



BEA WebLogic XML/Non-XML Translator

A Component of BEA WebLogic Integration

Plug-In Guide

BEA WebLogic XML/Non-XML Translator Release 2.0
Document Edition 2.0
July 2001

Copyright

Copyright © 2001 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, WebLogic, Tuxedo, and Jolt are registered trademarks of BEA Systems, Inc. How Business Becomes E-Business, Operating System for the Internet, Liquid Data, BEA WebLogic E-Business Platform, BEA Builder, BEA Manager, BEA eLink, BEA Campaign Manager for WebLogic, BEA WebLogic Commerce Server, BEA WebLogic Personalization Server, BEA WebLogic Process Integrator, BEA WebLogic Collaborate, BEA WebLogic Enterprise, BEA WebLogic Server, and BEA WebLogic Integration are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective company.

BEA WebLogic XML/Non-XML Translator Installation and Configuration Guide

Document Edition	Part Number	Date	Software Version
2.0	N/A	July 2001	BEA WebLogic XML/Non-XML Translator 2.0

Contents

1. Understanding the XML Translator Plug-In

Understanding XML Translation	1-1
What is XML Translator?.....	1-3
The Design-Time Component.....	1-4
The Run-Time Component.....	1-5
Run-Time Plug-In to WebLogic Process Integrator	1-5
Using the Repository	1-6
XML Translator Plug-In Prerequisites	1-7

2. Using the XML Translator Plug-In

Data Translation with the XML Translator Plug-In	2-2
Translate XML to Binary	2-3
Translate Binary to XML	2-6
Processing Event Data.....	2-8
Enhancing Data Translation Performance.....	2-9
Variable Types and the XML Translator Plug-In.....	2-12
Custom Data Types and the XML Translator Plug-In	2-13
Configuring User Defined Data Types.....	2-13
Using Format Builder.....	2-13
Using the Repository Import Utility	2-15
WebLogic Server Clustering Support	2-16
Configuring the XML Translator Plug-in for Clustering	2-16

3. Running the WebLogic Process Integrator Sample Applications

Prerequisite Considerations	3-1
Running the WebLogic Process Integrator Servlet Sample	3-2

What is Included in the Servlet Sample	3-2
How to Run the Servlet Sample	3-3
Step 1. Configure and Run WebLogic Process Integrator	3-3
Step 2. Deploy the Web Application.....	3-4
Step 3. Configure the Mail Session.....	3-6
Step 4. Create a New Template and Activate the Workflow	3-8
Step 5. Store the SampleData.mfl File in the Repository.....	3-9
Step 6. Generate the XML Data and Send the Message	3-10
Running the WebLogic Process Integrator EJB Sample	3-11
What is Included in the EJB Sample	3-11
How to Run the EJB Sample	3-12
Step 1. Configure and Run WebLogic Process Integrator	3-12
Step 2. Import the Workflow Definition	3-13
Step 3. Open the Template	3-15
Step 4. Start the Workflow	3-18

1 Understanding the XML Translator Plug-In

This guide describes the functionality and operation of the XML Translator Plug-In. The following topics are discussed:

- Understanding the XML Translator Plug-In
- Using the XML Translator Plug-In
- Running the WebLogic Process Integrator Sample Applications

Understanding XML Translation

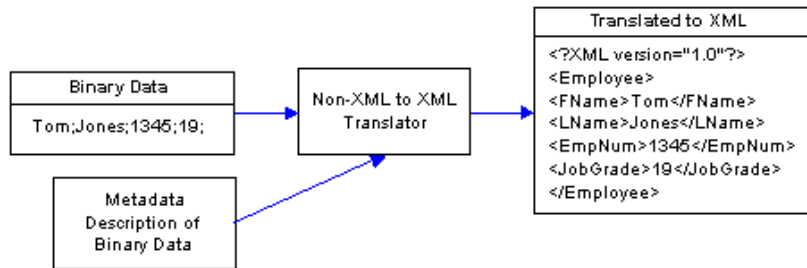
Data that is sent to, or received from, legacy applications is often platform-specific binary data that is in the native machine representation. Binary data is not self-describing, so in order to be understood by an application, the layout of this data (metadata) must be embedded within each application that uses the binary data.

XML is becoming the standard for exchanging information between applications because XML embeds a description of the data within the data stream, thus allowing applications to share data more easily. XML is easily parsed and can represent complex data structures. As a result, the coupling of applications no longer requires metadata to be embedded within each application.

When you translate binary to XML data, you convert structured binary data to an XML document so that the data can be accessed via standard XML parsing methods. You must create the metadata used to perform the conversion. The translation process

converts each field of binary data to XML according to the metadata defined for each field of data. In the metadata you specify the name of the field, the data type, the size, and whether the field is always present or optional. It is this description of the binary data that is used to translate the binary data to XML. Figure 1-1 shows a sample of XML data translation.

Figure 1-1 XML Data Translation of: Tom;Jones;1345;19;



Applications developed on the WebLogic platform often use XML as the standard data format. If you want the data from your legacy system to be accessible to applications on the WebLogic platform, you may use XML Translator to translate it from binary to XML or from XML to binary. If you need the XML in a particular XML dialect for end use, you must transform it using an XML data mapping tool.

What is XML Translator?

