BEA WebLogic Integration™

BPM – Workshop Interoperability Sample

Release 7.0 SP2
Document Date: February 2003

Copyright

Restricted Rights Legend

Trademarks or Service Marks

BEA WebLogic Integration 7.0 Interoperability Sample

Part Number	Date	Software Version
N/A	February 2003	7.0 SP2

Contents

Index

About This Document

This document explains how to use the WebLogic Integration BPM – Workshop Interoperability Sample.

The document is organized as follows:

- Chapter 1, “Overview,”provides information about the two types of message exchanges demonstrated by the sample: SOAP over HTTP and XML over HTTP.
- Chapter 2, “Running the Sample,” provides information about how to set up and run the sample.
- Chapter 3, “BPM-Workshop Interoperability Process,” describes what happens when you run the sample.

What You Need to Know

This document is intended for users who want to exchange XML documents using Web services.

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.beasys.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 using this product.

- *Learning to Use BPM with WebLogic Integration*
- *Using the WebLogic Integration Studio*

Contact Us!

Your feedback on the WebLogic Integration documentation is important to us. Send us e-mail at docsupport@beasys.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 WebLogic Integration Worklist, or if you have problems installing and running the product, contact BEA Customer Support through BEA WebSUPPORT at `www.beasys.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.

Convention	Item

`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`*
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.

Convention	Item

. . .	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.

1 Overview

The WebLogic Integration BPM – Workshop sample shows how Web services and workflows can invoke one another by exchanging XML documents. The sample demonstrates two scenarios:

- A workflow invokes a Web service with XML and later receives XML from the Web service.
- A Web service invokes a workflow with XML and later receives XML from the workflow.

The sample also demonstrates two message formats:

- Simple Object Access Protocol (SOAP) using Hypertext Transfer Protocol protocol (HTTP). The exchanges can be conversational or include asynchronous replies.
- Raw XML using HTTP. The message exchanges are non-conversational.

Note: The BPM – Workshop sample runs only on Window systems.

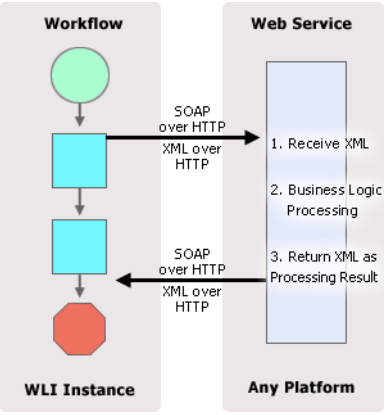
This section contains information on the following topics:

- Interoperation Scenario
- How the Sample Demonstrates the Scenario

Interoperation Scenario

Figure 1-1 shows how a workflow can invoke a Web service with an XML message and then some time later receive an XML message from that Web service.

Figure 1-1 Workflow Invokes Web Service with XML and Later Receives XML from the Web Service



The workflow invokes the Web service with an XML document by passing in an XML message over SOAP-HTTP or HTTP-XML. The Web service then begins processing business logic. At some point during or after the processing, the Web service sends a message containing an XML document back to the workflow.

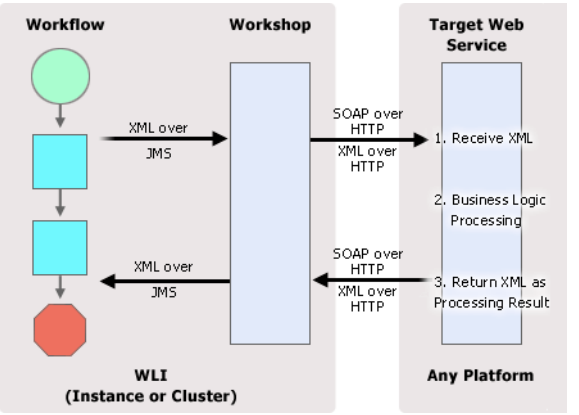
This scenario features an asynchronous Web service because the workflow is free to perform its processing after instantiating the Web service. The workflow can then later solicit the response from the Web service. Note that each instance of the Web service must send the XML result back to the workflow instance that invoked it.

Note: The Web service methods in the XML over HTTP sample are, strictly speaking, synchronous. This is due to a feature in WebLogic Workshop where the JWS methods that support XML-only bindings over HTTP do not support void return values. Subsequently, it is not possible to have asynchronous XML/HTTP only methods in a Workshop JWS. The sample handles this limitation by having the JWS return dummy integers as return values.

How the Scenario is Implemented

An intermediary Web service is employed to decouple the workflow from the protocol and message format of the target Web service, as shown in Figure 1-2.

Figure 1-2 Workflow Invokes a Web Service with XML and Later Receives XML from the Web Service



The intermediary Web service ensures that the reply XML received from the target Web service comes from the identical Web service instance that it invoked by sending the original XML message. The intermediary must also ensure that it conveys the XML result from the target Web service back to the exact workflow instance that passed it the original XML document.

The workflow requires a mechanism for transmitting its raw XML to the intermediary Web service and receiving an XML response from the intermediary Web service at a later time. Note that both of these XML transmissions must be guaranteed to take place between the same intermediary Web service instance and workflow instance. The intermediary Web service also frees up the workflows from having to support a SOAP/XML over HTTP interface. The workflows act as JMS producers and consumers leaving the Web services to handle the SOAP/XML message format translation.

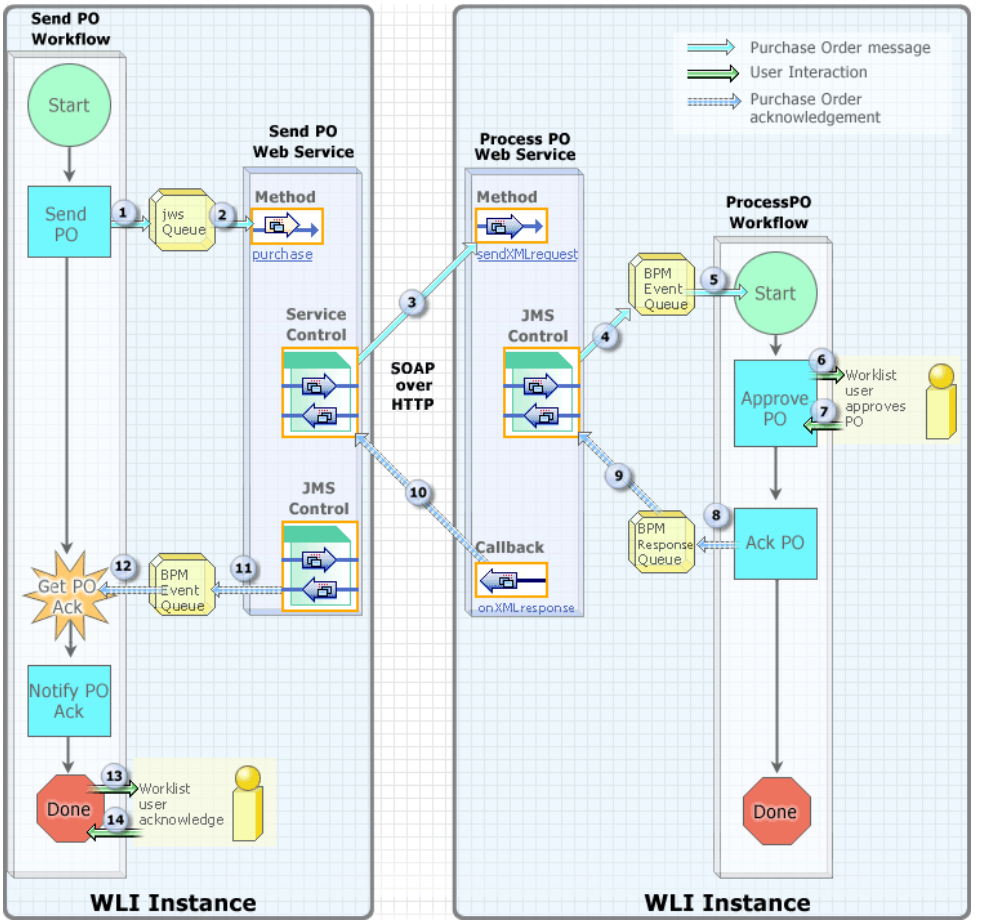
How the Sample Demonstrates the Scenario

As previously mentioned, the sample demonstrates that a workflow can generate an XML document containing purchase order information. In the sample, the SendPO workflow uses an XML document to trigger the ProcessPO workflow. The ProcessPO workflow eventually returns a result XML document to the SendPO workflow after it completes processing the purchase order information.

Figure 1-3 shows the interoperability process for the SOAP over HTTP sample. Table 1-1 describes the process in detail.

Figure 1-4 shows the interoperability process for the XML over HTTP sample. Table 1-2 describes the process in detail.

Figure 1-3 Workflow – Workshop Interaction—SOAP



Note: Each step is described in Table 1-1.

As shown in Figure 1-3, the ProcessPO workflow is invoked by the SendPO workflow. The Process PO Web service is deployed as means of invoking the ProcessPO workflow and retrieving its response XML asynchronously. This allows the SendPO workflow to effectively act as a client of the Process PO Web service when it invokes the ProcessPO workflow and receives its result. For this to be feasible, the Process PO Web service must be reachable via HTTP from the WebLogic Integration instance that executes the SendPO workflow.

