial: Building Your First Business Process
- Tutorial: Building Your First Data Transformation

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on 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/.

Related Information

The following WebLogic Integration documents contain information that is relevant to building WOrklist solutions:

- Using the Worklist System
- Worklist Administration in Managing WebLogic Integration Solutions
- Business Calendar Configuration in Managing WebLogic Integration Solutions

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 version of the WebLogic Integration product and 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 **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
boldface text	Indicates items that are displayed on the User Interface.
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 file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`

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

Tutorial: Building a Worklist Application

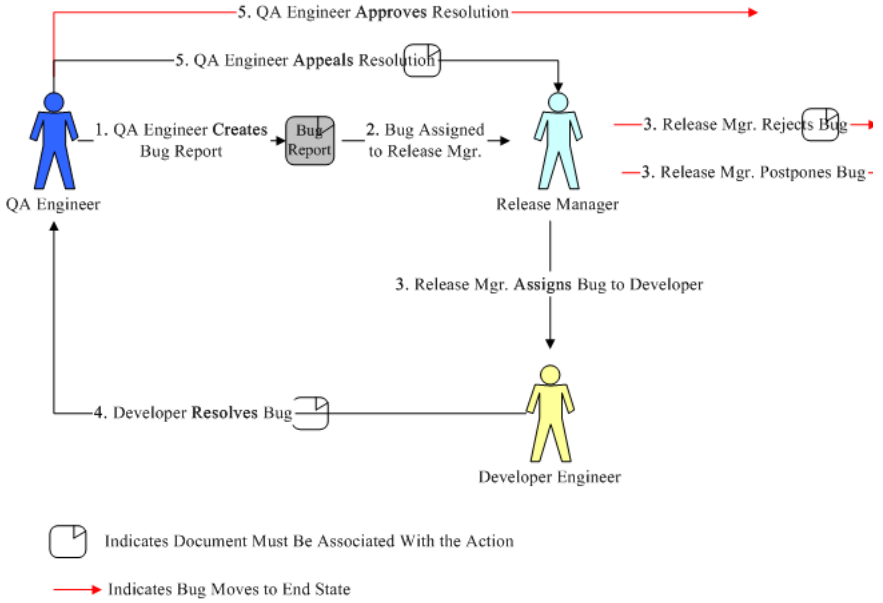
WebLogic Integration's business process management (BPM) functionality enables the integration of diverse applications and human participants. The WebLogic Integration Worklist system enables people to collaborate in business processes—including assigning tasks, tracking the status of tasks, handling approvals, and so on.

This tutorial provides a tour of the features available to design the interaction of people with business processes in the WebLogic Workshop graphical design environment. It describes how to create a business process that uses Worklist controls to orchestrate the resolution of a bug in a software company's bug tracking system.

Tutorial Overview

The tutorial scenario is based on a portion of the life cycle of a software bug at a fictitious software company named SoftCo.

The following figure and sequence of events outlines the path of a software bug through the SoftCo organization and the actors in the scenario:



1. A Quality Assurance engineer creates a bug report. To do so, they specify the following information in an online form: title of bug, description, a priority (1,2, or 3), and the steps to reproduce the bug.
2. New bugs are assigned to a release manager with the fewest bugs in their queue. SoftCo policy states that the release manager must act on the bug report within two business days.
3. When a release manager receives a bug in their queue, they can reject it, pass it on to the development engineers, or postpone to the next release:
   - **Reject**—If a release manager rejects a bug, they must provide documentation that records the reason for the rejection.
   - **Postpone**—Bugs that are postponed to a future release are reinstated to the release manager's queue when the next release cycle begins.
   - **Assign**—When a bug is assigned to the development engineers group, any development engineer can review the bug and claim it if it is appropriate to do so.

4. Development engineers can resolve a bug as rejected or fixed:
   - **Reject**—Development engineers can reject the bug if it can not be reproduced or if for some other reason it will not be fixed. They must provide documentation that records the reason for the rejection.
   - **Fix**—Development engineers can fix the bug. They must record the details of the fix in the bug report.
5. When a bug is resolved as rejected or fixed, the Quality Assurance engineer who created the bug must either approve the resolution or appeal it. They must take action on the bug resolution within two business days.
   - **Appeal**—When a QA engineer appeals the resolution of a bug, they must provide documentation that describes the reason or reasons for the appeal. A bug that is appealed is routed back to the release manager who was originally assigned to review the bug. Bug resolutions that are not appealed or approved within two days are assumed approved.
   - **Approve**—When a QA engineer approves the resolution of a given bug, no further action is required for the bug.
6. Release managers can reject appeals, in which case the bug becomes permanently resolved.

Note: To keep the team apprised of due dates, tasks that become overdue cause emails to be sent. If a bug is not claimed by a developer engineer, the release manager receives an email every two business days. If a bug is claimed by a developer engineer, but is not resolved within four business days, the developer engineer receives an email every two business days.

Values That Describe the Status of Software Bugs Include

```
NEW, NOT_REPRO, WILL_NOT_FIX, FIXED, POSTPONED, APPEALED, APPEAL_REJECTED
```

SoftCo Business Hours

SoftCo engineers work Monday through Friday 10AM-8PM, and on Saturdays from 10AM-2PM.

SoftCo Users and Groups

The following groups compose the team at SoftCo company: Quality Engineers, Release Managers, Development Engineers. Each worker is a member of one of these groups.

Steps in This Tutorial

The steps in this tutorial describe how to create a business process that uses Worklist controls to orchestrate the part of the bug tracking process that starts with the bug being resolved. That is steps 4 and 5 of the life cycle, as described in the preceding section.

In this scenario, a document that describes the resolution of the bug is created by the developer engineer—the business process you create by following the steps in this tutorial starts when it receives this document. Subsequently, the task of accepting or rejecting the resolution of the bug must be assigned to the QA engineer who created the bug in the first place. The QA engineer either approves or appeals the resolution of the bug, the business process sends a message to the client indicating whether the resolution is approved or appealed, and the business process terminates.

The tutorial includes the following steps:

Step 1. Set Up Your Environment

Learn how to set up the actors in the tutorial, that is the users and groups required to complete the tasks you design in the Resolution Approval business process you create. You also learn how to create a business calendar for your SoftCo enterprise

Step 2. Create Your Application

Learn how to create a WebLogic Workshop application that holds the files you create as you work through this tutorial. Specifically, you create the starting application, and add the XML Schema files you need as you build the bug tracking application.

Step 3. Design How to Start the Business Process

In this step, you begin the design of the Resolution Approval business process. Specifically, you design how the business process is started at run time.

Step 4. Create Task and Assign to User

Learn how to build the part of the business process that orchestrates the assignment of the task of approving the resolution of a bug in the SoftCo bug tracking system. Learn to create and use Task controls in the business process.

Step 5. Receive Resolution Approval From the Task Owner

Learn how to design the business process to handle the possible events returned by the Task control.

Step 6. Run the Resolution Approval Business Process

Run and test the functionality of the business process you created. You use the WebLogic Workshop's browser-based interface to start the business process (playing the role of the

client to the business process), and you use the Worklist user interface to play the role of a quality engineer assigned to the task of approving the resolution of the bug.

Step 1. Set Up Your Environment

In this step, you set up the actors in the tutorial, that is the users and groups required to complete the tasks you design in the tutorial business process. Specifically, if you have not already done so, you create an integration domain for your worklist application, then use the WebLogic Integration Administration Console to configure users and groups, and create a business calendar for your SoftCo enterprise.

Complete the following tasks in Step 1:

- Before You Begin
- Create a New WebLogic Integration Domain
- Configure the Users and Groups
- Create a Business Calendar

Before You Begin

Before you begin the Worklist tutorial you must have WebLogic Platform 8.1 installed on your system. For more information, see *Installing WebLogic Platform*.

Create a New WebLogic Integration Domain

The Worklist tutorial requires a WebLogic Integration domain. If you have not already done so, create a WebLogic Integration domain using the WebLogic Platform Configuration Wizard, as described in the following steps.

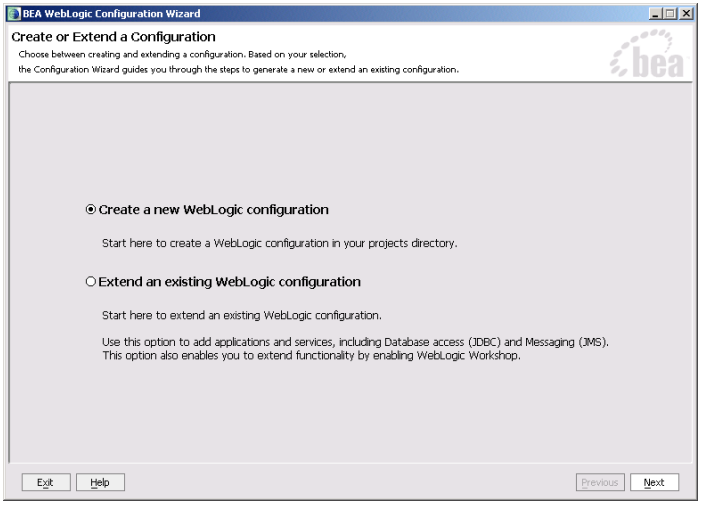
Note: If you already created an integration domain and you want to use it for the tutorial application, you can skip the procedure to create a new domain and instead start WebLogic Server in the existing domain.