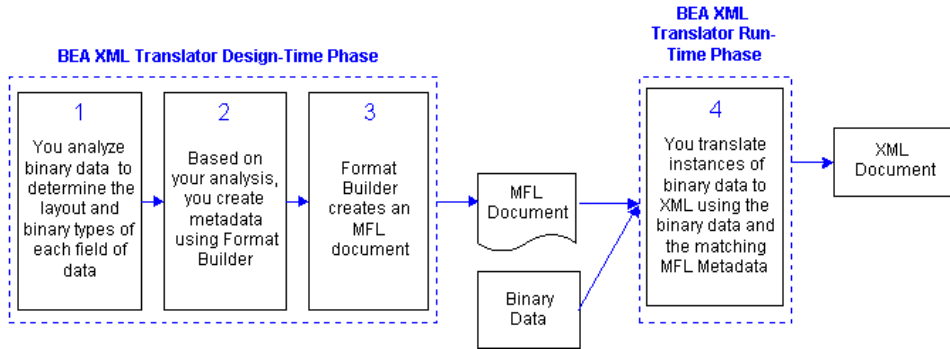
XML Translator, a component of BEA WebLogic Integration, facilitates the integration of data from diverse enterprise applications by supporting data translations between binary formats from legacy systems and XML. XML Translator normalizes legacy data into XML so it may be directly consumed by XML applications, transformed into a specific XML grammar, or used directly to start workflows in WebLogic Process Integrator, another component of BEA WebLogic Integration. XML Translator supports non-XML to XML translation and vice versa and is made up of three primary components:

- The Design-Time Component
- The Run-Time Component
- The Run-Time Plug-In to Process Integrator

To perform a translation, you create a description of your binary data using the design-time component (Format Builder). This involves analyzing binary data so that its record layout is accurately reflected in the metadata you create in Format Builder. You then create a description of the input data in Format Builder and save this metadata as a Message Format Language (MFL) document. XML Translator includes importers that automatically create message format definitions from common sources of binary metadata, such as COBOL copybooks.

You can then use XML Translator's run-time component to translate instances of binary data to XML. Figure 1-2 shows the event flow for non-XML to XML data translation. A Plug-In to Process Integrator allows for easy access to configuring translations.

Figure 1-2 Event Flow for Non-XML to XML Translation Using XML Translator



The Design-Time Component

The design-time component is a Java application called Format Builder. Format Builder is used to create descriptions of binary data records. Format Builder allows you to describe the layout and hierarchy of the binary data so that it can be translated to or from XML. The description you create in Format Builder is saved in an XML grammar called Message Format Language (MFL). MFL documents are metadata used by the run-time component of XML Translator and the plug-in to Process Integrator to translate an instance of a binary data record to an instance of an XML document (or vice-versa). Format Builder will also create a DTD or XML Schema document that describes the XML document created from a translation.

For more information on the design-time component, refer to the *BEA WebLogic XML/Non-XML Translator User Guide*.

The Run-Time Component

The run-time component of XML Translator is a Java class with various methods used to translate data between binary and XML formats. This Java class can be deployed in an EJB using BEA WebLogic Server, invoked as a business operation from a workflow in BEA WebLogic Process Integrator, or integrated into any Java application.

For more information on the run-time component, refer to the *BEA WebLogic XML/Non-XML Translator User Guide*.

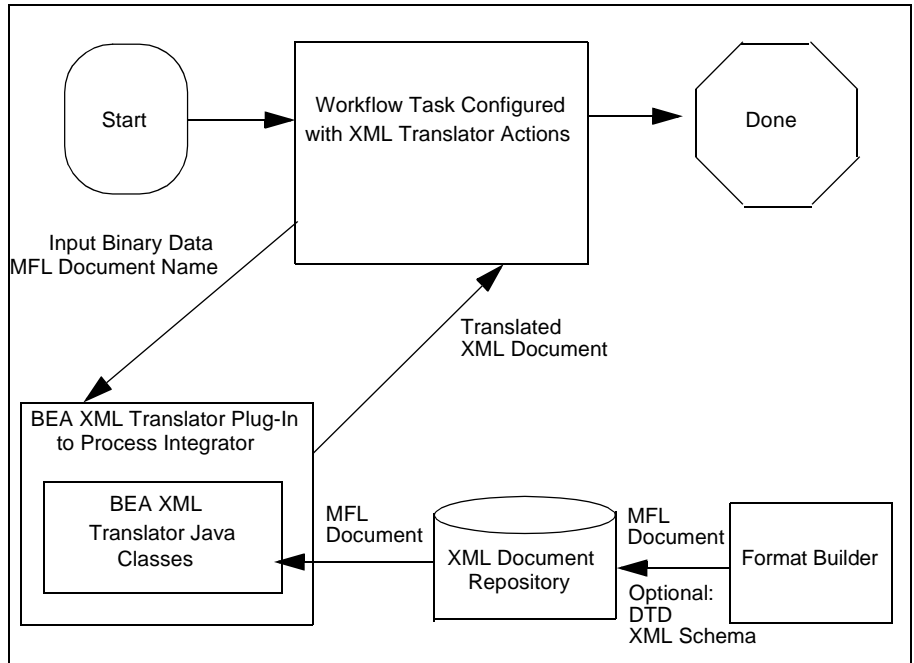
Run-Time Plug-In to WebLogic Process Integrator

BEA XML Translator Plug-In for WebLogic Process Integrator provides for an exchange of information between applications by supporting data translations between binary formats from legacy systems and XML. The XML Translator Plug-In provides Process Integrator actions that allow you to access XML to Binary and Binary to XML translations.

In addition to this data translation capability, the XML Translator Plug-In provides event data processing in binary format, in-memory caching of MFL documents and translation object pooling to boost performance, a `BinaryData` variable type to edit and display binary data, and execution within a WebLogic Server clustered environment.

1 Understanding the XML Translator Plug-In

The following illustration describes the relationship between XML Translator and Process Integrator.



Using the Repository

The XML Translator repository feature provides a centralized document storage mechanism that supports the following four document types:

- MFL - Message Format Language document
- DTD - XML Document Type Definition document
- XSD - XML Schema document
- XSLT - XSLT Stylesheet

The Repository provides access to these document types and allows you to share them between XML Translator, Process Integrator, Application Integration, and Collaborate. The repository also includes a batch import utility that allows previously constructed MFL, DTD, XSD, and XSLT documents to be easily migrated into the repository.