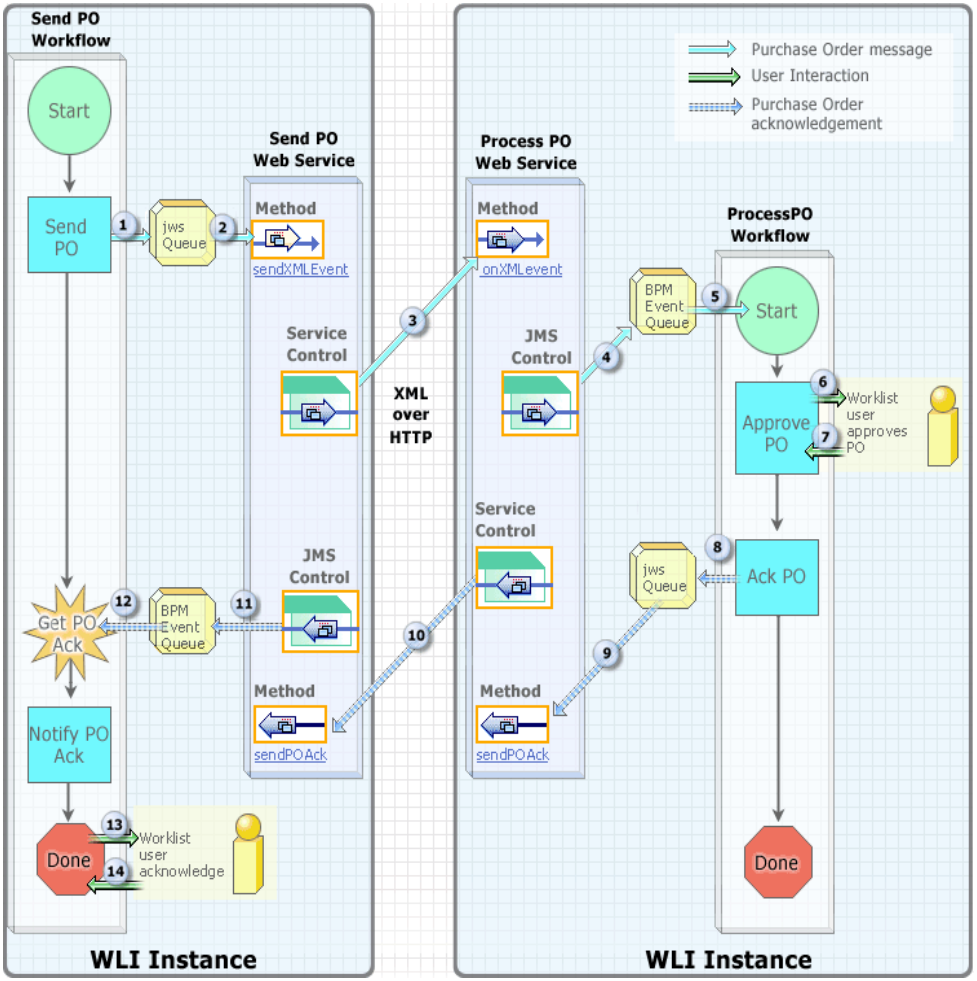
The Send PO Web service handles the details of invoking the Process PO Web service and receiving a result. When the Send PO Web service has the result XML, it conveys it to the SendPO workflow. The SendPO and ProcessPO workflows use the Send PO and Process PO Web services as a means of communicating XML documents to each other asynchronously over HTTP.

Table 1-1 SOAP Over HTTP Sample Scenario Steps

Step	Description
1	The SendPO workflow posts the XML document containing the purchase order on the JMS queue dedicated for Web services (`jws.queue`). The XML document also includes the SendPO workflow instance ID. When the SendPO Web service returns a response (step 11), it assigns that instance ID to the JMS property WLPIInstanceIDs. This ensures the following: ■ The SendPO workflow instance that sent the purchase order is the same workflow instance that receives the response. ■ The response will be received even if the SendPO workflow event node Get PO Ack has not yet been activated by the time the response arrives.
2	The Send PO Web service purchase method has a parameter that maps to the purchase order XML document. Additionally, this method handles the XML message that arrived on `jws.queue`. **Note:** The receive queue of a JMS control cannot be used to receive unsolicited messages. For more information, see the WebLogic Workshop online help—Guide to Building JMS Control: Using Java Message Service Queues and Topics from Your Web Service—Messaging Scenarios Not Supported by the JMS Control.
3	The Send PO Web service sends the purchase order document to the Process PO Web service. The Send PO Web service uses a service control created using the Process PO jws file. The SendPO Web service and the Process PO Web service exchange SOAP formatted XML messages over HTTP. SOAP message exchanges allow the SendPO Web service and the Process PO Web service to reside on different machines, which allows remote communication between the SendPO workflow and the ProcessPO workflow. The SOAP message exchange between the Send PO Web service and the Process PO Web service is conversational, and to ensure request/response correlation between the two Web services, it carries the SOAP message conversation ID in the SOAP header. This sample can also run on a single machine.

Step	Description
4	The Process PO Web service uses a JMS control to place the purchase order XML document on the BPM Event Queue, which is monitored by the ProcessPO workflow. The JMS message contains the SOAP conversation ID in its JMSCorrelationID header field to ensure that the JMS reply message returns to the Process PO Web service instance that sent the outgoing message.
5	The ProcessPO workflow retrieves the XML message from the BPM Event Queue and begins processing.
6	The ProcessPO assigns a task to a WLI Worklist user that requests the user to accept the purchase order.
7	The user executes the task, accepting the purchase order.
8	The ProcessPO workflow places a purchase order acknowledgement XML message on the BPM Response Queue. The acknowledgement message contains the SOAP message conversation ID in its header field JMSCorrelationID to ensure that only the intended instance of the Process PO Web service receives this message.
9	The Process PO Web service has a JMS control that retrieves the purchase order acknowledgement XML message from the BPM Response Queue.
10	The Process PO Web service invokes its callback method to forward the purchase order acknowledgement message to the SendPO Web service. As previously mentioned, either SOAP over HTTP is used for communication between the Web services.
11	The callback for the service control uses a JMS control to place the purchase order acknowledgement XML message on the BPM Event Queue.
12	The SendPO workflow retrieves the purchase order acknowledgement XML message from the BPM Event Queue. The Get PO Ack event node is triggered by the receipt of an addressed message containing the workflow instance ID of the Send PO Workflow in the message header. This ensures that the workflow instance that had sent out the purchase order document has its event node triggered on the receipt of an PO acknowledgement message.
13	The SendPO workflow assigns a task to a WebLogic Integration Worklist user, which requests the user to accept the purchase order acknowledgment.
14	The user executes the task, accepting the purchase order acknowledgement.

Figure 1-4 Workflow – Workshop Interaction—XML



Note: Each step is described in Table 1-2.

As shown in Figure 1-4, the ProcessPO workflow is invoked by the SendPO workflow. The Process PO Web service is deployed as means of invoking the ProcessPO workflow and retrieving its response XML asynchronously. This allows the SendPO workflow to effectively act as a client of the Process PO Web service when it invokes

the ProcessPO workflow and receives its result. For this to be feasible, the Process PO Web service must be reachable via HTTP from the WebLogic Integration instance that executes the SendPO workflow.

The Send PO Web service handles the details of invoking the Process PO Web service and receiving a result. When the Send PO Web service has the result XML, it conveys it to the SendPO workflow. The SendPO and ProcessPO workflows use the Send PO and Process PO Web services as a means of communicating XML documents to each other asynchronously over HTTP.

Table 1-2 XML Over HTTP Sample Scenario Steps

Step	Description
1	The SendPO workflow posts the XML document containing the purchase order on the JMS queue dedicated for Web services (`jws.queue`). The XML document also includes the SendPO workflow instance ID. When the SendPO Web service returns the PO Ack response (step 11), the message payload carries the Send PO workflow instance ID. The Send PO workflow instance's Get PO Ack event node looks for an event key containing the instance ID in the message payload. **Note:** Notice that this form of request response correlation is different from the SOAP sample, where the Send PO workflow's Get PO Ack event node is triggered by the receipt of an addressed message.
2	The Send PO Web service `sendXMLEvent` method has a parameter that maps to the purchase order XML document. Additionally, this method handles the XML message that arrived on `jws.queue`. **Note:** The receive queue of a JMS control cannot be used to receive unsolicited messages. For more information, see the WebLogic Workshop online help—Guide to Building JMS Control: Using Java Message Service Queues and Topics from Your Web Service—Messaging Scenarios Not Supported by the JMS Control.
3	The Send PO Web service sends the purchase order XML document to the Process PO Web service by calling the Process PO Web service's `onXMLEvent` method. The Send PO Web service uses a service control created using the Process PO Web service's jws file (`WLW2BPM.jws`). The Send PO Web service and the Process PO Web service exchange raw XML messages over HTTP. **Note:** Notice that this contrasts with the SOAP sample, where the two Web services communicated with each other using SOAP messages.

Step	Description
4	The Process PO Web service uses a JMS control to place the purchase order XML document on the BPM Event Queue, which is monitored by the ProcessPO workflow. Note the message payload carries the instance ID of the Send PO workflow.
5	The ProcessPO workflow retrieves the XML message from the BPM Event Queue and begins processing.
6	The ProcessPO assigns a task to a WebLogic Integration Worklist user that requests the user to accept the purchase order.
7	The user executes the task, accepting the purchase order.
8	The ProcessPO workflow places a purchase order acknowledgement XML message on a Workshop JMS Queue (`jws.queue`). The message header carries the information identifying the Process PO Web service method (sendPOAck in `WLW2BPM.jws`) that will be invoked when the message is received on the JMS queue.
9 **10**	The `sendPOAck` method in the Process PO Web service invokes the `sendPOAck` method of the Send PO Web service through the Service control defined for the Send PO Web service (`BPM2WLWControl.ctrl`). As previously mentioned, the message exchange between the Process PO Web service and the Send PO Web service is in raw XML format over HTTP.
11	The Send PO Web service uses a JMS control to place the purchase order acknowledgement XML message on the BPM Event Queue. The acknowledgement message carries the instance ID of the Send PO workflow in the message payload.
12	The SendPO workflow retrieves the purchase order acknowledgement XML message from the BPM Event Queue. The Get PO Ack event node uses an event key value expression to accept only the message that contains the workflow instance ID which is the same as its own.
13	The SendPO workflow assigns a task to a WebLogic Integration Worklist user that requests the user to accept the purchase order acknowledgment.
14	The user executes the task, accepting the purchase order acknowledgement.

Why Only One Instance of WebLogic Integration

The sample runs on one instance of WebLogic Integration. In actuality the SendPO workflow and Send PO Web service can be deployed on one instance and the Process PO Web service and ProcessPO workflow can be deployed on another instance, as long as the two instances can reach one another over HTTP.

So that the sample can be run a step at a time, the WebLogic Integration Swing Worklist client is used to get input from a user at two intervals during the sample. The Worklist interaction serves only to break the running of the sample into sections that the user can watch execute one at a time.

2 Running the Sample

Setting Up the Sample

This section tells you how to run the BEA WebLogic Integration BPM WebLogic Workshop Interoperability Sample.

Requirements

Note: Before running this sample, you must have installed WebLogic Platform SP2 using the Typical Installation option. A custom install does not install the Platform Domain, which is used by the Sample. For more information, see Installing WebLogic Platform at http://edocs.bea.com/platform/docs70/install/index.html.