The domain name used in this document is `worktutorial`, but you can use any valid domain name.

To Create a New WebLogic Integration Domain

1. Start the Configuration Wizard:
   - **Windows**: From the Start menu, choose **Programs→BEA WebLogic Platform→Configuration Wizard**.
   - **Unix**: Open a command shell, change to the `/common/bin` subdirectory of the product installation directory (such as `/bea/weblogic81/common/bin`), and enter the following command: `sh config.sh`.

   The Configuration Wizard starts and displays the initial screen.



2. In the Configuration Wizard, make the following selections in each screen, and then click **Next** to continue.

Table 2-1 Selections in the Configuration Wizard

Screen Name	Recommended Selection
Create or Extend a Configuration	Create a new WebLogic configuration (default)
Select a Configuration Template	Basic WebLogic Integration Domain (from the list of WebLogic configuration templates).
Choose Express or Custom Configuration	Express (default)
Configure Administrative Username and Password	Specify the administrator password (required) and change defaults as needed.
Configure Server Start Mode and Java SDK	Startup Mode: Development Mode (default) Java SDK Selection: Sun SDK (default)
Create WebLogic Configuration	In the **Configuration Name** field, specify `worktutorial` or another name that you want to use for this domain. Click **Create**.

3. Click **Create**.

   By default, the Configuration Wizard creates the new domain in the following location: *`BEA_HOME`*`\user_projects\domains\`*`domainName`*, where:

   – *`BEA_HOME`* is the directory in which you installed WebLogic platform (for example, `c:\bea`).

   – *`domainName`* is the name of the domain you created (`worktutorial`, in this case).

4. On the **Creating Configuration** screen, select **Start Admin Server**.

5. Click **Done**.

   WebLogic Server starts in the `worktutorial` domain.

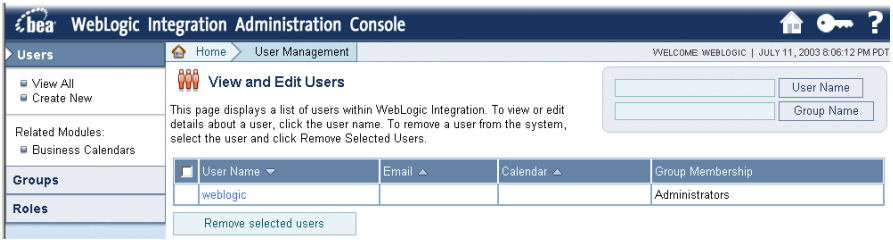
Configure the Users and Groups

You use the WebLogic Integration Administration Console to configure the following groups in your SoftCo enterprise: Quality Engineers, Release Managers, Development Engineers.

1. Start the WebLogic Integration Administration Console by doing either of the following:
   - From the Start menu, choose **Programs→BEA WebLogic Platform→Examples→WebLogic Integration→Integration Admin Console**.
   - In a Web browser, enter the following URL: `http://localhost:7001/wliconsole`.
2. When prompted, enter the username and password you specified for your domain.

   The WebLogic Integration Administration Console displays the home page.
3. Select the **User Management** module:

   The User Management screen is displayed:



4. Click **Groups** to open the screen that allows you to create new groups.
5. Click **Create New** to create a new group:
   a. In the **Group Name** field, enter **QualityEngineers.**
   b. In the **Group Membership** field, select **IntegrationUsers** from the list of **Available Groups** and use the arrow to move the **IntegrationUsers** group into the **Current Groups** listing.
6. Repeat Step 5 two more times to create two more groups—name the groups **ReleaseManagers**, and **DevelopmentEngineers.**

7. The groups you create are added to the list of Groups on the User Management screen:

Group Name	Calendar	Description	Group Membership
QualityEngineers		SoftCo QA Engineers	IntegrationUsers
ReleaseManagers		SoftCo Release Managers	IntegrationUsers
DevelopmentEngineers		SoftCo Development Engineers	IntegrationUsers

8. Click **Users** in the User Management screen to open the screen that allows you to create new users in each of the groups you created.

9. Click **Create New** to open the **Add New User** screen. Create one user in each group. The following table describes how to specify the users for the tutorial scenario:

User Name	Email	Password	Group Membership
QualityEngineerA	You can specify an email address for each user, but for the sake of simplicity in the tutorial scenario, you can specify the same email address for each—an address that will work for your testing purposes.	Specify a password for each user.	QualityEngineers IntegrationUsers
QualityEngineerB			QualityEngineers IntegrationUsers
ReleaseManagerA			ReleaseManagers IntegrationUsers
ReleaseManagerB			ReleaseManagers IntegrationUsers
DevEngineerA			DevelopmentEngineers IntegrationUsers
DevEngineerB			DevelopmentEngineers IntegrationUsers

Create a Business Calendar

You must create a Business Calendar to be used by the SoftCo enterprise during the tutorial scenario. A Business Calendar specifies the dates and times an enterprise is *open for business*.

1. Navigate to the **Business Calendar Configuration** module in the WebLogic Integration Administration Console.
2. Click **Create New** to open the **Create Business Calendar** screen.
3. In the **Business Calendar Name** field, enter **SoftwareTeamCalendar.**
4. Click **Create**. The calendar is created and listed on the **Business Calendar Management** screen.
5. Click **SoftwareTeamCalendar** in the list to open a screen which displays the details about your business calendar, and allows you to specify the time period rules.

   The default rules in the business calendar specify that business hours for employees at SoftCo include Monday to Friday, 8AM to 5PM. You must change the default rules to create the appropriate calendar for the SoftCo enterprise. You can change the business rules for the calendar by editing the **Time Period Rules** table. To do so:

   a. For each of the days listed in the default calendar, click the Time Period to open an update page.

   b. Change the Start Hour and End Hour to **10** and **20**, respectively.

      (Recall that SoftCo engineers work Monday through Friday 10AM-8PM, and on Saturdays from 10AM-2PM.)

   c. Click **Submit**. The business hours for Monday to Friday are updated in the **Time Period Rules** table.

   d. To add a rule for Saturday business hours, click **Add a New Rule**. A page is displayed in which you can specify the time period rules for Saturday. Select the following specifications in the appropriate fields:

      Day of Week: **Sat**
      Start Hour and Minute: 10 00
      End Hour and Minute: 14 00
      Free or Busy: Free

      Click **Submit**. The business hours for Saturday are updated in the **Time Period Rules** table.

The **Time Period Rules** table is shown in the following figure:

Business Calendar Name: SoftwareTeamCalendar
Time Zone: America/Los_Angeles
Is a system calendar false

Edit Calendar Details.

Time Period Rules:

	Time Periods	Free or Busy
☐	Mon, 10:00AM - 8:00PM	Free
☐	Tue, 10:00AM - 8:00PM	Free
☐	Wed, 10:00AM - 8:00PM	Free
☐	Thu, 10:00AM - 8:00PM	Free
☐	Fri, 10:00AM - 8:00PM	Free
☐	Sat, 10:00AM - 2:00PM	Free

Note: To learn more about Business Calendars and defining the time period rules, access the Business Calendar Configuration topic in the administration console online help.

Step 2. Create Your Application

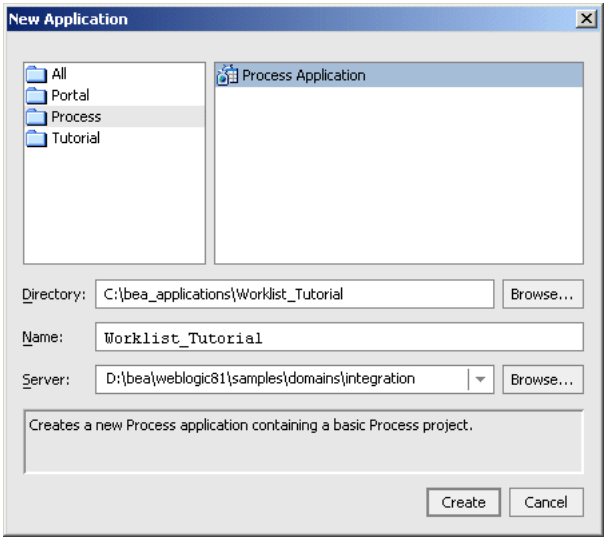
In this step, you create a WebLogic Workshop application that holds the files you create as you work through this tutorial. In this step you create the starting application, and add the XML Schema files you need as you build the bug tracking application. In subsequent steps, you build the business process that orchestrates the resolution of a bug in the SoftCo bug tracking system. The tasks in this step include:

- Create a WebLogic Workshop Application
- Create the Tutorial Schemas

Create a WebLogic Workshop Application

Complete the following steps to create a WebLogic Workshop application in which to build the Bug Resolution business process:

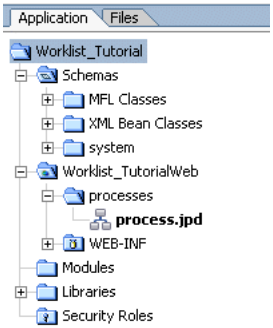
1. From the WebLogic Workshop menu, click **File→New→Application**. The **New Application** dialog box is displayed.