XML Translator Plug-In Prerequisites

Before using the XML Translator Plug-In, you should perform the following tasks:

- Install BEA WebLogic Integration, specifically the WebLogic Process Integrator component, following the instructions in the *BEA WebLogic Process Integrator* documentation.
- Install XML Translator following the instructions in the *BEA WebLogic XML/Non-XML Translator Installation and Configuration Guide*.
- Perform the configuration tasks for the XML Translator Plug-In following the instructions in the *BEA WebLogic XML/Non-XML Translator Installation and Configuration Guide*.

2 Using the XML Translator Plug-In

Within most enterprise application integration (EAI) problem domains, data translation is an inherent part of an EAI solution. XML is quickly becoming the standard for exchanging information between applications, and is invaluable in integrating disparate applications. However, most data transformation engines do not support translations between binary data formats and XML. XML Translator Plug-In for WebLogic Process Integrator provides for an exchange of information between applications by supporting data translations between binary formats from legacy systems and XML.

In addition to this data translation capability, the XML Translator Plug-In provides a binary data event handler, in-memory caching of MFL documents and translation object pooling to boost performance, a `BinaryData` variable type to edit and display binary data, and execution within a WebLogic Server clustered environment.

This section provides information about the following topics:

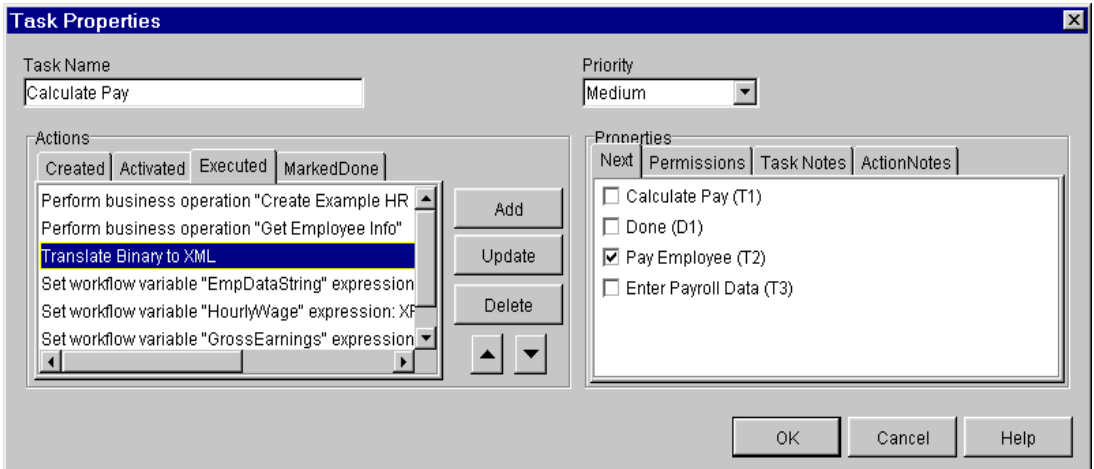
- Data Translation with the XML Translator Plug-In
- Processing Event Data
- Enhancing Data Translation Performance
- Custom Data Types and the XML Translator Plug-In
- WebLogic Server Clustering Support

Data Translation with the XML Translator Plug-In

The XML Translator Plug-In provides XML and non-XML translation capabilities from within WebLogic Process Integrator. To perform one of these translation actions, follow the steps below. For more information on the actions specific to WebLogic Process Integrator, refer to the WebLogic Process Integrator documentation.

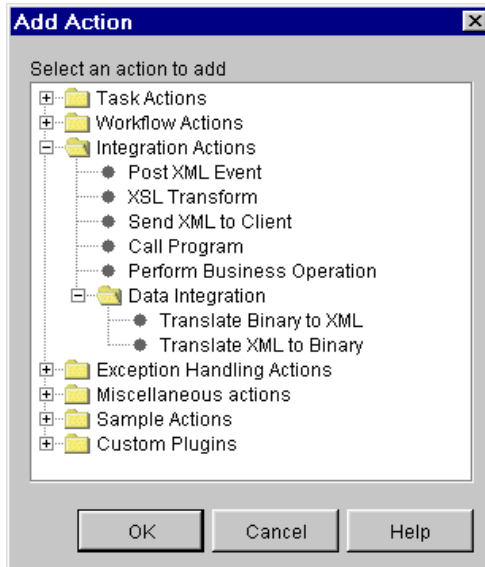
1. Start WebLogic Process Integrator Server and Studio.
2. Open the desired template definition and double-click a task. The Task Properties dialog opens (Figure 2-1).

Figure 2-1 Task Properties Dialog



3. If the task contains the data translation action, select it from the list and click Update; then, proceed to step 4. Otherwise, click Add to add a new action. The Add Action dialog opens (Figure 2-2).

Figure 2-2 Add Action dialog



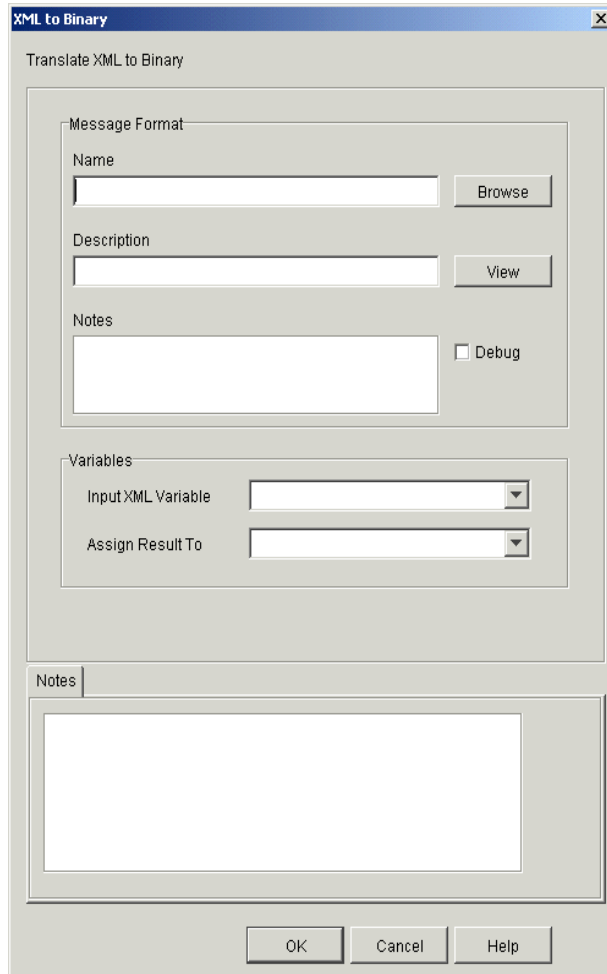
4. Select Integration Actions to expand its action list, then select Data Integration and choose the action you want to perform ([Translate XML to Binary](#) or [Translate Binary to XML](#)).