The sample requires a basic knowledge of WebLogic Integration and WebLogic Workshop. You should be familiar with the following concepts:

- Creating a Web service that supports HTTP or JMS
- Using a Service Control or JMS Control in a Web service
- Adding Conversational Support to a Web service

For more information, see the WebLogic Workshop documentation at http://e-docs.bea.com/workshop/docs70/index.html and *Learning to Use BPM with WebLogic Integration* at http://e-docs.bea.com/wli/docs70/bpmtutor/index.htm

How to set Up the Sample

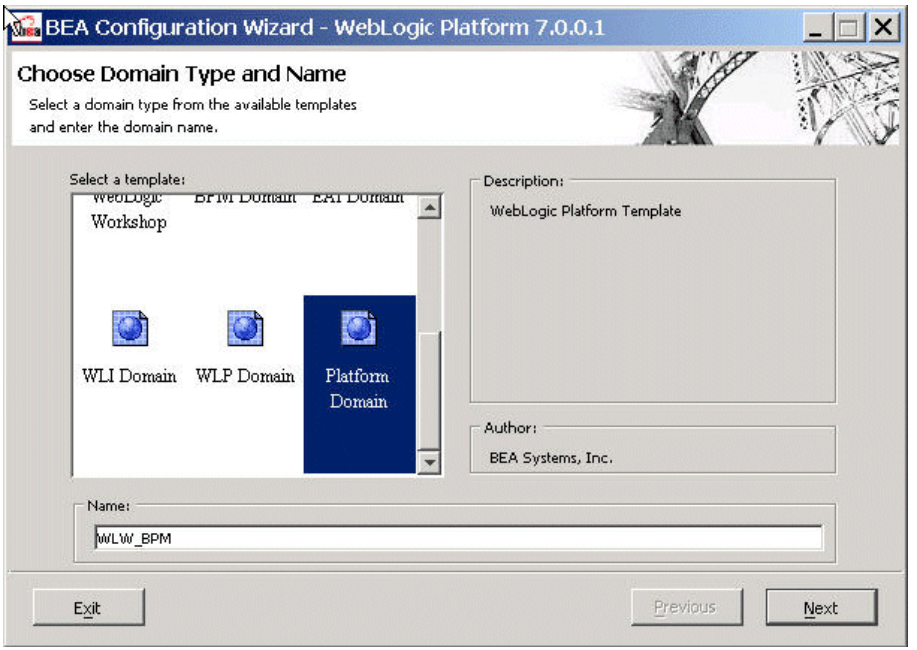
To set up the sample, complete the following steps:

Step: 1 Run the Domain Configuration Wizard

Note: If you have already created the WLW_BPM domain, go to "Step 2: Configure WebLogic Server" on page 2-7.

1. Start the Domain Configuration Wizard by choosing Start→Programs→BEA WebLogic Platform 7.0→Domain Configuration Wizard.
2. After the Configuration Wizard is running, select the Platform Domain template, as shown in Figure 2-1.

Figure 2-1 BEA Configuration Wizard



3. In the Name field, enter "WLW_BPM", and then click Next.
4. In the Choose Server Type window, select Single Server, and then click Next.

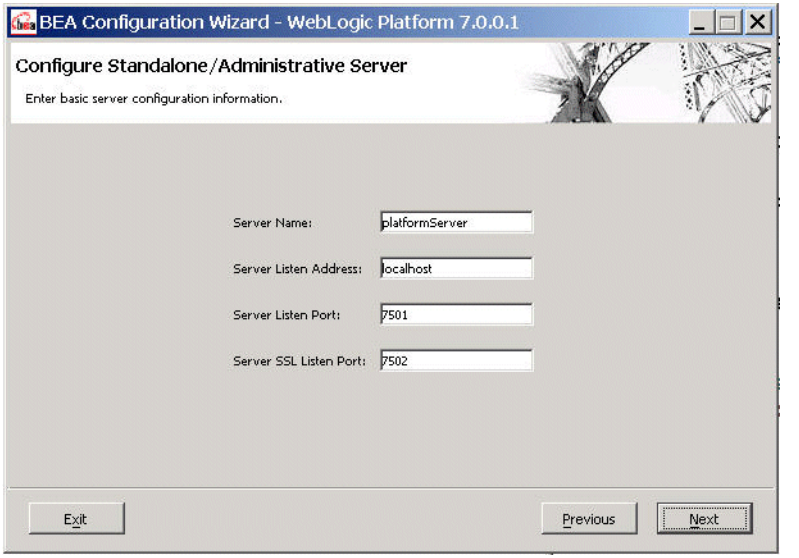
5. In the Choose Domain Location window, specify the following directory for the WLW_BPM domain, and then click Next:

   BEA_HOME\user_projects\

   This will create the *BEA_HOME*\user_projects\WLW_BPM directory.

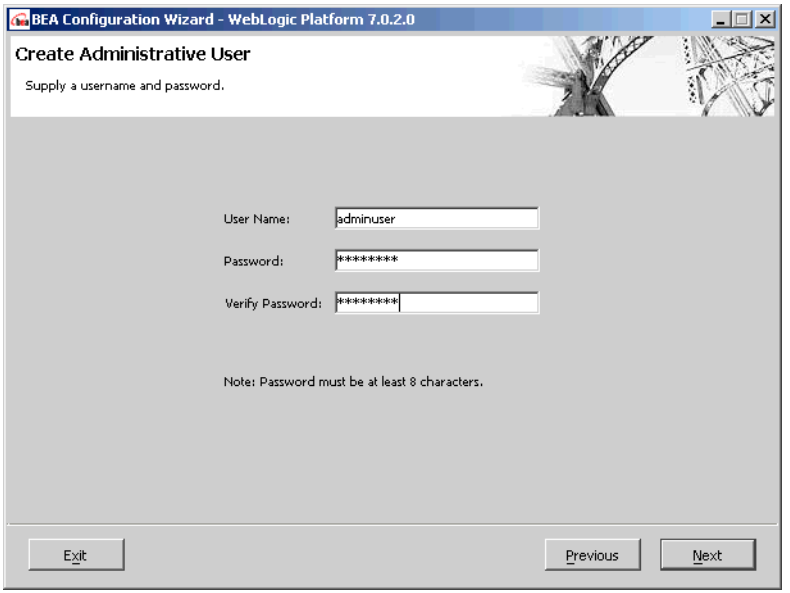
6. In the Configure Standalone/Administrative Server window, chose the default values, as shown in Figure 2-2, and then click Next.

Figure 2-2 Configure Standalone/Administrative Server



7. In the Create Administrative User window (Figure 2-3), enter a username and password, and then click Next.

Figure 2-3 Create Administrative User

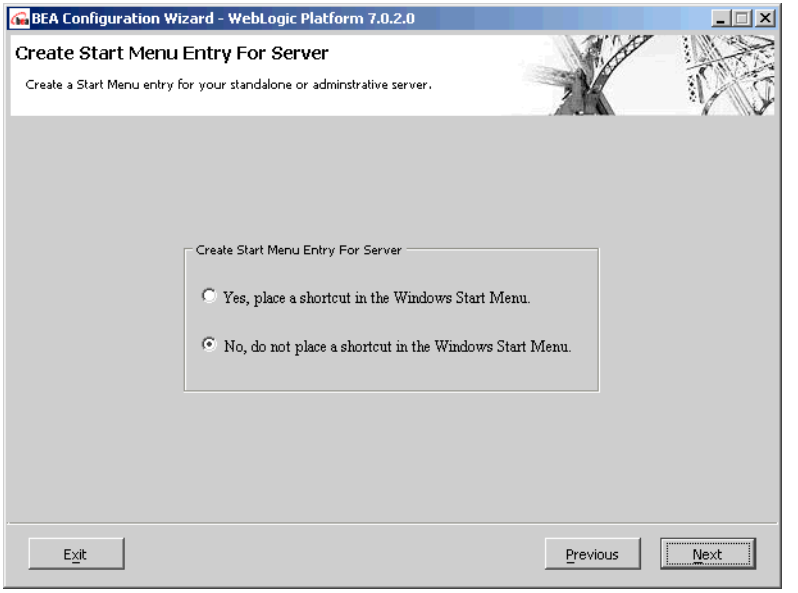


8. In the Configure Database Mail Session window, enter an email address and host, and then click Next.

   Note: The workflows in the sample do not send email. However, you must provide this information to proceed.

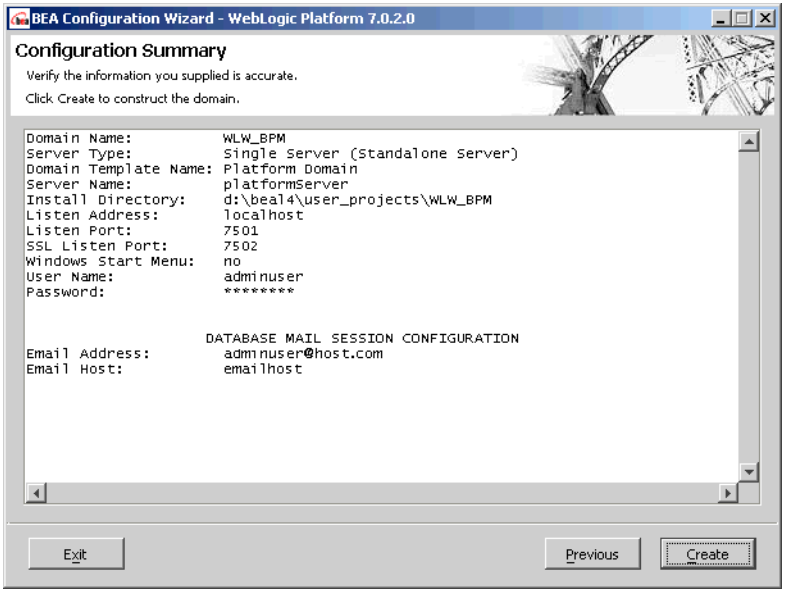
9. *Windows only.* In the Create Start Menu Entry For Server window, select the radio button of your choice, and then click Next. If you are unsure, select No.