2. In the **New Application** dialog box, select **Process** in the left pane, and select **Process Application** in the right pane.
3. In the **Directory** field, select the directory in which you want to create your application. The paths used in the remainder of this tutorial assumes that the application is created in `C:\bea_applications\`. If you create the directory in a different location or use another name, adjust the instructions accordingly.
4. In the **Name** field, enter `Worklist_Tutorial`.
5. Click the arrow beside the **Server** field to display a list of servers. Then choose an integration domain. For example, if you created a new integration domain in the default location, the path to the integration server is:

   BEA_HOME`\user_projects\domains\worktutorial`

   where *BEA_HOME* is the directory in which you installed WebLogic Platform.
6. Click **Create**.
7. Your Worklist Tutorial Application is created and displayed in the **Application** pane. If the **Application** pane is not visible in WebLogic Workshop, choose **View→Application** from the menu bar.



The **Application** pane displays a hierarchical representation of the files and resources available in your application. The components we work with in this tutorial include the following:

Worklist_Tutorial—The application folder.

Schemas—A Schemas project that contains the system XML Schemas used in the application.

Worklist_TutorialWeb—A Web application project folder. Every application contains one or more projects. Projects represent WebLogic Server Web applications. In other words, when you create a project, you are creating a Web application. (The name of your project is included in the URL your clients use to access your application.)

Web Applications are J2EE deployment units that define a collection of Web resources such as business processes, Web services, JSPs, servlets, HTML pages, and can define references to external resources such as EJBs.

Note: The Web application project folder is named by appending **Web** to the name you gave your application.

processes—The default folder created when you create a new application—your project files are located in this folder. By default, one file is created in a new Process application: **process.jpd**.

8. Rename the **processes** folder and the **process.jpd** file.

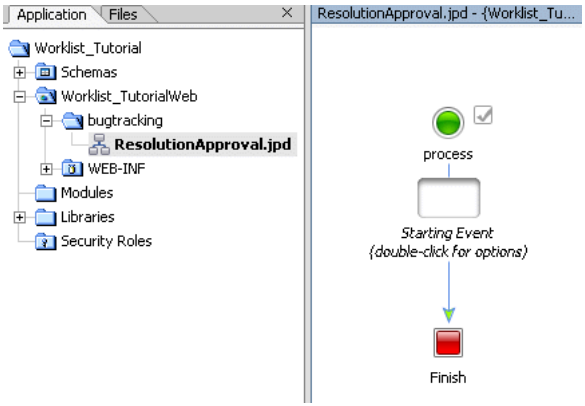
   a. Right-click the **processes** folder in the **Application** tab and select **Rename** from the drop-down menu.

   b. Enter **bugtracking** as the new name, and press **Enter**.

c. Right-click **process.jpd** in the **Application** tab and select **Rename** from the drop-down menu.

d. Enter **ResolutionApproval.jpd** as the new name, and press **Enter.**

The **Application** tab and the **Design View** now resemble that shown in the following figure:



Create the Tutorial Schemas

In this tutorial, XML schemas are used to validate the XML data that is exchanged between the Worklist and the Bug Resolution business process. This section describes how to create the schemas in the Schemas project in your Worklist Tutorial application.

To Create the Tutorial Schemas

1. In the **Application** tab, right-click on a **Schemas** folder.
2. From the drop-down menu, select **New→XML Schema**.

   The **New File** dialog box is displayed.
3. Enter **BugResolution.xsd** as the name of your schema file, and click **Create**. The **BugResolution.xsd** file is created and displayed in the **Application** pane.
4. Double click **BugResolution.xsd** in the **Application** pane to open the file in the **Design View**.

Note: In this tutorial scenario, XML Schema data is provided in Appendix A. Test XML. You can open the appendix in a Web browser and copy the XML schemas to create the schemas required in the application.

5. In this case, locate the **BugResolution.xsd** in Appendix A. Test XML. Then copy the contents of **BugResolution.xsd** and paste it into the **Design View** to replace the default schema file.

6. Repeat steps 1 through 5, but this time create an XML Schema file named **ResolutionAppeal.xsd.** Then locate the **ResolutionAppeal.xsd** in Appendix A. Test XML. Then copy the contents of **ResolutionAppeal.xsd** and paste it into the **Design View** to create a the **ResolutionAppeal.xsd** schema file in your Schemas project.

7. Select **File→Save** to save your work.

   This step completes the set up of your new application. Proceed to the next step to begin the work of adding the business logic to the Resolution Approval business process (ResolutionApproval.jpd).

Note: When you create an XSD, a build of the Schemas project folder is triggered. (The build verifies that the schema file is well formed. It also verifies that the element and attribute names in the XML Schema do not conflict with the XSD files that are in the Schemas project.) For more information about what gets generated when you import schemas, see Importing Schemas, which is available at the following URL:

`http://edocs.bea.com/workshop/docs81/doc/en/integration/dtguide/dtguidemaperimportschemas.html`

Step 3. Design How to Start the Business Process

In this step, you begin the design of the Resolution Approval business process. Specifically, you design how the business process is started at run time.

In this step, you design the **Start** node in your business process to receive a *Bug Resolution* message from a client—the receipt of this message is the trigger that starts the business process.You also create a variable to hold the incoming message.

In **Design View**, interactions between a business process and a client application are represented by **Client Request** and **Client Response** nodes. In this case, you add a **Client Request** node to your business process and subsequently create the code on this node to handle the receipt of a message from a client.

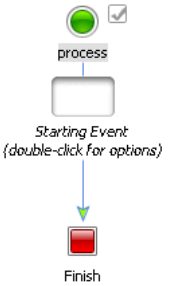
Complete the following tasks to design the **Client Request** node that starts your business process:

- Create a Start Node in Your Business Process
- Design The Communication Between a Client and the Business Process

Create a Start Node in Your Business Process

In Step 2. Create Your Application, you created a business process in the Worklist Tutorial application. This step begins the design of that business process with the bug tracking business logic.

1. With the `Worklist_Application` open in WebLogic Workshop, click **ResolutionApproval.jpd** on the **Application** pane. The business process is displayed in **Design View**.

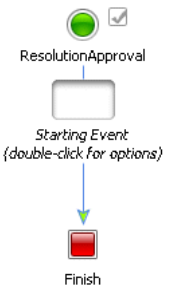


2. Rename the business process to better represent its function:

   a. Click the name (**process**) in the **Design View**

   b. Click **F2** and enter the name you want to give the node—in this case **ResolutionApproval—**click **Enter** on your keyboard**.**

      The business process in **Design View** is named **ResolutionApproval** to match the name you gave the JPD file in the **Application** tab.



3. Double-click the empty **Starting Event** target to display the Start node builder.

   The node builder displays the following options. You select one to specify the method by which you want your business process to start:

   - **Invoked via a Client Request**
   - **Invoked synchronously via a Client Request with Return**
   - **Invoked via one of several Client Requests or Subscriptions (Event Choice)**
   - **Subscribe to a Message Broker channel and start via an event (Timer, Email, File, Adapter, etc.)**

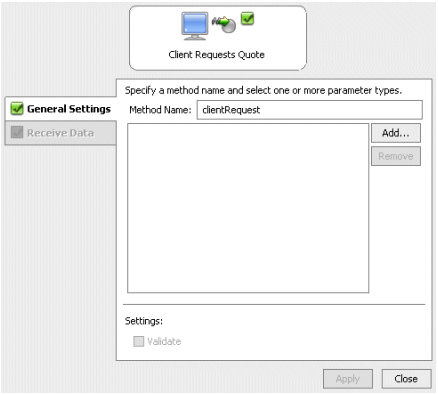
4. Select **Invoked via a Client Request**.
5. Click **OK** to close the node builder. The empty node that was associated with the **Start** node is now populated with a **Client Request** node.

To Design Your Client Request Node

Designing your **Client Request** node includes creating a method and parameters that your client uses to trigger the start of your business process, and designing the logic for handling the receipt of a request from a client.