Translate XML to Binary

To perform an XML to binary translation:

1. From the Add Action dialog (Figure 2-2), choose XML to Binary Translation. The Translate XML to Binary dialog opens (Figure 2-3).

Figure 2-3 Translate XML to Binary Dialog



2. Enter data in the fields as described in the following table.

Field	Description
Message Format Parameters	
Name	The name of the message format. You can type a name directly in the text box, or click Browse to select the document from the repository.

Field	Description
Description	Displays the description of the message format. Note: This field is display only. You cannot edit this field.
Notes	Displays the notes attached to the message format. Note: This field is display only. You cannot edit this field.
Debug	Enable or disable debug messaging. When you select this option, the translation actions are written to the WebLogic Server log file.

Message Format Action Buttons

Browse	Allows you to browse MFL documents in the repository. Refer to “Retrieving and Storing Repository Documents” in the <i>BEA WebLogic XML/Non-XML Translator User Guide</i> for specific instructions.
View	Displays the items contained in the message format so you can verify that you have selected the correct document type for translation.

Variable Parameters

Input XML Variable	Displays the XML workflow variables. Select the variable you want to use in the translation, or create a new variable as follows: <ol style="list-style-type: none">1. Type the name you want to assign to the new variable and click OK. A confirmation message box displays.2. Click Yes to create the new variable.
Assign Result To	Displays the Binary Data workflow variables. Select the variable you want to use to store the translated information, or create a new variable as follows: <ol style="list-style-type: none">1. Type the name you want to assign to the new variable and click OK.2. Click Yes to create the new variable.

3. Click OK to save the translation information to the workflow.

Translate Binary to XML

To perform a binary to XML translation:

1. From the Add Action dialog (Figure 2-2), choose Integration Actions→Data Integration→Translate Binary to XML. The Translate Binary to XML dialog opens (Figure 2-4).

Figure 2-4 Translate Binary to XML Dialog

The dialog box is titled "Binary to XML" and contains the following elements:

- Message Format:**
 - Name:** A text input field with a "Browse" button to its right.
 - Description:** A text input field with a "View" button to its right.
 - Notes:** A large text area with a "Debug" checkbox to its right.
- Variables:**
 - Input Binary Variable:** A dropdown menu.
 - Assign Result To:** A dropdown menu.
- Notes:** A large text area at the bottom of the main content area.
- Buttons:** "OK", "Cancel", and "Help" buttons are located at the bottom right of the dialog.

2. Enter data in the fields as described in the following table.

Field	Description
Message Format Parameters	
Name	The name of the message format. You can type a name directly in the text box, or click Browse to select the message format from the repository.
Description	Displays the description of the message format. Note: This field is display only. You cannot edit this field.
Notes	Displays the notes attached to the message format. Note: This field is display only. You cannot edit this field.
Debug	Enable or disable debug messaging. When you select this option, the translation actions are written to the WebLogic Server log file.
Message Format Action Buttons	
Browse	Allows you to browse MFL documents in the repository. Refer to “Retrieving and Storing Repository Documents” in the <i>BEA WebLogic XML/Non-XML Translator User Guide</i> for specific instructions.
View	Displays the items contained in the message format so you can verify that you have selected the correct document type for translation.
Variable Parameters	
Input Binary Variable	Displays the binary workflow variables. Select the variable you want to use in the translation, or create a new variable as follows: <ol style="list-style-type: none"> 1. Type the name you want to assign to the new variable and click OK. A confirmation message box displays. 2. Click Yes to create the new variable.
Assign Result To	Displays the XML Data workflow variables. Select the variable you want to use to store the translated information, or create a new variable as follows: <ol style="list-style-type: none"> 1. Type the name you want to assign to the new variable and click OK. 2. Click Yes to create the new variable.

3. Click OK to save the translation information to the workflow.

Processing Event Data

The XML Translator Plug-In provides functionality that allows binary data to trigger WebLogic Process Integrator workflows by converting the binary data to XML, or pre-processing it at the front end of WebLogic Process Integrator event processing. This functionality is referred to as the “event handler.” Publishing JMS messages to a topic causes the event handler to run.

There are three JMS properties required for the message to be pre-processed by the XML Translator Plug-In:

- `WLPIContentType`: "binary/x-application/wlxt"
- `WLPIPlugin`: "com.bea.wlxt.WLXTPlugin"
- `WLPIEventDescriptor`: MFL document name

The first two JMS message properties are constant for all messages addressed to the event handler. The third property contains the name of the MFL document that describes the binary data in the message.

Note: This MFL document must be stored in the repository.

Listing 2-1 is a sample of the code used to build a message that is to be processed by the XML Translator event handler.

Listing 2-1 Sample Event Handler Code

```
byte[] bindata = ... the binary data ...
pub = sess.createPublisher(topic);
BytesMessage msg = sess.createBytesMessage();
msg.writeBytes(bindata);
msg.setStringProperty("WLPIPlugin", "com.bea.wlxt.WLXTPlugin");
msg.setStringProperty("WLPIContentType",
    "binary/x-application/wlxt");
msg.setStringProperty("WLPIEventDescriptor", "mymfldoc");
pub.publish(msg);
```

This process is illustrated in the servlet sample application (see [Running the WebLogic Process Integrator Servlet Sample](#)).