Figure 2-4 Create Start Menu Entry for Server



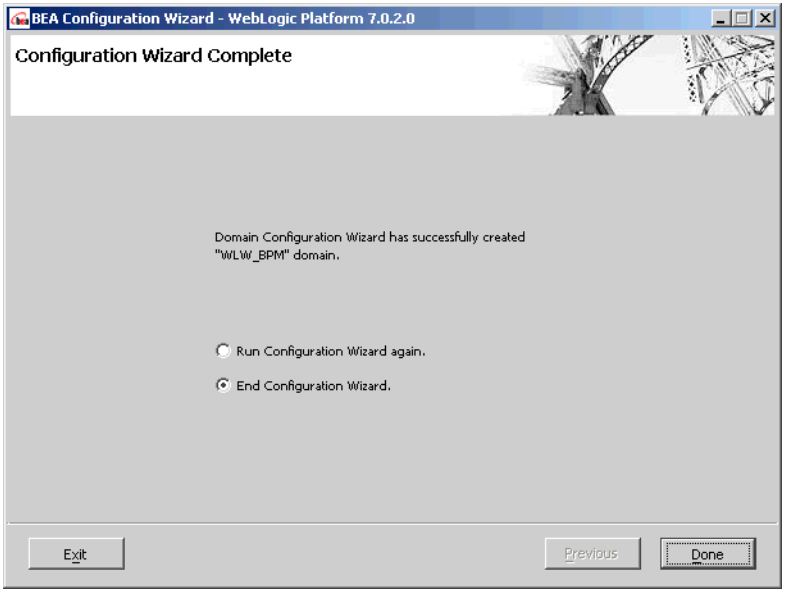
10. In the Confirmation Summary window (Figure 2-5), click Create. The domain is created.

Figure 2-5 Confirmation Summary



11. In the Configuration Wizard Complete window (Figure 2-6), make sure the End Configuration radio button is selected, and then click Done.

Figure 2-6 Configuration Wizard Complete



Step 2: Configure WebLogic Server

1. Start WebLogic Server by running:

   ```
   BEA_HOME\user_projects\WLW_BPM\startWebLogic.cmd
   ```

   The server begins booting and the Pointbase database is created. (Database creation only occurs the first time you start the server in the new domain.)

2. After the database is created, the command line asks for your username and password. Enter the username and password you created in the Domain Configuration Wizard.

3. Later in the booting process, when the command line asks if you want to create credit card encryption/decryption keys, type "y".

   You should wait until you see a line containing **<Server started in RUNNING mode>** before moving on to the next step.

   Note: If you previously created the JMS Response Queue, go to "Step 3: Set Up WebLogic Integration Studio" on page 2-10.

4. After the command line indicates that the server is running, create the JMS Response Queue, as follows:

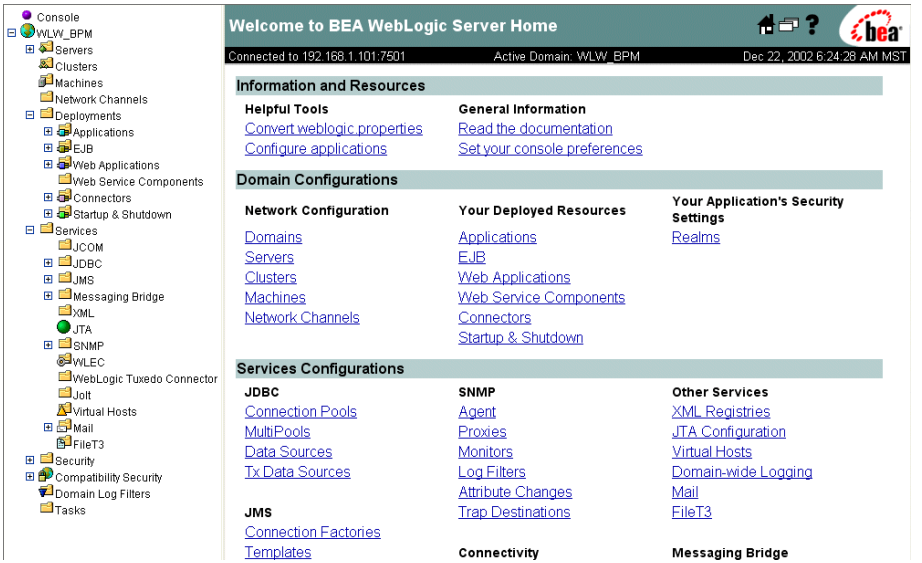
   a. Open a browser and enter the following URL:

      ```
      http://localhost:7501/console
      ```

      The WebLogic Server Administration console opens.

   b. Enter the username and password you created in the Domain Configuration Wizard, and then click Sign In. The WebLogic Server Home page is displayed, as shown in Figure 2-7.

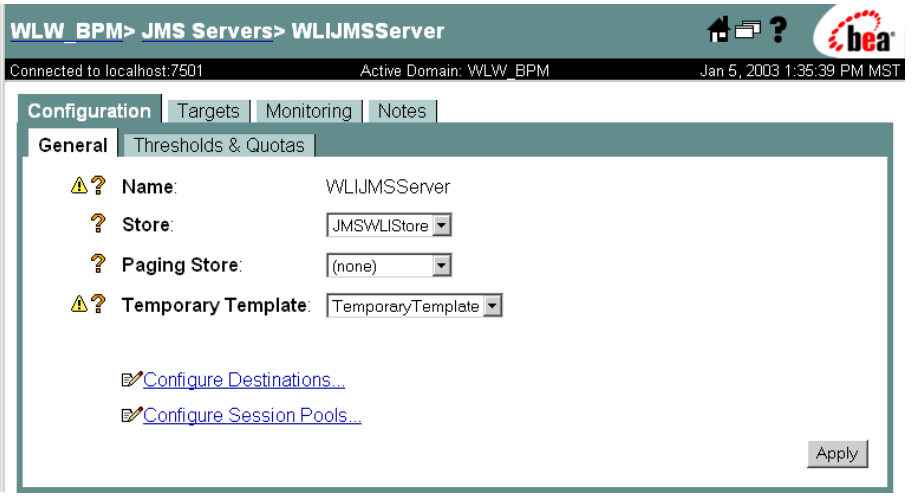
Figure 2-7 WebLogic Server Administration Console



   c. In the left pane, open the WLW_BPM→Services→JMS→Servers node.

   d. Click WLIJMSServer. The WLIJMSServer page is displayed, as shown in Figure 2-8.

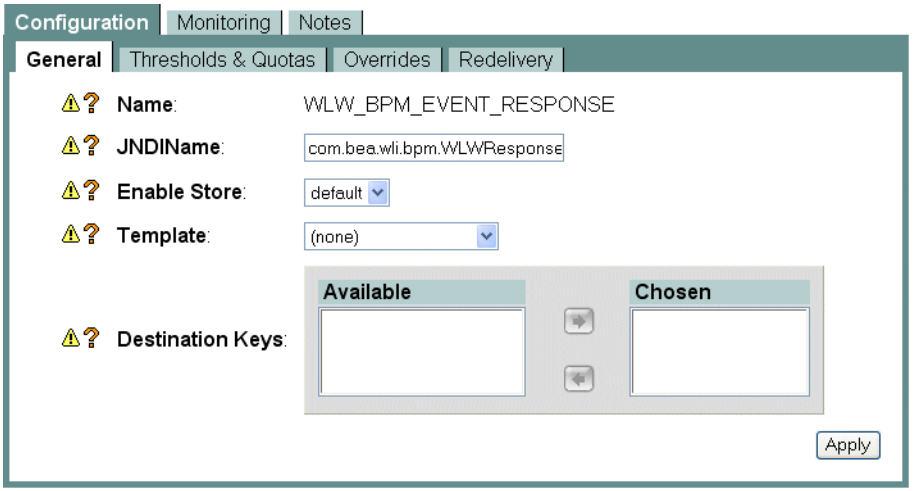
Figure 2-8 WLIJMSServer



e. Click Configure Destinations. The JMS Destinations page is displayed.

f. Click Configure a new JMSQueue. The Create a new JMSQueue page is displayed, as shown in Figure 2-9.

Figure 2-9 Create a New JMSQueue



g. In the Name field, enter "`WLW_BPM_EVENT_RESPONSE`".

h. In the JNDI name field, enter "`com.bea.wli.bpm.WLWResponseQueue`".

i. Click Create. The new destination appears in the tree under the `Servers\WLIJMSSERVER\Destinations` directory.

Step 3: Set Up WebLogic Integration Studio

1. Start the Studio by choosing Start—Programs—BEA WebLogic Platform 7.0—WebLogic Integration 7.0—Studio.
2. Login to the Studio using these parameters: (See Figure 2-10.)
   - User Name: wlisystem
   - Password: wlisystem
   - Server URL: t3://localhost:7501

   Note: Be certain your port number is the same one you chose in the Domain Wizard.

Figure 2-10 Figure 4-1: Logon to WebLogic Integration Studio

3. Create an Organization named BPMWLW:

   Note: If the you have already created BPMWLW organization from running the either the SOAP-HTTP or XML-HTTP sample, delete all the template definitions before proceeding on to the next step. To delete template definitions, in the Organization tree, right-click each of the template definitions and choose Delete from the pop-up menu and go to step 4.

a. From the Studio menu, choose Configuration→Organizations. The Define Organizations window opens.

b. Click Add and create an organization named "BPMWLW".

c. After the Define Organization window appears, select BPMWLW organization, and then click Close.

4. In the Studio, select BPMWLW from the Organization drop-list.

5. Import the BPM_WLW workflows:

   a. From the Studio menu, choose Tools→Import Package.

   b. If setting up the SOAP over HTTP sample, import the `bpmwlwsoaphttp.jar` from the *`BEA_HOME`*`/weblogic700/samples/integration/samples/BPM-WLW/workflows` directory.

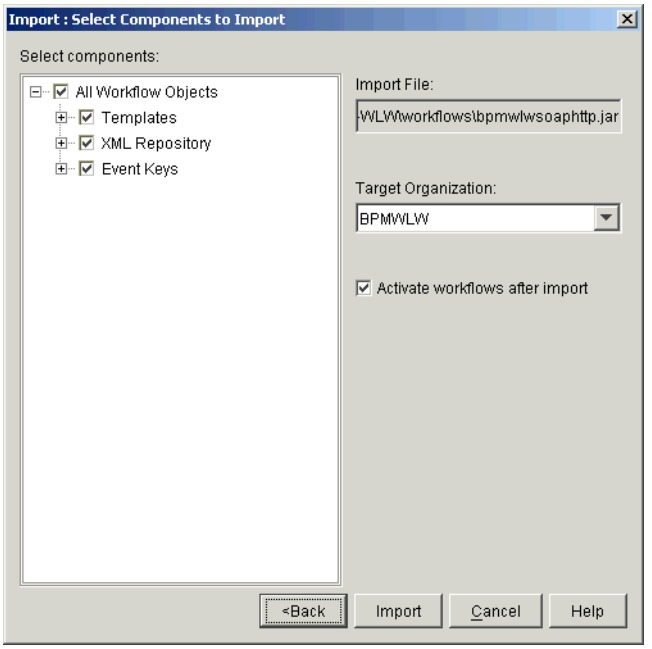
      If setting up the XML over HTTP sample, import the `bpmwlwxmlhttp.jar` file from the *`BEA_HOME`*`/weblogic700/samples/integration/samples/BPM-WLW/workflows` directory.