1. Rename the **Client Request** node. To do so, click the **Client Request** node and press **F2**. Enter **Receive Resolution** as the name to replace **Client Request** for the node. Press **Enter**. Your business process should now resemble that shown in the following figure:

2. In **Design View**, double-click the **Receive Resolution** node to invoke the node builder, as shown in the following figure:



Node builders provide a task-driven user interface that helps you design your the communication between a business process and its clients and other resources.

Design The Communication Between a Client and the Business Process

As shown in the preceding figure, the node builder for a **Client Request** node displays the following tabs to guide your design of the communication between a client and a business process:

- To Specify General Settings
- To Specify Receive Data

To Specify General Settings

The following steps describe how to specify the method exposed by your business process to clients—clients invoke this method to start and make requests on your business process.

1. In the **Method Name** field on the **General Settings** tab, change the default method name (**clientRequest**) to **receiveResolution**.

   Note: When you make your business process available as a service, the name you assign to a method on a **Client Request** node is the name of the method that is exposed via the Web Services Description Language (WSDL). In general, we recommend that you define a name that is representative of the service offered by your business process.

2. Specify a data type for the parameter to your **receiveResolution** method:

   a. Click **Add** on the **General Settings** tab. A panel, which shows the data types you can use is displayed:

   The Bug Resolution message from clients is an XML message. Therefore we are concerned with **XML Types** at this node.

   b. Select **XML**. The panel is populated with a list of XML Schema files (*Typed XML*) and a list of *Untyped* XML objects available in your project.

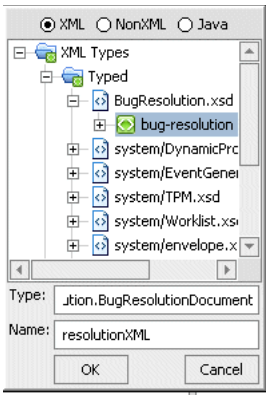
   Note: Remember that you added the XML Schemas you need as you build the Receive Resolution business process in "Create the Tutorial Schemas" on page 3-4.

   In this step, we use **BugResolution.xsd**, to specify the structure of documents that clients can send to start your business process.

c. Click the + associated with **BugResolution.xsd** in the list of **XML Types.**

A graphical representation of the XML Schema defined by `BugResolution.xsd` is displayed in the node builder pane.

d. Click the **bug-resolution** node. (It represents the parent element in your XML document.) The **Type** field is populated with the XML type: **com.bea.wliWorklistTutorial.bugResolution.BugResolutionDocument**.



e. In the **Name** field, replace the default parameter name (**x0**) with **resolutionXML**.

3. Click **OK**. The parameter specifications you made (parameter type is **BugResolutionDocument**, parameter name is **resolutionXML**) is displayed in **General Settings** tab in the node builder.

This step completes the specification of the method exposed to clients by your business process. Messages from clients are expected to be *typed* XML. That is, the messages received from clients must contain XML that is valid against an XML Schema (specifically `BugResolution.xsd`, in this case).

To Specify Receive Data

1. Click **Receive Data** to open the tab that allows you to specify a variable, to which a Bug Resolution message, received from a client, is assigned at run time. By default, the **Receive Data** tab opens on the **Variable Assignment** panel.

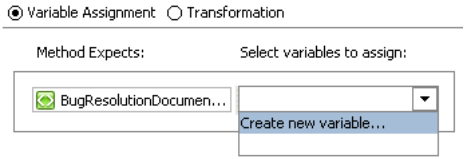
Receive Data tabs have two modes:

- **Variable Assignment**—Use this mode when you want to assign the data received from the client to a variable of the same data type.

- **Transformation—**Use this mode when you want to create a transformation between the data assigned to a variable and that expected by the method parameter.

In this case, we use the **Variable Assignment** mode because we want to assign the XML message received from the client directly to a variable of the same data type—that is, you create a variable of typed XML (`BugResolutionDocument`) to which your process assigns the incoming Bug Resolution from clients.

2. Under **Select variables to assign**, click the arrow and select **Create new variable...**



3. In the **Create Variable** dialog box.

   a. In the **Variable Name** field, enter **resolutionXML**.

   b. In the **Select Variable Type** field, ensure that the **bug-resolution** element is selected (in **BugResolution.xsd** in the list of **XML Types**)

   c. Click **OK**. Your new variable is created and displayed in the **Receive Data** tab. Note that the **resolutionXML** variable is also listed as an **XML** variable in the **Data Palette**.

4. Click **Apply**.

   Both tabs in the node builder (**General Settings** and **Receive Data**) are marked complete .

5. Click **Close** to close the **Client Receive** node builder. This step completes the design of the Start node for your business process.

CHAPTER 5

Step 4. Create Task and Assign to User

This step describes how you design a common pattern in worklist business processes—one that creates a task, assigns the task to a user, and specifies a date by which the completion of the task is due. You design this integration of your business process with a user using a Worklist control—specifically the Task control.

This step includes the following tasks:

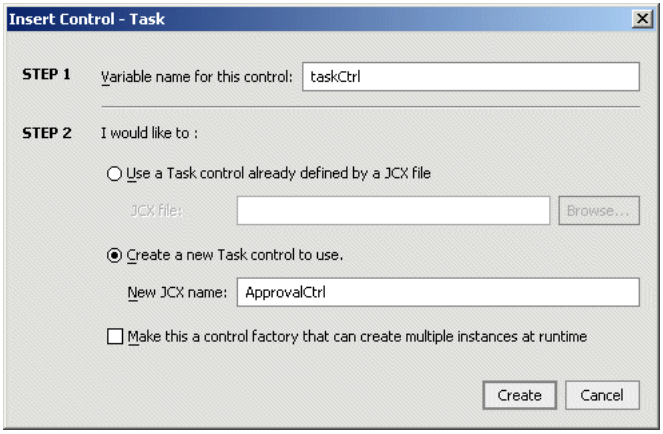
- Create a Task Control
- Create the Approve Resolution Task
- Assign the Task to a User
- Specify a Due Date for Completion of the Task

Create a Task Control

A Task control creates and manages an instance of a Task. To create the Task control that manages the task of approving the resolution of a bug in our SoftCo enterprise, complete the following:

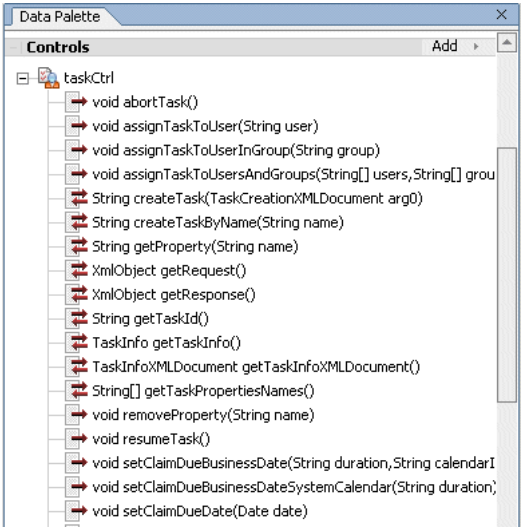
1. In the **Application** pane, double-click **ResolutionApproval.jpd** to ensure that the business process is displayed in the **Design View** tab.
2. Click **Add→Integration Controls→Task** on the **Controls** tab of the **Data Palette** to display the **Insert Control** dialog box.

Note: If the Controls tab is not visible in WebLogic Workshop, click **View→Windows→Data Palette** from the menu bar.



3. In the **Insert Control** dialog box (**Step 1**), enter **taskCtrl** as the name for the instance of this control.
4. In the **Insert Control** dialog box (**Step 2**), select **Create a new Task control to use**. Then, in the **New JCX name** field, enter a **ApprovalCtrl**.
5. Click **Create**. A new Task control and an instance of it are created and the **Insert Control** dialog box is closed.

   A new JCX file (**ApprovalCtrl.jcx**) is created and displayed in the **Application** tab. The instance of the control is displayed in the **Controls** tab on the **Data Palette**. Expand the Control Instance by clicking the + beside its name on the **Data Palette** to display the base methods provided for this control.



For a complete list of the base methods on Task controls, see Using the Worklist Control Methods in *Using the Worklist System*, which is available at the following URL:

`http://edocs.bea.com/wli/docs81/worklist/indexs.html`

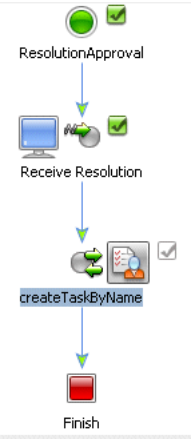
Create the Approve Resolution Task

In this step you design the interaction of the business process with the Task control by simply dragging and dropping the Task control methods from the **Data Palette** onto the **Design View** at the point in your business process at which you want to create the appropriate logic. In this case you create a node in the process that creates the task and another that sends the data about the task to the Task control. To do so follow the steps in the following sections:

- To Create the Approve Resolution Task
- To Send Information About the Task to the Task Control

To Create the Approve Resolution Task

1. Click the + beside the **taskCtrl** control in the **Data Palette** to expand the list of methods available on it.
2. Select the following method: `String createTaskByName(String name)`. Then drag and drop it onto the business process in **Design View**, placing it on the process immediately after the **Receive Resolution** node.