Enhancing Data Translation Performance

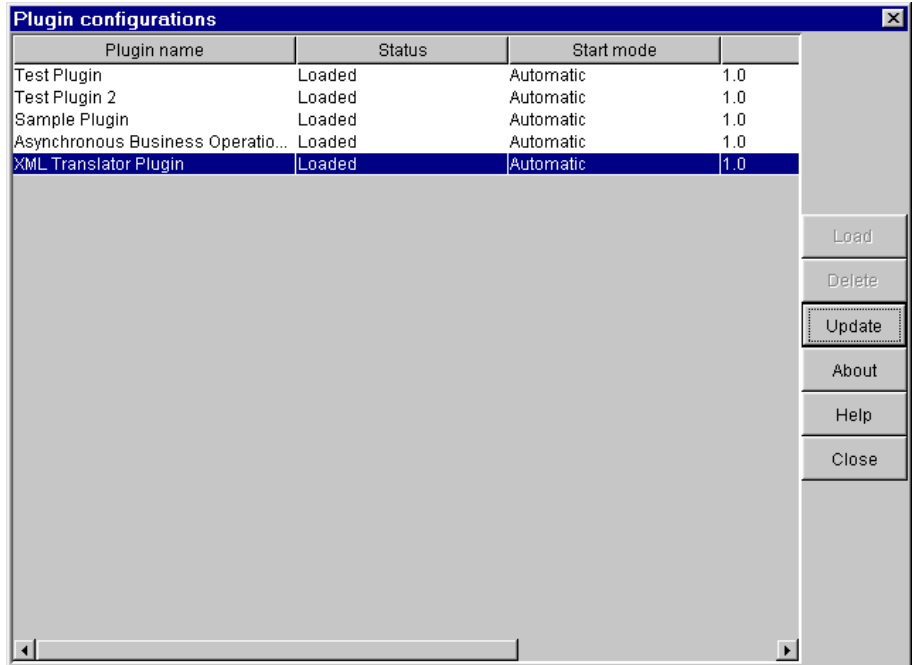
The XML Translator Plug-In provides a configuration panel to administer and monitor the MFL document in-memory cache and enable or disable event handler debugging. Using this panel, you can adjust the in-memory cache and translation object pool to enhance the performance of your data translations.

Note: You must clear the MFL document in-memory cache in order for any updates you make to an MFL document to take effect.

To access the configuration panel, follow the steps below. For more information on the actions specific to WebLogic Process Integrator, refer to the WebLogic Process Integrator documentation.

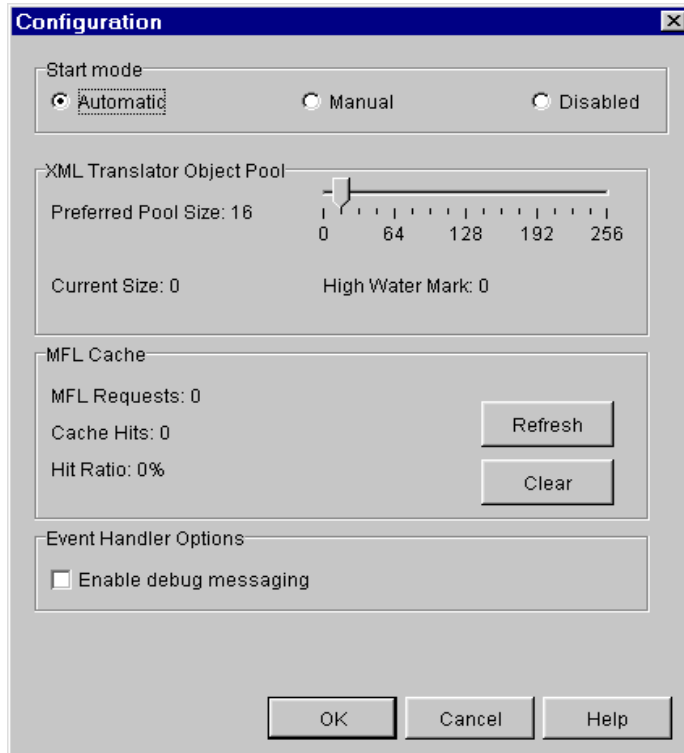
1. Start WebLogic Process Integrator Studio.
2. Choose Configuration→Plugins. The Plugin Configuration dialog opens (Figure 2-5).

Figure 2-5 Plugin Configuration Dialog



3. Choose XML Translator Plug-in and click Update. The Configuration dialog for the XML Translator Plug-in opens (Figure 2-6).

Figure 2-6 Configuration Dialog for XML Translator Plug-In



4. Use the fields as described in the table below to monitor and enhance translation performance.

Field	Description
XML Translator Object Pool	
Preferred Pool Size	Defines the maximum number of permanent objects in the pool. Use the slider to set the pool size to the desired number. Note: The translation engine creates temporary pool objects if the demand exceeds the preferred pool size you have set. These objects are deleted when they are returned to the pool.
Current Size	Displays the number of objects currently in the pool.

Field	Description
High Water Mark	Displays the largest number of objects in the pool since the server was started.
MFL Cache	
MFL Requests	Displays the total number of requests for translation of MFL documents.
Cache Hits	Displays the number of requests where the MFL document needed was already in the cache.
Hit Ratio	Displays the percentage of requests satisfied by retrieving MFL documents from the cache, rather than from the database.
MFL Cache Action Buttons	
Refresh	Sends a request to the server to update the MFL cache statistics.
Clear	Clears the MFL document cache. This requires all future translation requests to load MFL documents from the repository.
Event Handler Options	
Enable Debug Messaging	Enables or disables debug messaging for the Event Handler. If enabled, debug messages are written to the WebLogic Server log file during translation.

The XML Translator Plug-in provides additional display and edit capabilities over the standard WebLogic Process Integrator functionality. These capabilities are provided by the Hex Editor component of Format Tester for displaying and editing binary data.