      In the preceding instructions, *`BEA_HOME`* represents the WebLogic Platform home directory.

   c. In the Import: Select Components to Import window, make sure all the components are selected and that the Activate workflows after import check box is checked, and then select Import.

   Note: If a message appears asking if you want to overwrite existing template definitions and event keys, click Yes to All. This ensures that template definitions from an earlier run of the samples is overwritten by the new one.

Figure 2-11 Select Components to Import



d. In the Import Summary window, click Close. The templates are now available in the Organization tree.

Step 4: Set up WebLogic Workshop

1. Unzip the sample jar files as follows:

   a. SOAP over HTTP sample – Unzip the *BEA_HOME*\weblogic700\samples\integration\samples\BPM-WLW\lib\wlw_bpm_soap.jar file to the BEA_HOME\user_projects\WLW_BPM\applications directory.

   This creates the wlw_bpm_soap directory, which contains the BPM2WLW.jws and WLW2BPM.jws files.

   b. XML over HTTP sample – Unzip the *BEA_HOME*\weblogic700\samples\integration\samples\BPM-WLW\lib\wlw_bpm_xml.jar file to the BEA_HOME\user_projects\WLW_BPM\applications directory.

This creates the `wlw_bpm_xml` directory, which contains the `BPM2WLW.jws` and `WLW2BPM.jws` files.

2. Start WebLogic Workshop by choosing Start→Programs→BEA WebLogic Platform 7.0→WebLogic Workshop→WebLogic Workshop. The Workshop window is displayed.

   Note: If you have already run a sample, Workshop will automatically start the project and server. From the File menu, select Close Project. Go to step 5.

3. From the Workshop menu, choose Tools→Preferences→Paths and set the following.

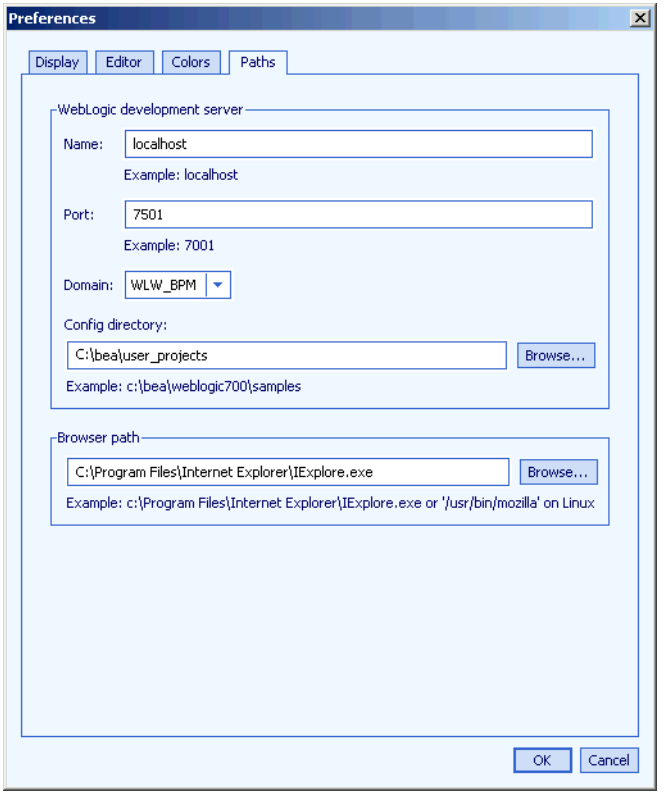
   a. Set the Config directory to "`BEA_HOME\user_projects`".

   b. Set the other parameters as follows:

   - Name: `localhost`
   - Port: `7501`
   - Domain: `WLW_BPM`

   c. Click OK.

Figure 2-12 Setting the Path information



4. Click OK. The Preferences window closes.
5. From the Workshop menu, choose File→Open Project.
6. In the Open Project window, select `wlw_bpm_soap` or `wlw_bpm_xml` project, and then click Open.

Step 5: Launch Web Services

1. Launch the Send PO Web service (`BPM2WLW.jws`):
   a. In the Workshop Project Tree, open the `bpm2wlw` folder.

b. Double-click the `BPM2WLW.jws` node. The Send PO Web service appears in the right pane, as shown in Figure 2-13 (SOAP over HTTP) or Figure 2-14 (XML over HTTP).

Figure 2-13 Send PO Web Service—SOAP Over HTTP

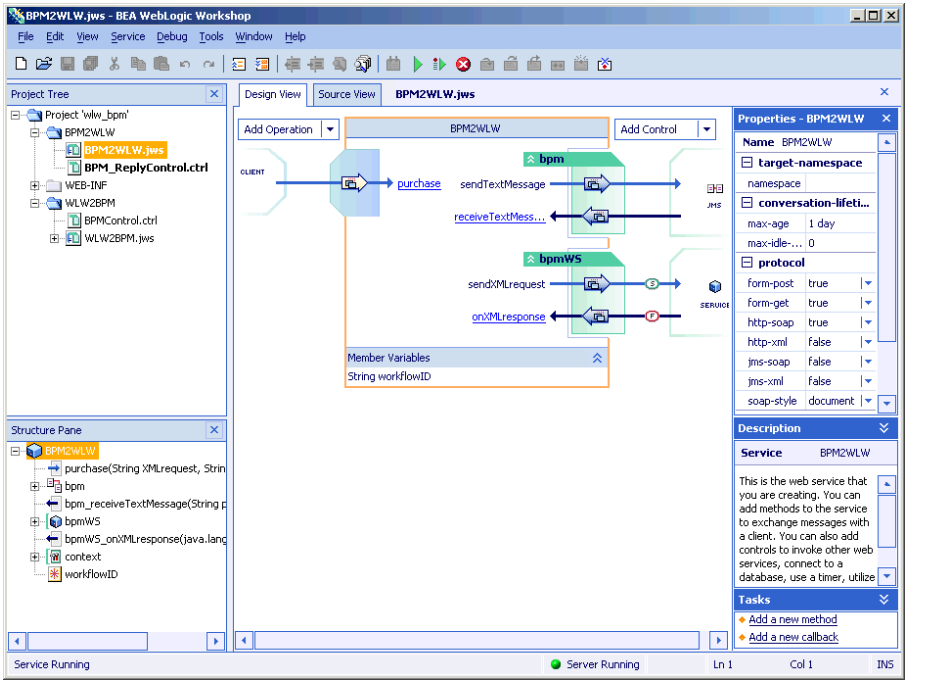
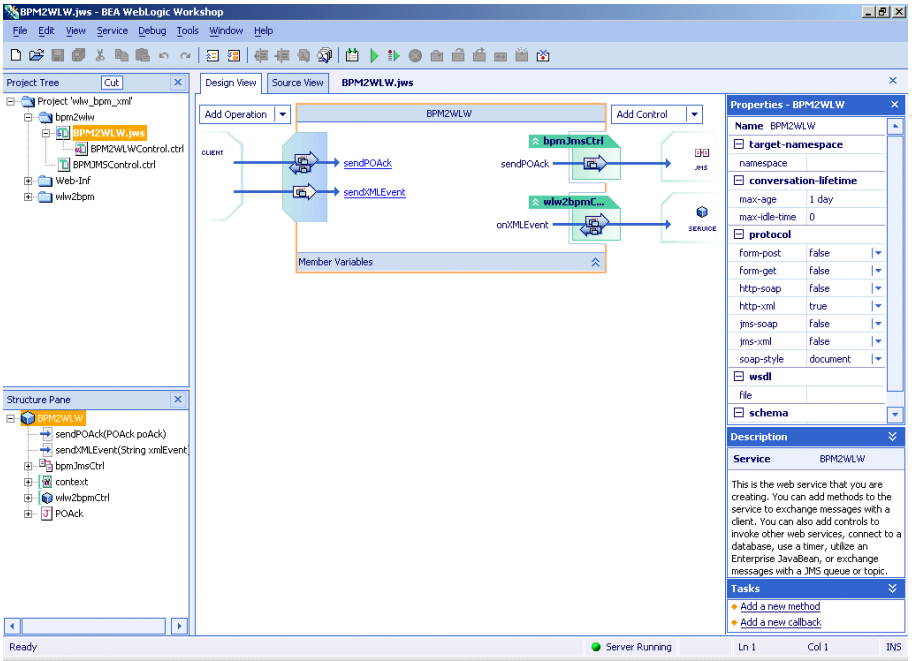


Figure 2-14 Send PO Web Service—XML Over HTTP



c. From the Debug menu, select Start. The lower-left corner of Workshop displays the Build Started message. After the build is complete, the message changes to Service Running. Next a browser window opens to the Test Form for the Send PO Web service (`BPM2WLW.jws`), as shown in Figure 2-15 (SOAP over HTTP) or Figure 2-16 (XML over HTTP).