A **Control Send with Return** node is created representing the synchronous call to your **taskCtrl** control. The node is named according to the name of the method you dragged onto the business process—in this case: **createTaskByName**.

Note: This interaction is designed to be synchronous, meaning that the business process blocks waiting for a response from the control. In the case of the `createTaskByName` method, the response is a String which contains the Task ID. A unique Task ID is assigned to each task created using an instance of the Task control.

3. Double-click the **createTaskByName** node to open its node builder. The node builder opens on the **General Settings** tab. The Control instance and target methods are already selected: **taskCtrl** and **String createTaskByName(String name)**, respectively.

4. Click the **Send Data** tab to open the second tab in the **createTaskByName** node.

   By default, the **Send Data** tab opens on the **Variable Assignment** pane. The **Method Expects** field is populated with the data type expected by the `createTaskByName()` method exposed by the **taskCtrl** control: `String name`.

5. Under **Select variables to assign**, click the arrow and select **Create new variable...** The **Create Variable** dialog box is displayed.

6. In the **Create Variable** dialog box.

   a. In the **Variable Name** field, enter **approveResolutionTask**.

   b. In the **Default Value** field, enter **Approve Resolution**. In this way, you specify the name of the task.

c. Select **Declare as constant**.

d. Click **OK**. Your new variable (of type String) is created and displayed in the **Receive Data** tab. Note that the **approveResolutionTask** variable is also listed as a **Java** variable in the **Data Palette**.

7. Click the **Receive Data** tab to open the third tab in the **createTaskByName** node.

   By default, the **Receive Data** tab opens on the **Variable Assignment** pane. The **Method Expects** field is populated with the data type expected to be returned by the `createTaskByName`() method: `String`.

8. Under **Select variables to assign**, click the arrow and select **Create new variable...** The **Create Variable** dialog box is displayed.

9. In the **Create Variable** dialog box.

   a. In the **Variable Name** field, enter **taskId**.

   b. Click **OK**. Your new variable (of type String) is created and displayed in the **Receive Data** tab.

10. Click **Apply**, then **Close** to save the specifications you made to the **assignTaskByName** node and close its node builder**.**

To Send Information About the Task to the Task Control

You designed the business process in this case to be invoked when it receives a Bug Resolution document from a client. The data contained in the Bug Resolution document can be used by the Task control when tasks, due dates, and so on are being assigned. In this step, you create a node in the business process to send the Bug Resolution document data to the Task control. To do so, complete the following steps:

1. Click the + beside the **taskCtrl** control in the **Data Palette** to expand the list of methods available on it.

2. Select the following method: `void setRequest(XmlObject xml)`. Then drag and drop it onto the business process in **Design View**, placing it on the business process immediately after the **createtaskByName** node. A new node (**setRequest**) is created.

3. Double-click the **setRequest** node to open its node builder.

4. Click the **Send Data** tab to open the second tab in the **setRequest** node.

5. Under **Select variables to assign**, click the arrow and select **resolutionXML (BugResolutionDocument)**.

6. Click Apply, then Close to save the specifications you made on this node and close its node builder.

   The Resolution Approval business process should now resemble that shown in the following figure:

Assign the Task to a User

In this step you assign the `approveResolution` task you created in the preceding step to a user in the SoftCo enterprise. You do so in the same way as you created the task—that is, you drag and drop a Task control method from the **Data Palette** onto the **Design View** at the point in your business process at which you want to design the interaction. You already learned in the preceding step how to use a node builder to design the communication on the node

1. On the **taskCtrl** in the **Data Palette**, select the following method: `void assignTaskToUser(String user)`. Then drag and drop it onto the business process in **Design View**, placing it on the process immediately after the **setRequest** node.

2. Double-click the **assignTaskToUser** node in the **Design View**, then click **Send Data** to open the tab that allows you to specify a variable from which you can get the user name to assign. The **Send Data** tab has two modes:

   - **Variable Assignment**—Use this mode when you want to directly assign a variable that holds the data expected by the method—in this case the user name (of type String).
   - **Transformation—**Use this mode when you want to create a transformation between the data assigned to a variable and that expected by the method parameter.

   In this case, you must switch to the **Transformation** mode because the data type required as input to the **taskCtrl** control is a Java String type, and the variable in which the Bug Resolution message (which includes the names of the users to which this bug can be assigned) is stored, is of type XML (that is, `BugResolutionDocument`, which is valid against the `BugResolution.xsd` schema).

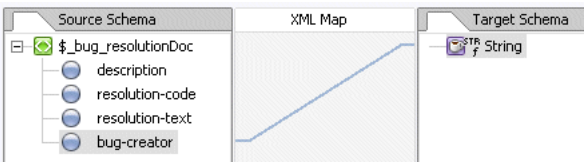
   WebLogic Integration provides a data mapping tool to map between heterogeneous data types. The data transformations you create using the tool are stored in Data Transformation Format (DTF) files.

3. Click **Transformation** to open the pane that allows you to define a transformation between your variable and the expected data type of the parameter on the control method.

4. In **Step 1**, click **Select Variable** to display the variables in your project. Then choose `resolutionXML` (`BugResolutionDocument`)—that is, the variable you created for the **Receive Resolution** node at the start of your business process.

5. In **Step 2**, click **Create Transformation** to open the Transformation tool, which displays a representation of the BugResolution XML document in the **Source Schema** pane, and a **String** in the **Target Schema** pane.

6. Click **bug-creator** in the **Source Schema** pane and drag your mouse pointer over to **String** in the **Target Schema** pane. A line is drawn between the **bug-creator** and **String** elements in the **XML Map** pane. It represents the transformation between the two data types.



7. Double-click **ResolutionApproval.jpd** in the **Application** pane to return to your business process.

   Note: Creating the transformation in the preceding steps creates a Transformation control in your project: A DTF file, named **RequestQuoteTransformation.dtf** is created. An XQ file, which contains the query (written in the XQuery language) for the transformation method is also created. Both the DTF and XQ files are displayed in the **Application** tab. Also, an instance of the Transformation control was created and is represented as **transformations** in the **Data Palette** (**Controls** tab).

8. Click **Close** in the **assignTaskToUser** node builder to close the node builder.

   This step completes the design of the **assignTaskToUser** node. The following node is added to the Resolution Approval business process. Through this node, you specified that the person that created the bug (the **bug-creator**) is assigned the task of resolving the bug:

Note: This method sets the assignees list to a specific Integration User—the user named in the bug-creator element of the XML document that starts this business process. Because this user is the only one on the assignees list, this operation automatically causes the task to be claimed for the specified user.

Specify a Due Date for Completion of the Task

In the preceding step, you designed your process to specify that the person who created the bug (the **bug-creator**) is assigned the task of resolving the bug. In this step, you specify a due date for the task—that is you specify a date by which the user who has been assigned the task must complete it.

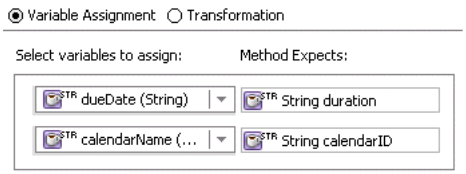
To do so, you create a node on your business process by dragging a method from the **taskCtrl** in the **Data Palette** onto the business process in the same way as you created nodes in the preceding two steps.

1. On the **taskCtrl** in the **Data Palette**, select the following method:

   ```
   void setCompletionDueBusinessDate(String duration String calendarID)
   ```

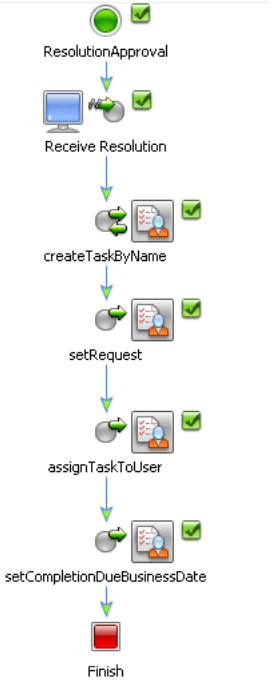
   Then drag and drop it onto the business process in **Design View**, placing it on the process immediately after the **assignTaskToUser** node.