Variable Types and the XML Translator Plug-In

The XML Translator Plug-In provides a `BinaryData` variable type, that you can use to edit and display binary data. The `BinaryData` variable acts as a container for a logical group of binary data with additional display capabilities. The `BinaryData`

variable is used by programs that call the actions provided by the XML Translator Plug-In to pass and receive binary data. It is also used by the Workflow instance monitor to display and edit the contents of a binary variable.

Custom Data Types and the XML Translator Plug-In

XML Translator includes a User Defined Type feature that allows you to create custom data types specific to your unique data type requirements. The User Defined Type feature allows these custom data types to be plugged in to the XML Translator runtime engine. Once a user defined data type is plugged-in, it is indistinguishable from a built-in data type in both features and function.

Configuring User Defined Data Types

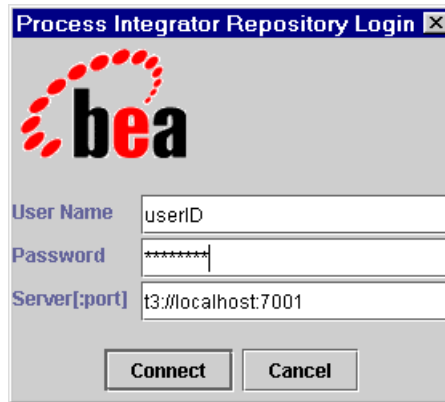
User Defined data types used by the XML Translator Plug-In are stored in the XML Translator repository as `CLASS` documents. At runtime, the XML Translator Plug-In loads user defined type classes from the repository as required. In addition, the XML Translator Plug-In will export the MFL and class files required to support the active template allowing a template to be imported on another Process Integrator instance intact. Class documents may be placed in the repository using one of the following methods:

- Using Format Builder
- Using the Repository Import Utility

Using Format Builder

Perform the following steps to publish a user defined type to the repository using Format Builder.

1. Start Format Builder by clicking Start→Programs→BEA WebLogic E-Business Platform→WebLogic Integration 2.0→xmltranslator→Format Builder. The Format Builder main window displays.
2. Choose Repository→Log In. The Process Integrator Repository Login window opens.

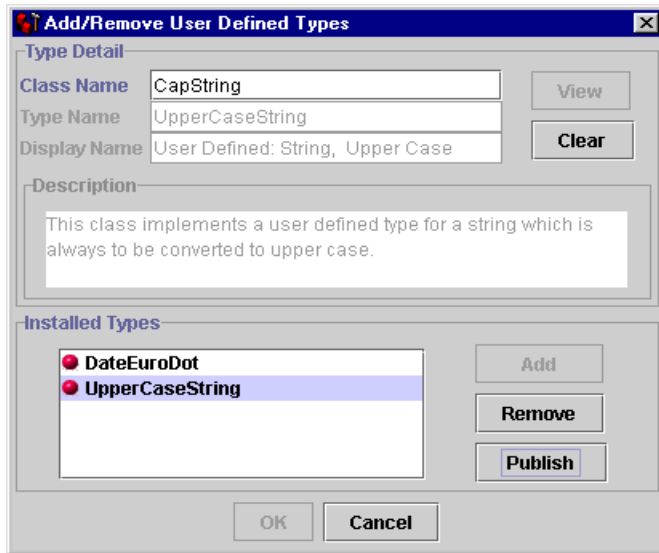


3. Enter the userid specified for the server in the User Name field.
4. Enter the password specified for the server in the Password field.
5. Enter the server name and Port number in the Server[:port] field.

Note: The Process Integrator Repository Login window allows up to three unsuccessful login attempts, after which, a login failure message is displayed. If you experience three login failures, choose Repository→Log In to repeat the login procedure.
6. Click Connect. If your login is successful, the Login window disappears and the Format Builder Title bar displays the server name and port number entered on the Process Integrator Repository Login window. You may now choose any of the active repository menu items to access.
7. Choose Tools→User Defined Types. The Add/Remove User Defined Types dialog box opens.

With a repository connection established, the Add/Remove User Defined Types dialog box displays the status of each registered user defined type and allows for its publication to the repository. The user defined type repository status is

reflected by an icon of a ball preceding the type name of each installed user defined type.



The color of the icon associated with each user defined type indicates its status:

- Green - The user defined type has been published to the repository.
 - Yellow - The user defined type has been published to the repository, however, the local version of the class differs from the repository version.
 - Red - The user defined type does not exist in the repository.
8. Select the class you want to publish from the list of Installed Types and click Publish. The icon for the selected entry should become green indicating the class was successfully placed in the repository.

Using the Repository Import Utility

Perform the following steps to use the repository import utility to import Java class files, including XML Translator user defined types.

1. Create a `wlxt-repository.properties` file in the CLASSPATH. The content of this file should be as follows:

```
wlxt.repository.url=<server url>
```

For example:

```
wlxt.repository.url=t3://localhost:7001
```

2. Type the following command to pass the class file name on the Import command line.

```
java com.bea.wlxt.repository.Import <file name>
```

For example, the following command imports all the class files in the current directory:

```
java com.bea.wlxt.repository.Import *.class
```

- Note:** Any Java class file may be imported to the repository using the Repository Import utility, not just user defined types. This is useful if a user defined type relies on additional class files that do not extend the `com.bea.wlxt.bintype.Bintype` class.

WebLogic Server Clustering Support

The XML Translator Plug-in can operate successfully in a WebLogic Server clustered environment. In a clustered environment, the plug-in administrator is connected to only one node of the cluster at any given time. Any commands issued by the administrator must be propagated to the other nodes in the cluster.

Communication among the various servers in a cluster is handled through the use of a JMS topic. The topic is used for communication between XML Translator components on different nodes in a cluster.

Configuring the XML Translator Plug-in for Clustering

If you want to take advantage of the clustering capability, you must configure the XML Translator Plug-In as follows:

1. Create a JMS topic on one of the servers within the cluster. The JNDI name of this topic must be as follows:

```
com.bea.wlxt.cluster.BroadcastTopic
```

Note: Refer to the WebLogic Server documentation for more information on creating JMS topics.