Figure 2-15 Send PO Web Service—SOAP over HTTP

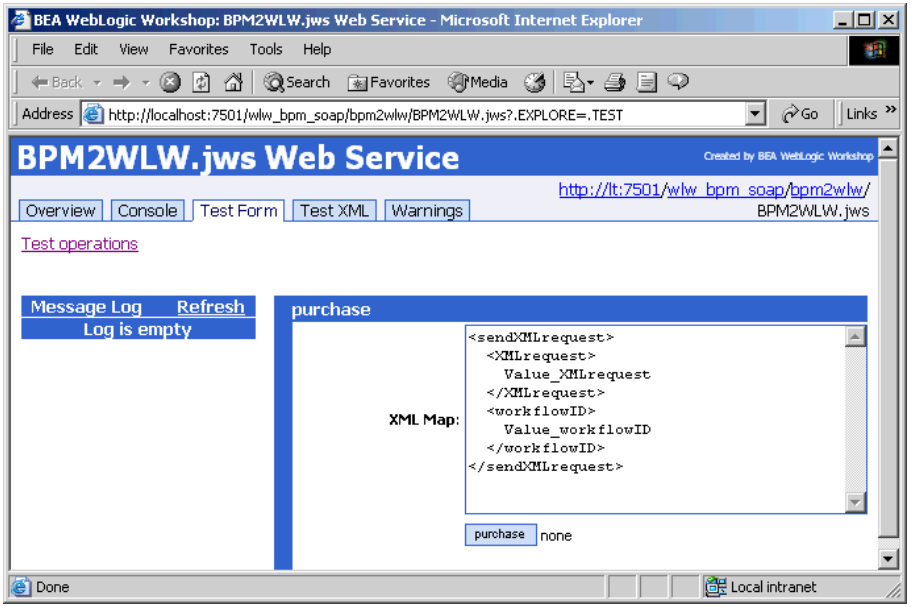
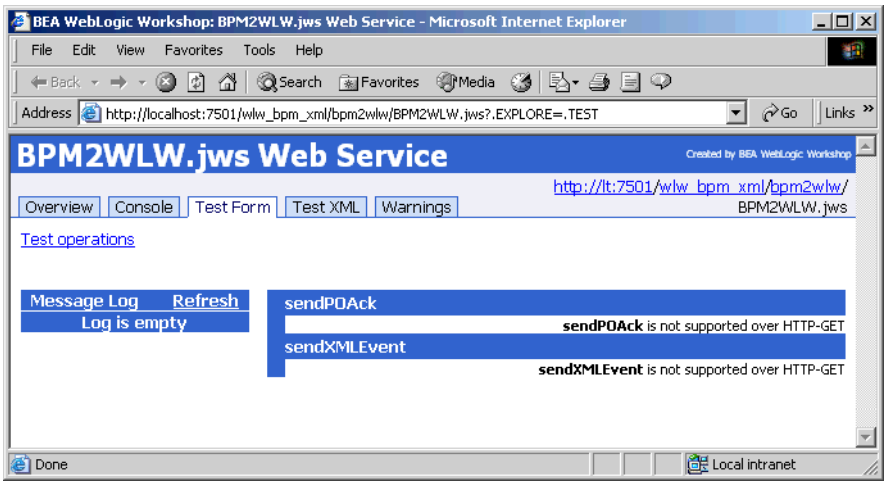


Figure 2-16 Send PO Web Service—XML over HTTP



d. In the Workshop Project Tree, open the `wlw2bpm` folder, and double-click the `WLW2BPM.jws` node. The Process PO Web service appears in the right pane, as shown in Figure 2-17 (SOAP over HTTP) or Figure 2-18 (XML over HTTP).

Note: If the Process PO Web service (`WLW2BPM.jws`) is not already running, go to the Debug menu and select Start. The lower-left corner of Workshop displays a Build Started message. After the build is complete, the message changes to "Service Running." Next a browser window opens to the Test Form for the Process PO Web service, as shown in Figure 2-19 (SOAP over HTTP) or Figure 2-20 (XML over HTTP).

Figure 2-17 Process PO Web Service—SOAP Over HTTP

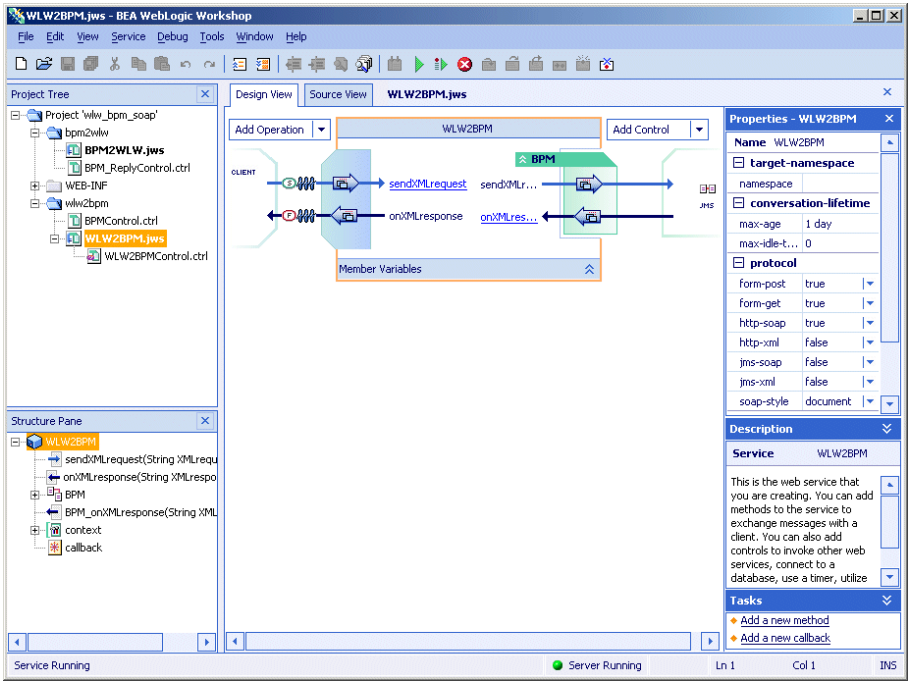
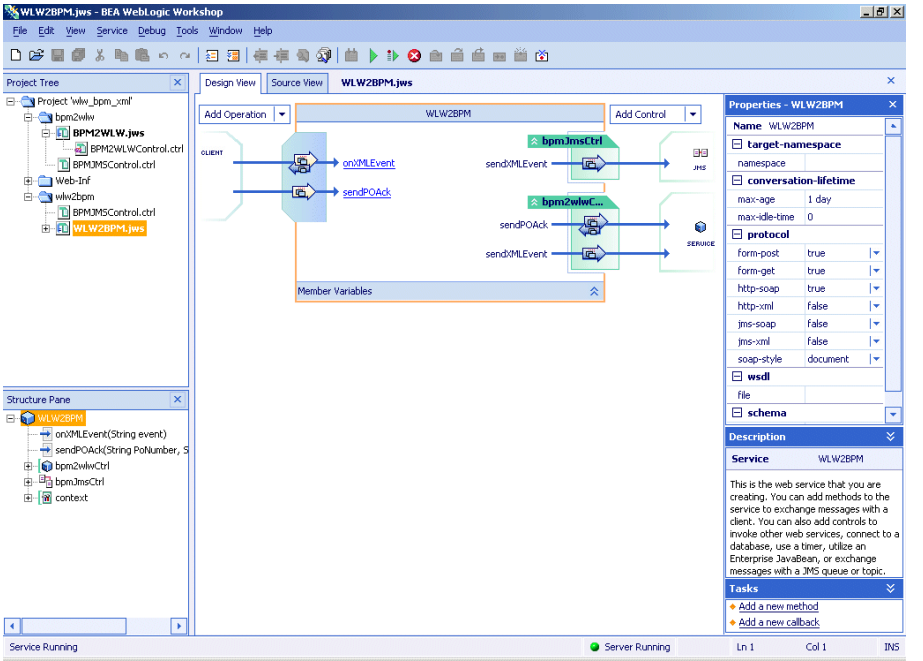


Figure 2-18 Process PO Web Service—XML Over HTTP



e. To view the Test Form for the Process PO Web service (`WLW2BPM.jws`), enter the following URLs in your browser:

- SOAP over HTTP:
  `http://localhost:7501/wlw_bpm_soap/wlw2bpm/WLW2BPM.jws?.EXPLORE=.TEST`
- XML over HTTP:
  `http://localhost:7501/wlw_bpm_xml/wlw2bpm/WLW2BPM.jws?.EXPLORE=.TEST`

  Figure 2-19 shows the SOAP over HTTP Test Form and Figure 2-20 XML over HTTP Test Form.

Figure 2-19 Process PO Web Service—SOAP over HTTP

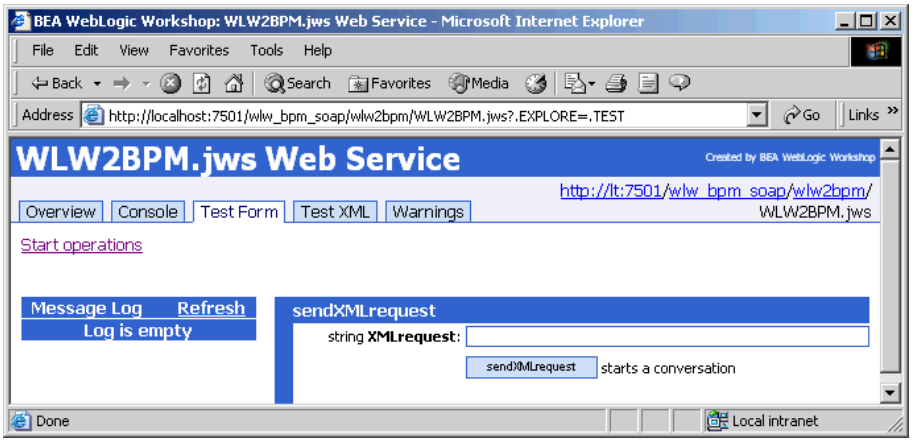
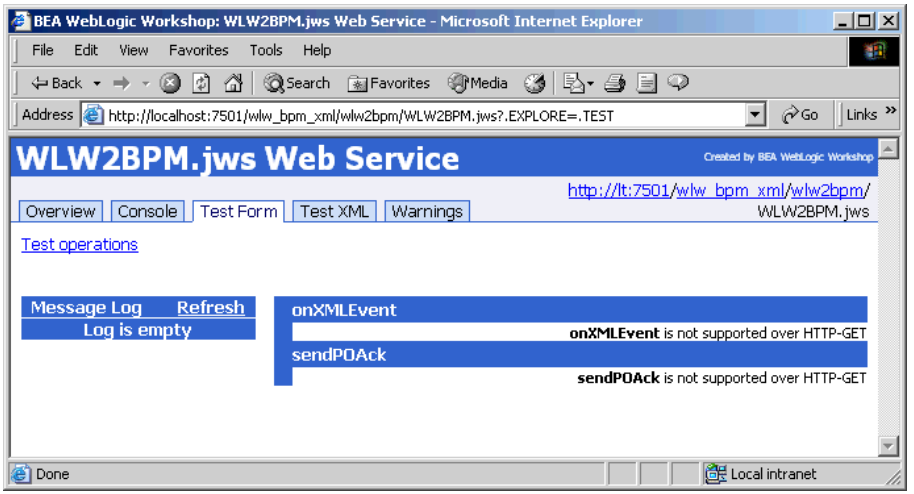


Figure 2-20 Process PO Web Service—XML over HTTP



Note: For more information about WebLogic Workshop, see the following in the WebLogic Workshop online help:

- *Guide to Building Web Services*—Controls: Using Resources from a Web Service
- *Guide to Building Web Services*—Controls: Using Resources from a Web Service—Service Control: Using Another Web Service

- *Guide to Building Web Services*→Maintaining State with Conversations
- *Guide to Building Web Services*→Using Asynchrony to Enable Long-Running Operations→Using Callbacks to Notify Clients of Events
- *WebLogic Workshop Reference*→Class Reference→JwsContext Interface

Step 6: Set Up the WebLogic Integration Swing Worklist

1. Start the Swing Worklist client by running:

   ```
   BEA_HOME\weblogic700\integration\bin\worklist_swing.cmd
   ```

2. Log on to the Worklist with the following parameters:

   - User Name: wlisystem
   - Password: wlisystem
   - Server URL: t3://localhost:7501

   The Worklist opens with the following message: "Your worklist contains no pending tasks."