2. Double-click the **setCompletionDueBusinessDate** node in the **Design View**, then click **Send Data** to open the tab that allows you to specify the variables from which your process gets the due date and the calendar name to pass to the task.
3. Under **Select variables to assign**, in the field associated with the **String duration** parameter in the **Method Expects** pane, click the arrow and select **Create new variable...** The **Create Variable** dialog box is displayed.
4. In the **Create Variable** dialog box.
   a. In the **Variable Name** field, enter **dueDate**.
   b. In the **Default Value** field, enter **2 d**. In this way, you specify the duration of business time, by the end of which the task is due to be completed.
   c. Select **Declare as constant**.
   d. Click **OK**. Your new variable (of type String) is created and displayed in the **Receive Data** tab.
5. Under **Select variables to assign**, in the field associated with the **String calendarID** parameter in the **Method Expects** pane, click the arrow and select **Create new variable...** The **Create Variable** dialog box is displayed.
6. In the **Create Variable** dialog box.
   a. In the **Variable Name** field, enter **calendarName**.
   b. In the **Default Value** field, enter **SoftwareTeamCalendar**. In this way, you specify the calendar on which the process bases the calculation of the **2 d** duration you specified for the **dueDate**.
   c. Select **Declare as constant**.
   d. Click **OK**. Your new variable (of type String) is created and displayed in the **Receive Data** tab.



7. Click **Apply**, then **Close** to save the variable assignments and close the **setCompletionDueBusinessDate** node builder.

   The Resolution Approval business process should now resemble that shown in the following figure:



Step 5. Receive Resolution Approval From the Task Owner

In the preceding step, you designed logic in the business process that assigns the task of approving the resolution of a bug to the user that created the bug.

You also designed the business process to allow the task owner two days to approve the resolution of the bug. Within the two days, the task owner can approve the resolution. Alternatively, they can appeal it. It is also possible that the task owner does not respond to the request within the 2 business days allowed.

This step describes how you design your business process to handle the possible events. The business logic in this scenario dictates that the business process handle the receipt of one of multiple possible events from the Task control. When you design business processes in WebLogic Workshop, you can use an **Event Choice** group to create logic that causes the process to wait to receive one of a number of events. In this step, you learn how to design an **Event Choice** group for this scenario.

It includes the following tasks:

- Create an Event Choice Group to Handle the Resolution Approval Events
- Specify the Events the Business Process Can Receive From the Task Control

Create an Event Choice Group to Handle the Resolution Approval Events

When you design business processes in WebLogic Workshop, you can use an **Event Choice** group to create logic that causes the process to wait to receive one of a number of events. After it receives one of the possible events, the flow of the business process continues. You design other nodes within an **Event Choice** group to handle the incoming events. The flow of execution proceeds along one branch in an **Event Choice** node; the branch containing the event that happens first.

To Create an Event Choice Group:

1. Ensure that the Resolution Approval business process is displayed in **Design View**.

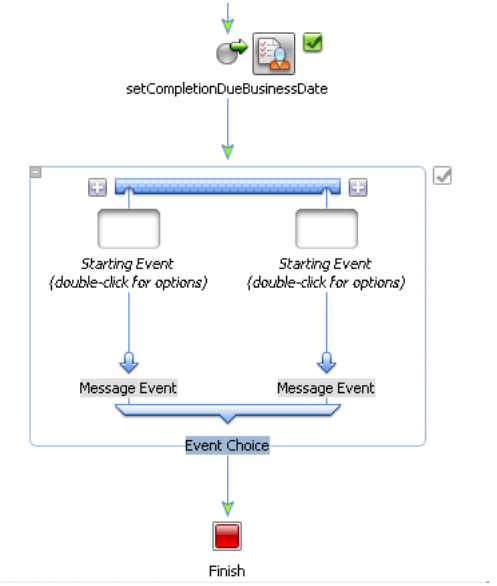
   Note: You can add logic to the business processes you design in WebLogic Workshop by dragging the process node that represents that logic from the **Palette** onto the process in the **Design View**.

   If the **Palette** is not visible in WebLogic Workshop, choose **View→Windows→Palette** from the WebLogic Workshop menu.

2. Click  **Event Choice** in the **Palette**. Then drag and drop it onto the Resolution Approval business process in **Design View**, placing it on the process at the point at which you want to handle the receipt of multiple events.

   In this case, that point is immediately after the **setCompletionDueBusinessDate** you designed when you specified a due date for completion of the task (see "Specify a Due Date for Completion of the Task" on page 5-8).

   The business process is updated in the **Design View** to contain a representation of the **Event Choice** group as shown in the following figure:



Note that by default, **Event Choice** nodes are created with two branches. In this step, you design the Resolution Approval business process to handle a business scenario in which the task control returns one of three events. Therefore, you must expand the default **Event Choice** group to include a third **Message Event** branch. To do so, proceed to the next step.

3. Click on the **Event Choice** group to create an additional **Message Event** branch.

   This step completes the creation of an Event Choice node that you can design to handle the receipt of one of three possible events from the Task control. To proceed with the design, proceed to the next section.

Specify the Events the Business Process Can Receive From the Task Control

The first node on each branch of an **Event Choice** group handles the receipt of one event. At run time, the flow of execution proceeds along one branch in an **Event Choice** group; the branch containing the event that happens first.

In this scenario, you design the Message Event branches to handle the following events:

- The task is completed by the task owner. In other words, the task owner approves the resolution of the bug.

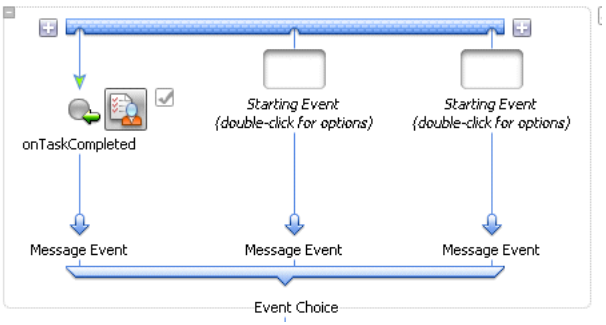
- The resolution of the bug is appealed by the task owner. Appeals must be accompanied by a document that describes the reason for the appeal.
- The task is not addressed by the task owner in the time period specified (in this case, two days). In other words, the task becomes overdue for completion. In our scenario, bugs that are not appealed or approved in two days are assumed approved.

To design the Message Event branches, you specify a callback method on the Task control (**taskCtrl**) to create and bind to a Control Receive node at the beginning of each branch in the **Event Choice** group. To do so, complete the following steps:

1. Click the + beside the **taskCtrl** control in the **Data Palette** to expand the list of methods available on it.
2. Select the following method:

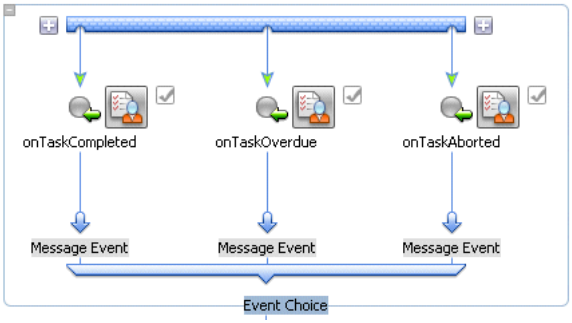
   ```
   void onTaskCompleted(XmlObject response)
   ```

   Then drag and drop it onto the **Event Choice** group in **Design View**, placing it on the left-most **Message Event** branch. By doing so, you specify a Control Receive as the **Starting Event** on the branch, and furthermore bind the **onTaskCompleted** callback method to that Control Receive node. The Event Choice node is updated in the **Design View** to reflect the changes you made:



3. Repeat the preceding step two more times, selecting in turn the following callback methods to drag and drop from the **taskCtrl** control in the **Data Palette**:
   - `void onTaskOverdue(Date time)`
   - `void onTaskAborted(XmlObject response)`

When you complete step 3, the **Event Choice** group in the **Design View** should resemble the following figure:



This step completes the specification of the possible events generated by the Task control. Proceed to the next section to further define how the callback events are handled by the business process.

Design How the Callback Events are Handled by the Resolution Approval Process

At run time, the Resolution Approval business process blocks waiting for an event from the Task control before it proceeds. The flow of execution proceeds along one branch of the **Event Choice** group—the branch that is started by the event that happens first.

- To Design For the Scenario in Which the Bug Resolution is Approved
- To Design For the Scenario in Which the Bug Resolution is Appealed

To Design For the Scenario in Which the Bug Resolution is Approved

No further design is required for the **onTaskCompleted** node or the **onTaskOverdue** node. At run time, for the case in which either of these message events is received, the business process exits the **Event Choice** group and proceeds to the next node in the process, which is a Client Response node that sends an approval message to the client, followed by the **Finish** node that marks the end of the business process. To create this Client Response node:

1. Click **Client Response** in the **Palette**. Then drag and drop it onto the Request Approval business process in **Design View**, placing it on the process immediately after the Event Choice group and immediately before the **Finish** node.
2. Rename the node from **Client Response** to **Resolution Approved**.

3. Double-click the **Resolution Approved** node to open its node builder.
4. On the **General Settings** tab, in the **Method Name** field, enter **approved**.
5. Click **Apply**, then **Close**.