3. Close the message by clicking OK.

   This completes the setup of the sample.

4. Go to Chapter 3, "BPM-Workshop Interoperability Process," to complete the demonstration of the sample.

3 BPM-Workshop Interoperability Process

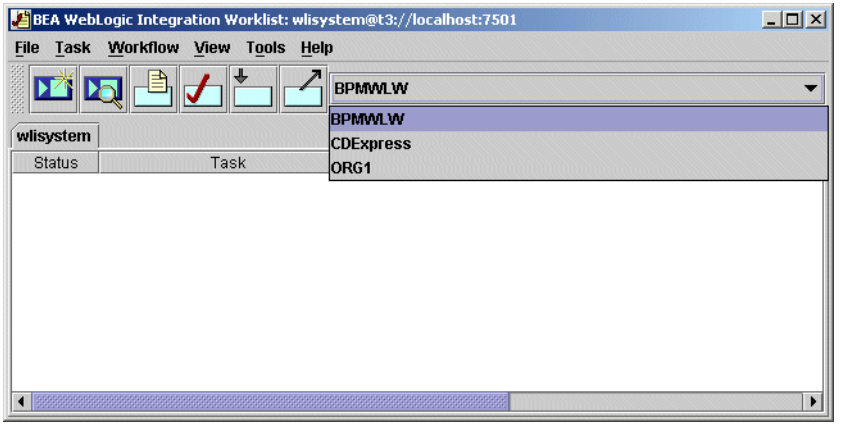
The section contains the steps necessary to complete the interoperability process and a description of messages exchanged in the WebLogic Integration BPM – Workshop Interoperability Sample.

Running the Sample

To run the sample, take the following steps:

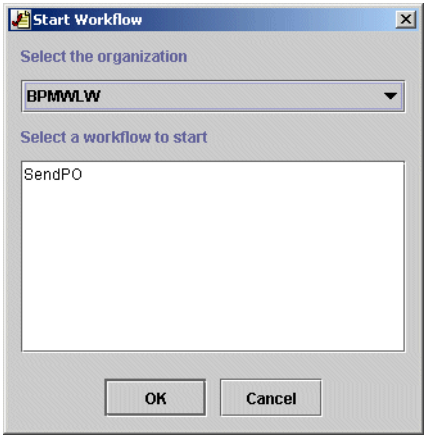
1. In the Worklist, select BPMWLW from the Organization drop-list, as shown in Figure 3-1.

Figure 3-1 Selecting the Organization in the Worklist



2. From the Workflow menu, select Start a Workflow. The Start Workflow window opens, as shown in Figure 3-2.

Figure 3-2 Starting Workflow



3. Make sure that the selected organization is the BPMWLW.
4. Select the SendPO workflow and click OK. A message box appears indicating that the workflow has started.
5. In the message box, select OK.

   After you start the SendPO workflow in the Worklist, the workflow sends a PO XML message to the `jws.queue` JMS queue with one of the following URIs:

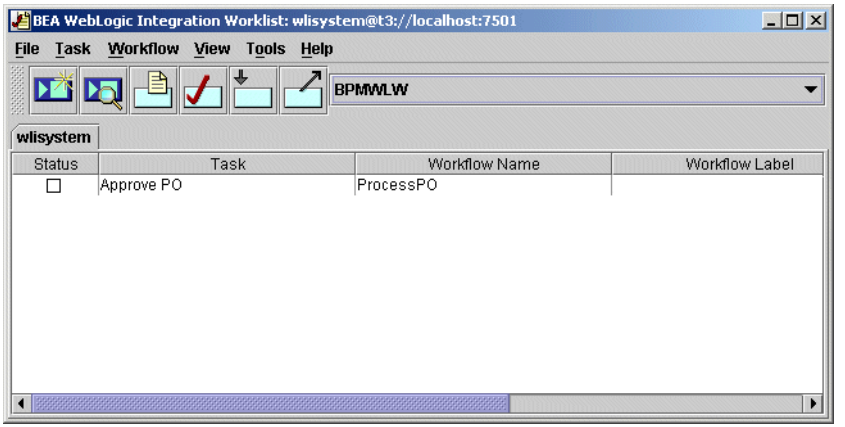
- SOAP over HTTP: `wlw_bpm_soap/bpm2wlw/BPM2WLW.jws`.
- XML over HTTP: `wlw_bpm_xml/bpm2wlw/BPM2WLW.jws`

In the SOAP over HTTP sample, the SendPO Web service receives the PO, and then fires its purchase method. The purchase method calls the `WLW2BPMControl.sendXMLrequest` method, which in turn calls the `BPMControl.sendXMLrequest` method. The `BPMControl.sendXMLEvent` method call places the PO on the `com.bea.wli.bpm.EventQueue` JMS queue.

In the XML over HTTP sample, the SendPO Web service receives the PO, and then fires its `sendXMLEvent` method. The `BPM2WLW.sendXMLEvent` method calls the `WLW2BPMControl.onXMLEvent` method, which in turn calls the `BPMControl.sendXMLEvent` method. The `BPMControl.sendXMLEvent` method call places the PO on the `com.bea.wli.bpm.EventQueue` JMS queue.

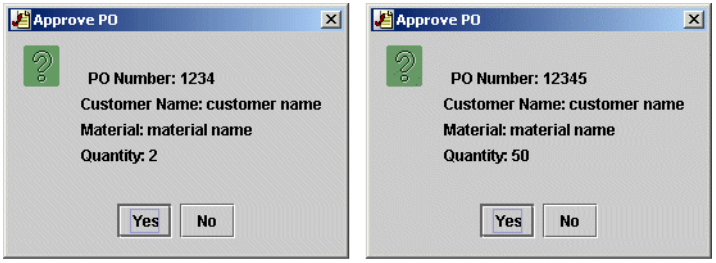
Next, the ProcessPO workflow Start node retrieves the PO from the queue and extracts its contents into variables. The ProcessPO workflow Approve PO node assigns a task to the wlisystem user, and waits for the user to execute the task in the Worklist, as shown in Figure 3-3.

Figure 3-3 Approve PO Workflow Node Assigns the User a Task



6. To execute the task, double-click the Approve PO task. The Approve PO node opens a window asking you to approve the PO, as shown in Figure 3-4.

Figure 3-4 Approve PO Window



SOAP over HTTP XML over HTTP

7. Click Yes to approve the task. The callback action for the question causes the Approve PO node to be marked as done.

   In the SOAP over HTTP sample, the following takes place:

   The workflow proceeds to the Ack PO node, which sends a POAck XML message to the `com.bea.wli.bpm.WLWResponseQueue`. The `BPMControl` JMS control in the WLW2BPM Web service retrieves the POAck message from the queue and triggers the `BPM_onXMLresponse` callback handler in the WLW2BPM Web service. The callback handler invokes the `onXMLresponse` handler, which posts the POAck message to the `com.bea.wli.bpm.EventQueue` JMS queue.

   In the SendPO workflow, the Get PO Ack node retrieves the POAck message and allows the execution to proceed to the Notify PO Ack node, which assigns the Notify PO Ack task to the Worklist, as shown in Figure 3-5.

   In the XML over HTTP sample, the following takes place:

   The workflow proceeds to the Ack PO node, which sends a POAck XML message to the `jws.queue`. Note that the POAck XML message tags the instance ID of the SendPO workflow into the payload of the message. The message also carries the header "URI" which points to `wlw_bpm_xml/wlw2bpm/WLW2BPM.jws`. This ensures the WLW2BPM Web service is invoked by the POAck XML message.

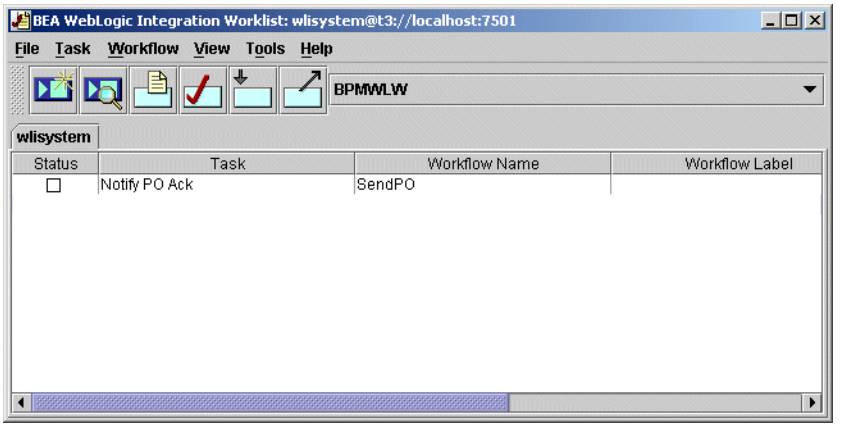
   The WLW2BPM Web service retrieves the POAck message from the queue and triggers the `WLW2BPM.sendPOAck` method. The `WLW2BPM.sendPOAcl` method calls the BPM2WLW control's `sendPOAck` method passing in the PO number.

   The `sendPOAck` method transmits the raw XML message over HTTP to the BPM2WLW Web service. The BPM2WLW Web service, on receiving the

acknowledgement, posts the POAck message to the `com.bea.wli.bpm.EventQueue` JMS queue.

The SendPO workflow has an event key that looks for the instance ID in the POAck message payload. When an event with the key-value expression that contains the instance ID arrives on the event queue, the instance triggers the Get PO Ack node. This results in the retrieval of the POAck message and allows the execution to proceed to the Notify PO Ack node, which assigns the Notify PO Ack task to the Worklist, as shown in Figure 3-5.

Figure 3-5 The Notify PO Ack Workflow Node Assigns the User a Task



8. Double-click the Notify PO Ack task to execute the task. The task disappears from the Worklist and execution proceeds to the Stop node. This terminates the sample.