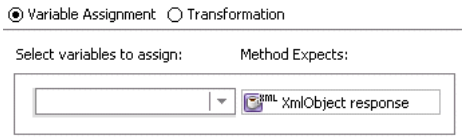
   Note that no further specifications are required on the **Resolution Approved** node. The business logic in our scenario does not require a document is sent to the client as the result of the bug resolution being approved.

To Design For the Scenario in Which the Bug Resolution is Appealed

However, because our business scenario dictates that appeals must be accompanied by a document that describes the reason for the appeal, we want to design additional logic in the process for the case in which an **onTaskAborted** message event is received. The steps in this section describe how to design the **onTaskAborted** branch of the **Event Choice** group to define the flow of execution for the case in which the Task owner appeals the resolution of the bug.

To Design the onTaskAborted Message Event Branch

1. Double-click the **onTaskAborted** node to open its node builder. The node builder opens on the **General Settings** tab. The Control instance and target methods are already selected. They are **taskCtrl** and **void onTaskAborted(XmlObject response)**, respectively.
2. Click the **Receive Data** tab to open the second tab in the **onTaskAborted** node:



3. Create a variable to which the document returned by the Task control (which contains the reason for the appeal) is assigned at run time:

   a. Under **Select variables to assign**, click the arrow and select **Create new variable...** The **Create Variable** dialog box is displayed.

   b. In the **Variable Name** field, enter **appealXML**.

   c. Click **OK**. Your new variable (of type XmlObject) is created and displayed in the **Receive Data** tab.

4. On the **Receive Data** tab, click **Apply**, then **Close**. This saves the specifications you made on this node and close its node builder.

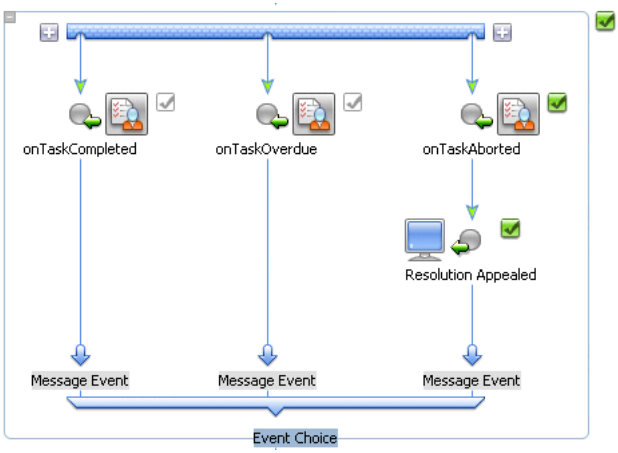
At run time, the document returned by the task owner describing the reasons for appealing the resolution of the bug is assigned to the **appealXML** variable.

It remains only to define a node in the process that returns the document, which contains the reasons for the appeal to the client. (Note that the client is the same resource that starts the Resolution Approval business process.)

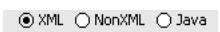
To Design the Client Response in the Case of an onTaskAborted Message

1. Click **Client Response** in the **Palette**. Then drag and drop it onto the Request Approval business process in **Design View**, placing it on the process immediately after the **onTaskAborted** node in the **Event Choice** group.
2. Rename the node from **Client Response** to **Resolution Appealed**.

   The business process is updated in the **Design View**, as shown in the following figure:



3. Double-click the **Resolution Appealed** node to open its node builder.
4. On the **General Settings** tab, in the **Method Name** field, enter **appealed**.
   a. Click **Add** on the **General Settings** tab. A panel, which shows the data types you can use is displayed:



   The appeal message returned to the business process from the Task control in the preceding node is an (untyped) XML message. Therefore we are concerned with **XML** at this node.

b. Select **XML**. Click the **+** beside **Untyped** to expand the list of *Untyped* XML objects available in your project.

c. Select **XmlObject**. Then in the **Name** field, enter **appealXML** to replace the default parameter name (**x0**).

d. Click **OK**. The parameter specifications you made (parameter type is **XmlObject**, parameter name is **appealXML**) are displayed in **General Settings** tab in the node builder.

5. Click the **Send Data** tab, and under **Select variables to assign**, select **appealXML**.

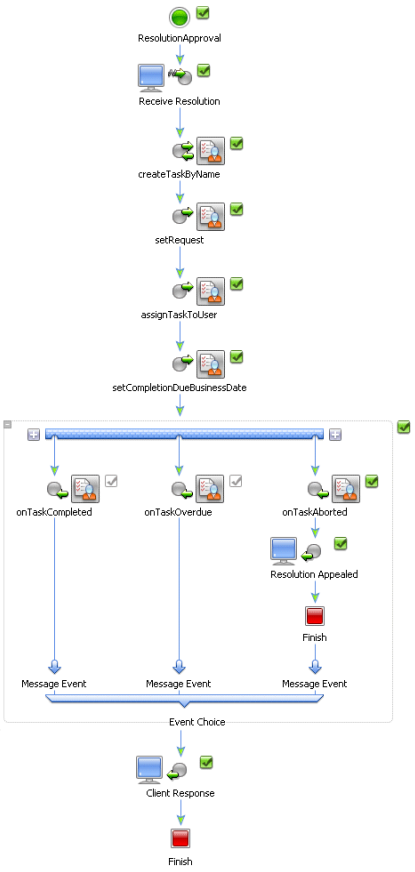
6. Click **Apply**, then **Close**.

   This step completes the specification of the callback method exposed to clients by the Resolution Approval business process.

7. Add a **Finish** node immediately after the **Appealed** node on the **onTaskAborted** branch, specifying that at run time the business process is terminated immediately after the **appealXML** document is sent to the client that invoked the business process.

   This step completes the design of the **Event Choice** group, and the Resolution Approval business process. In the **Design View**, the completed Resolution Approval process should resemble that shown in the following figure.

The Resolution Approval Business Process



Step 5. Receive Resolution Approval From the Task Owner

Step 6. Run the Resolution Approval Business Process

You can run and test the functionality of the business process you created using WebLogic Workshop's browser-based interface. Using the Workshop Test Browser, you play the role of the client, invoking the methods on the business process and viewing the responses.

You also use the Worklist user interface to play the role of QualityEngineerA at the SoftCo enterprise. This step describes how to test two scenarios; one in which QualityEngineerA approves the resolution of the bug, one in which QualityEngineerA appeals the resolution of the bug.

To Launch the Test Browser

1. In the **Application** pane, select **RequestQuote.jpd**—the business process you want to test.
2. If it not already selected, select the **Design View** tab. The business process you selected in the **Application** pane is displayed in **Design View**.
3. If it is not already running, start WebLogic Server. To do so, choose **Tools→WebLogic Server→Start WebLogic Server** from the WebLogic Workshop menu.

   If WebLogic Server is running, the following indicator is visible in the status bar at the bottom of the WebLogic Workshop visual development environment:

4. Press **F7**, or from the WebLogic Workshop menu, click **Build→Build Application**. WebLogic Workshop builds your application.

5. When the build is complete, click the Start button  on the menu bar to run your business process. The **Workshop Test Browser** is launched, through which you can test your business process using sample input values.

6. Click the **Test Form** tab to open the Test Form page.

   You can enter data that your business process can receive as part of a client request directly on the **Test Form** page.

   Note About Test XML Data

   In this tutorial scenario, sample XML data is provided in Appendix A. Test XML. You can open the appendix in a Web browser and copy the test XML data into the Test Form tab in the WebLogic Workshop Test Browser.

7. In this case, locate the **sampleFixResolution.xml** in Appendix A. Test XML.

8. Select the XML data provided in **sampleFixResolution.xml**, then copy it and paste it into the Test Form page— in the box labeled **xml resolutionXML**.

9. Click the button labeled with the method name on your business process (**receiveResolution**) to invoke the method. The **Test Form** page refreshes to display a summary of your request parameters and the calls in the **Message Log**.

   The first call is logged in the Message Log. It is the **receiveResolution** call you made to invoke the business process.

   To Launch the Worklist User Interface

10. To drive the further execution of the business process, you must log into the Worklist user interface to play the role of **QualityEngineerA**, the person who created the bug (examine the **bug-creator** element in **sampleFixResolution.xml**—the test XML you used to invoke the business process), and one of the users you set up for the SoftCo scenario.

    Log on to the Worklist UI in one of the following ways:

    – From the WebLogic Workshop menu:

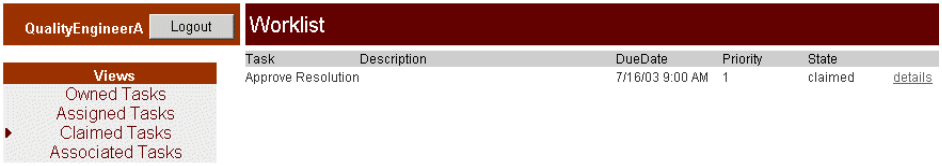
      Tools→WebLogic Integration→Worklist

    – Enter the following URL in a Web browser:

      `http://localhost:7001/worklist`

    Enter **QualityEngineerA** for the username and use the password you set up for QualityEngineerA in "Configure the Users and Groups" on page 2-4.