9. To view the messages passed between the Web services, use the following URLs:

   - SOAP over HTTP:

     ```
     http://localhost:7501/wlw_bpm_soap/wlw2bpm/WLW2BPM.jws?.EXPLORE=.TEST
     ```

     ```
     http://localhost:7501/wlw_bpm_soap/bpm2wlw/BPM2WLW.jws?.EXPLORE=.TEST
     ```

   - XML over HTTP:

     ```
     http://localhost:7501/wlw_bpm_xml/wlw2bpm/WLW2BPM.jws?.EXPLORE=.TEST
     ```

     ```
     http://localhost:7501/wlw_bpm_xml/bpm2wlw/BPM2WLW.jws?.EXPLORE=.TEST
     ```

Note: If the Message Log is empty, click Refresh in the Message Log section of the Web services page.

10. To view the messages, click the messages in the Message Log.

    The following figures show the messages:

Figure 3-6 Send PO Web Service Message Log—SOAP over HTTP

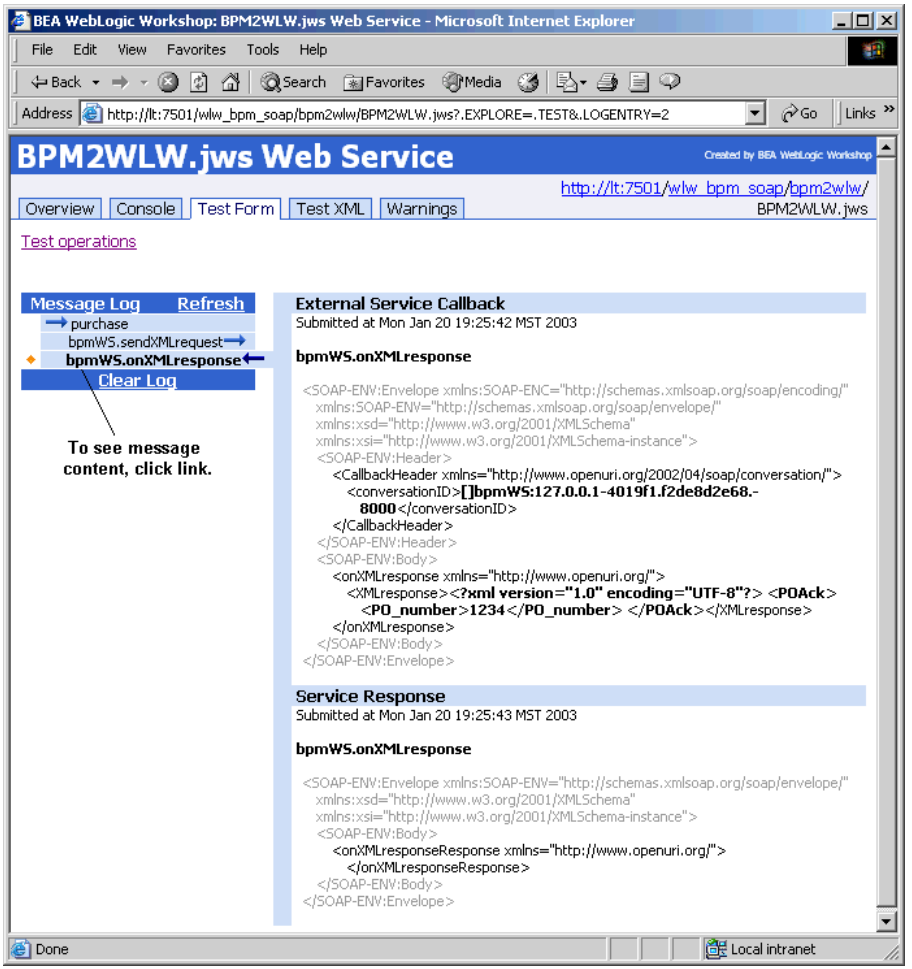


Figure 3-7 Process PO Web Service Message Log—SOAP over HTTP

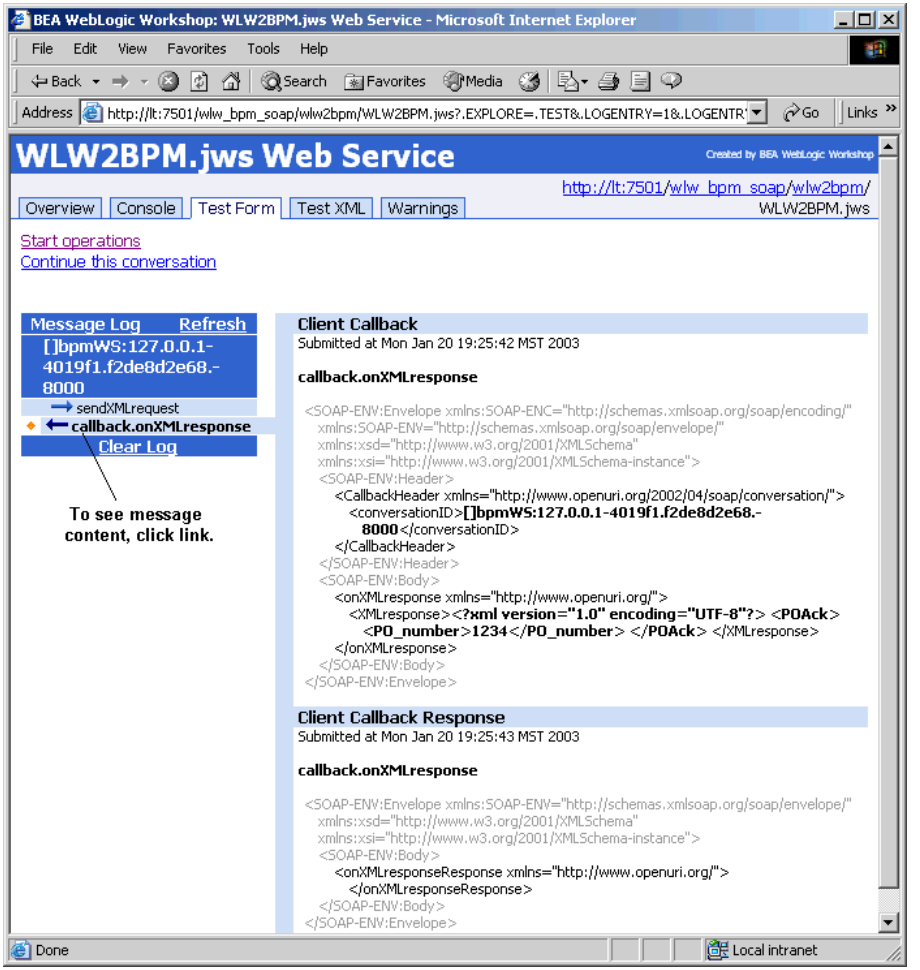


Figure 3-8 Send PO Web Service Message Log—XML over HTTP

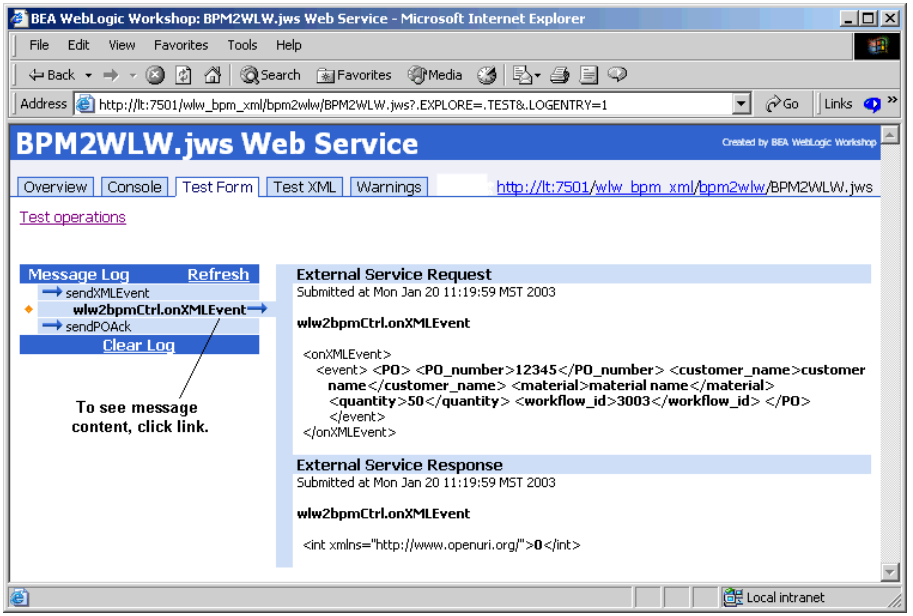
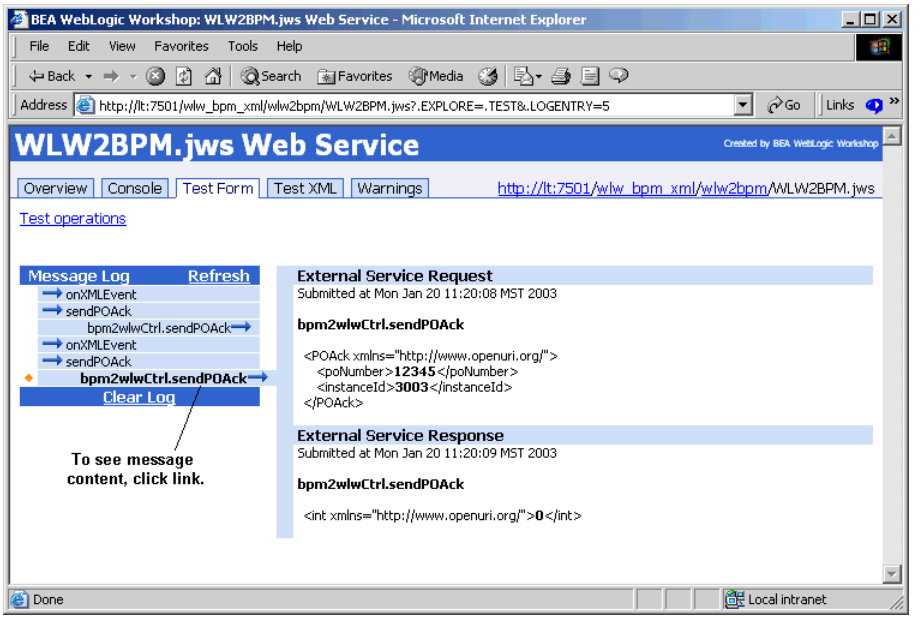


Figure 3-9 Send PO Web Service Message Log—XML over HTTP



Index