11. Click **Claimed Tasks** to see the task. (The task was assigned directly to QualityEngineerA, who claims it automatically.)



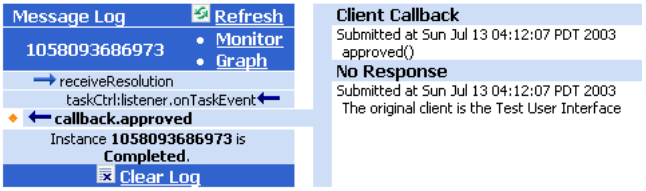
12. Click **details** to open a page that displays the details of the claimed task.

13. Click **response** to open a page that displays the XML describing the resolution (the RequestXML).

14. Click the **Back** button in your browser to go back to the **Task details** page.

15. On the **Task details** page, click **start**, then **complete**. In this way you complete the task to approve the resolution.

16. Return to the WebLogic Workshop Test Browser and click **Refresh** on the **Message Log** pane. Note that the business process runs to completion.



The following calls are logged in the Message Log:

- **receiveResolution**—the call you made (acting in the role of the client) to invoke the business process.
- **taskCtrl:listener.onTaskEvent**—the call from the Task Control to the Resolution Approval business process.
- **callback.approved**—the callback to the client that invoked the business process. In this case the method is the **approved** method you created on the **Resolution Approved** node in the business process.

Congratulations! You have successfully completed running and testing the Resolution Approval business process for a scenario in which the user (QualityEngineerA) approves the resolution of the bug.

The following steps describe the data and steps you need to run the business process again, but this time, QualityEngineerA appeals the task—that is, QualityEngineerA appeals the resolution of the bug.

Run ResolutionApproval.jpd Again . . .

1. Run the ResolutionApproval.jpd business process again.
2. Locate the **sampleFixResolution.xml** in Appendix A. Test XML.
3. Select the XML data provided in **sampleFixResolution.xml**, then copy it and paste it into the Test Form page in the WebLogic Workshop Test Browser— in the box labeled **xml resolutionXML**.
4. Click the button labeled with the method name on your business process (**receiveResolution**) to invoke the method. The first call is logged in the Message Log. It is the **receiveResolution** call you made to invoke the business process.
5. Access the Worklist UI, again playing the role of QualityEngineerA.
6. Click **details** to open a page that displays the details of the claimed task.

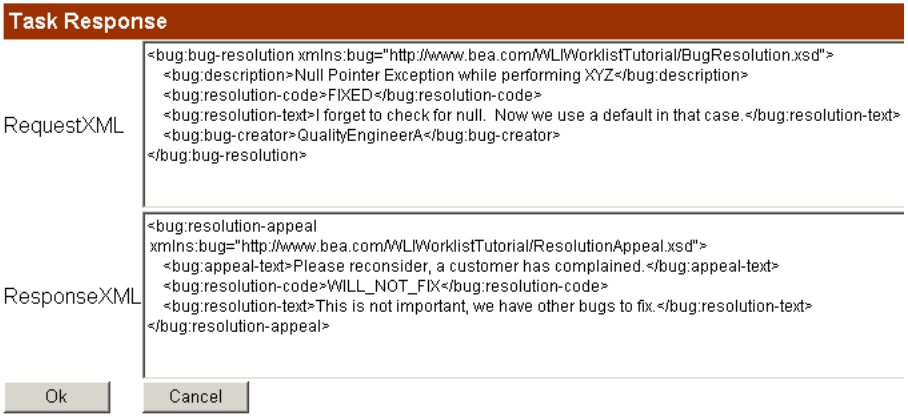
   Note: On the Details page, note the values in the Creation Date and the Due Date fields. Note that the owner of the task (QualityEngineerA) has two business days in which to accept or appeal the resolution of the software bug.

7. Click **response** to open a page that displays the XML describing the resolution (the RequestXML).

   Note: Recall that the SoftCo enterprise requires that users who appeal the resolution of a bug attach a document to describe the reason for their appeal. In this scenario, QualityEngineerA appeals the task and must include the document that describes why. The following steps describe how to test this scenario.

8. Locate the **sampleAppeal.xml** in Appendix A. Test XML.

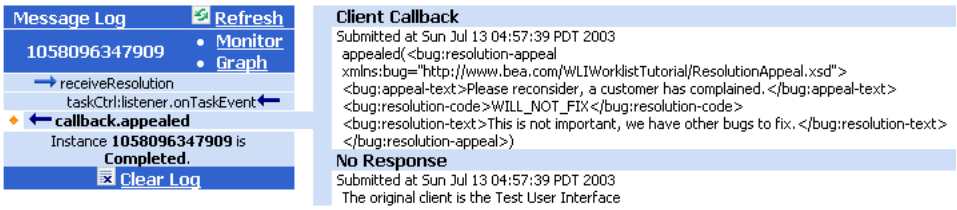
9. Select the XML data provided in **sampleAppeal.xml**, then copy it and paste it into the **Task Response** page, specifically into the **ResponseXML** box:



10. Click **Ok**. The browser returns you to the **Task details** page.

11. On the **Task details** page, click **start**, then **abort**. In this way you abort the task to appeal the resolution.

12. Return to the WebLogic Workshop Test Browser and click **Refresh** on the **Message Log** pane. Note that the business process runs to completion.



Note that in this case, the final callback logged in the Message Log is **callback.appealed**—the callback to the client that invoked the business process. In this case the method is the **appealed** method you created on the **Resolution Appealed** node in the **onTaskAborted** Message Event branch in the business process.

This step completes running and testing the Resolution Approval business process for a scenario in which the user (QualityEngineerA) appeals the resolution of the bug.

A Test XML

This appendix provides test XML data for you to use when you run and test the Resolution Approval business process you create by working through this tutorial. To use the sample data, copy and paste the contents of the listings as described in "Step 2. Create Your Application" on page 3-1 and "Step 6. Run the Resolution Approval Business Process" on page 7-1.

The following XML Schema and sample XML documents are included in this section:

- BugResolution.xsd
- ResolutionAppeal.xsd
- sampleFixResolution.xml
- sampleAppeal.xml

Listing A-1 BugResolution.xsd

```
<?xml version="1.0"?>
<xs:schema elementFormDefault="qualified"
attributeFormDefault="unqualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://www.bea.com/WLIWorklistTutorial/BugResoluti
on.xsd"
    targetNamespace="http://www.bea.com/WLIWorklistTutorial/BugRe
solution.xsd">

    <xs:element name="bug-resolution"
type="tns:bug-resolution-type"/>

        <xs:complexType name="bug-resolution-type">
      <xs:sequence>
```

```
        <xs:element name="description" type="xs:string"/>
        <xs:element name="resolution-code" type="xs:string"/>
        <xs:element name="resolution-text" type="xs:string"/>
        <xs:element name="bug-creator" type="xs:integer"/>
      </xs:sequence>
        </xs:complexType>
</xs:schema>
```

Listing A-2 ResolutionAppeal.xsd

```
<?xml version="1.0"?>

<xs:schema elementFormDefault="qualified"
attributeFormDefault="unqualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://www.bea.com/WLIWorklistTutorial/ResolutionA
ppeal.xsd"


targetNamespace="http://www.bea.com/WLIWorklistTutorial/Resolutio
nAppeal.xsd">

        <xs:element name="resolution-appeal"
type="tns:resolution-appeal-type"/>

              <xs:complexType name="resolution-appeal-type">

            <xs:sequence>

                <xs:element name="appeal-text" type="xs:string"/>

             <xs:element name="resolution-code" type="xs:string"/>

     <xs:element name="resolution-text" type="xs:string"/>

            </xs:sequence>

              </xs:complexType>

</xs:schema>
```

Listing A-3 sampleFixResolution.xml

```
<bug:bug-resolution


xmlns:bug="http://www.bea.com/WLIWorklistTutorial/BugResolution.x
sd">

    <bug:description>Null Pointer Exception while performing
XYZ</bug:description>

    <bug:resolution-code>FIXED</bug:resolution-code>

   <bug:resolution-text>I forget to check for null.  Now we use a
default in that case.</bug:resolution-text>

    <bug:bug-creator>QualityEngineerA</bug:bug-creator>

</bug:bug-resolution>
```

Listing A-4 sampleAppeal.xml

```
<bug:resolution-appeal


xmlns:bug="http://www.bea.com/WLIWorklistTutorial/ResolutionAppea
l.xsd">

    <bug:appeal-text>Please reconsider, a customer has
complained.</bug:appeal-text>

    <bug:resolution-code>WILL_NOT_FIX</bug:resolution-code>

   <bug:resolution-text>This is not important, we have other bugs
to fix.</bug:resolution-text>

</bug:resolution-appeal>
```