2. Open the `config.xml` file in a text editor. This file can be found in the `config` directory where you have WebLogic Process Integrator installed.

Note: The `config` directory contains separate subdirectories for each domain you have created. Each of these subdirectories contains its own `config.xml` file. Make sure you open the file under the correct domain.

3. Locate the `<Application>` section for WebLogic Process Integrator and add the following anywhere within this section:

```
<EJBComponent Name="wlxt-cluster"
  DeploymentOrder="99"
  Targets="[server_name]"
  URI="wlxtmb.jar"
/>
```

4. Save the `config.xml` file.

Note: You must restart the server in order for the change to the `config.xml` file to be recognized.

3 Running the WebLogic Process Integrator Sample Applications

The BEA WebLogic XML/Non-XML Translator software includes two sample applications designed to illustrate the integration of XML Translator with BEA WebLogic Process Integrator. This section describes these samples and gives you step-by-step instructions for running the samples. The following topics are discussed:

- Prerequisite Considerations
- Running the WebLogic Process Integrator Servlet Sample
- Running the WebLogic Process Integrator EJB Sample

Prerequisite Considerations

There are certain software applications that must be installed and tasks that must be performed prior to running these samples. Please refer to the *BEA WebLogic XML/Non-XML Translator Release Notes* for more information.

Note: The instructions presented in this section assume that you have a good working knowledge of BEA WebLogic Integration, specifically the WebLogic Process Integrator component, and BEA WebLogic Server. You should have

successfully installed the WebLogic Process Integrator component of WebLogic Integration and run a sample workflow prior to running the sample applications.

Running the WebLogic Process Integrator Servlet Sample

This sample application implements a Web Archive (`WLPI_sample.war`) that installs a servlet to accept requests for conversion of binary data to XML. The servlet is accessed via a browser and responds by displaying the generated XML data. In addition, the data is posted to the WebLogic Process Integrator event topic in either XML or binary format. The data may then be used to start a WebLogic Process Integrator workflow.

What is Included in the Servlet Sample

The following table provides a listing and description of the files included in the WebLogic Process Integrator Servlet sample application. This sample application can be found in the `samples\wpli\servlet` directory.

Table 3-1 List of Servlet Sample Application Files

Directory	File	Description
<code>\servlet\source</code>	<code>WLPI_sample.java</code>	The source code for the servlet used to present the HTML screen and process binary data to XML. This XML may be placed, optionally, onto the WebLogic Process Integrator JMS topic.
<code>\servlet</code>	<code>SampleData.mfl</code>	The Message Format Language description of the sample binary data file used to start the sample WebLogic Process Integrator workflow.
<code>\servlet</code>	<code>SampleData.data</code>	The sample data file used as input to start the sample WebLogic Process Integrator workflow.

Table 3-1 List of Servlet Sample Application Files

Directory	File	Description
\servlet	SampleWorkflow.xml	The exported WebLogic Process Integrator workflow used in the sample. This workflow should be imported via the WebLogic Process Integrator Studio GUI to setup the workflow tasks involved in the sample.
\servlet	Makefile	Make file for building the sample source to a .jar file.
\servlet	build.cmd	Builds the .jar file from source.
\servlet	WLPI_sample.war	A Web Archive file containing all executable sample code and configuration files.
\servlet\images	bealogo.jpg	The BEA logo image displayed on the HTML page rendered by the sample servlet.
\servlet\WEB-INF	hello.html	The HTML page used by the sample servlet to obtain input data from the user.
\servlet\WEB-INF	web.xml	The J2EE configuration file defining deployment information for the sample servlet.
\servlet\WEB-INF	weblogic.xml	The BEA configuration file defining WebLogic-specific information for the sample servlet.
\servlet\WEB-INF \lib	*.jar	Utility libraries, including XML Translator, that are used in the execution of the sample code.

How to Run the Servlet Sample

Follow the steps below to run the servlet sample. For instructions on the tasks specific to WebLogic Server and WebLogic Process Integrator, refer to the documentation that accompanies those applications.

Step 1. Configure and Run WebLogic Process Integrator

1. Start the WebLogic Process Integrator Server.
2. Copy the `WLPI_sample.war` file to the default web application directory. By default, these directories are:

3 Running the WebLogic Process Integrator Sample Applications

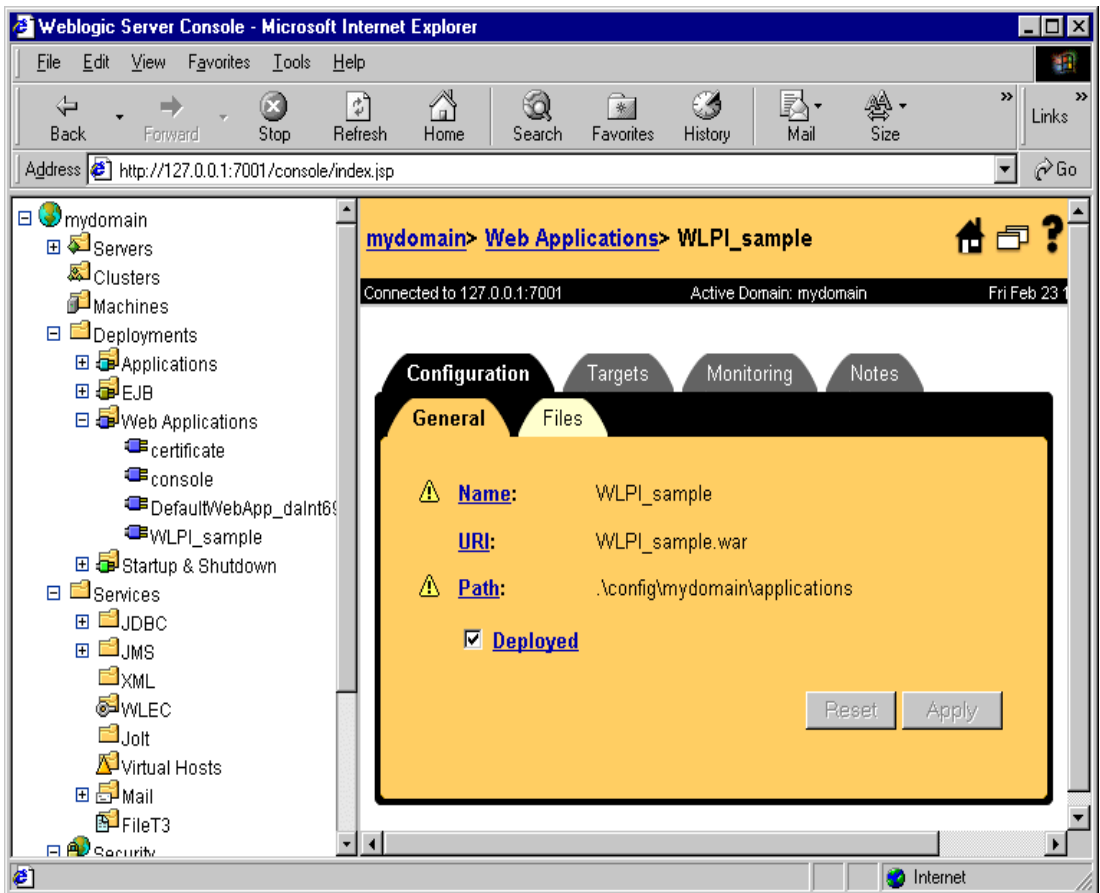
C:\bea\wlserver6.0\config\mydomain\applications (for Windows NT or Windows 2000)

\$BEA_HOME/wlserver6.0/config/mydomain/applications (for Unix)

Step 2. Deploy the Web Application

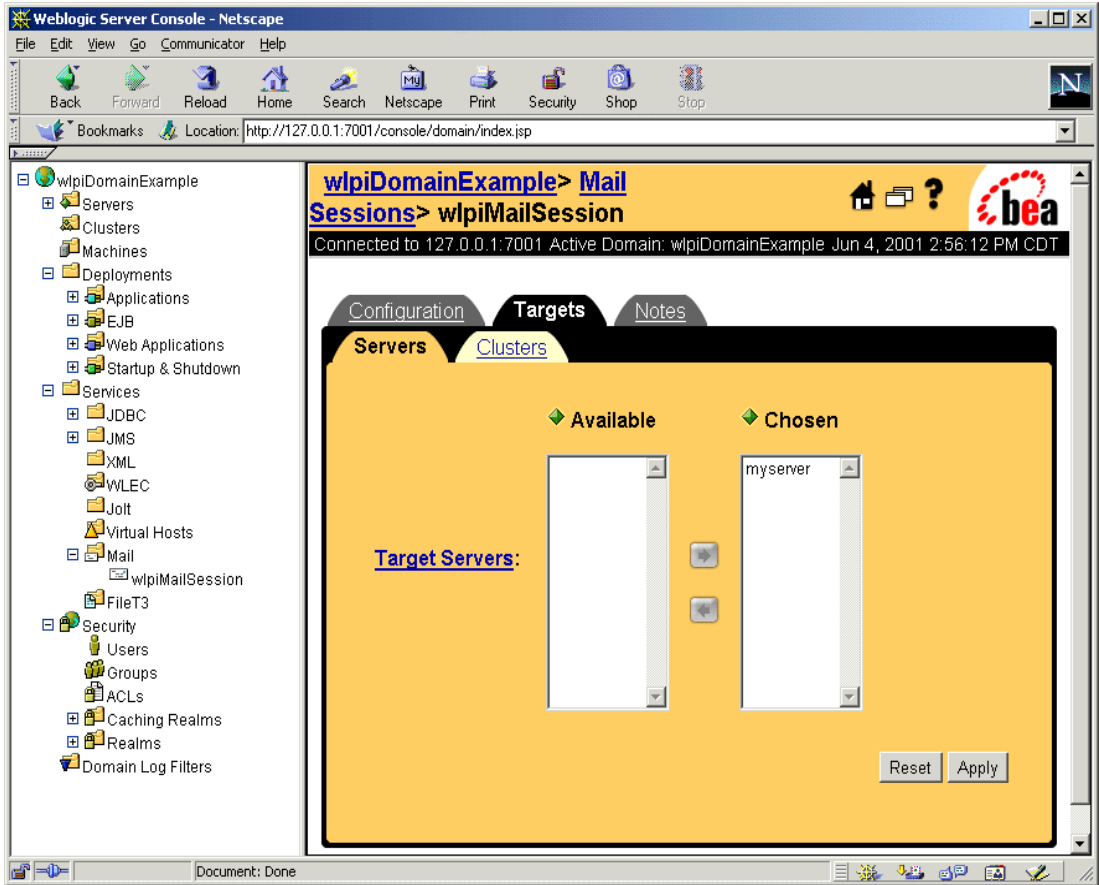
1. If your WebLogic Server has auto-deployment enabled, the sample web application is deployed within several seconds. Verify the deployment setting using the WebLogic Server Console, as shown in Figure 3-1.

Figure 3-1 WebLogic Server Console



2. If you have disabled auto-deployment on your WebLogic Server, you must statically deploy the web application using the WebLogic Server Console. To do this, follow the steps below:
 - a. From the tree in the left pane, choose Web Applications.
 - b. In the right pane, click Install a New Web Application.
 - c. Enter the path and file name of the `WLPI_sample.war` file and press Upload.
After the `WLPI_sample.war` file is successfully uploaded, `WLPI_sample` appears in the tree under Web Applications.
 - d. Select the new `WLPI_sample` node from the tree.
 - e. Click the Deployed checkbox in the right pane and click Apply.
 - f. Select the Targets tab.
 - g. Move your server name from the Available column to the Chosen column, as shown in Figure 3-2.

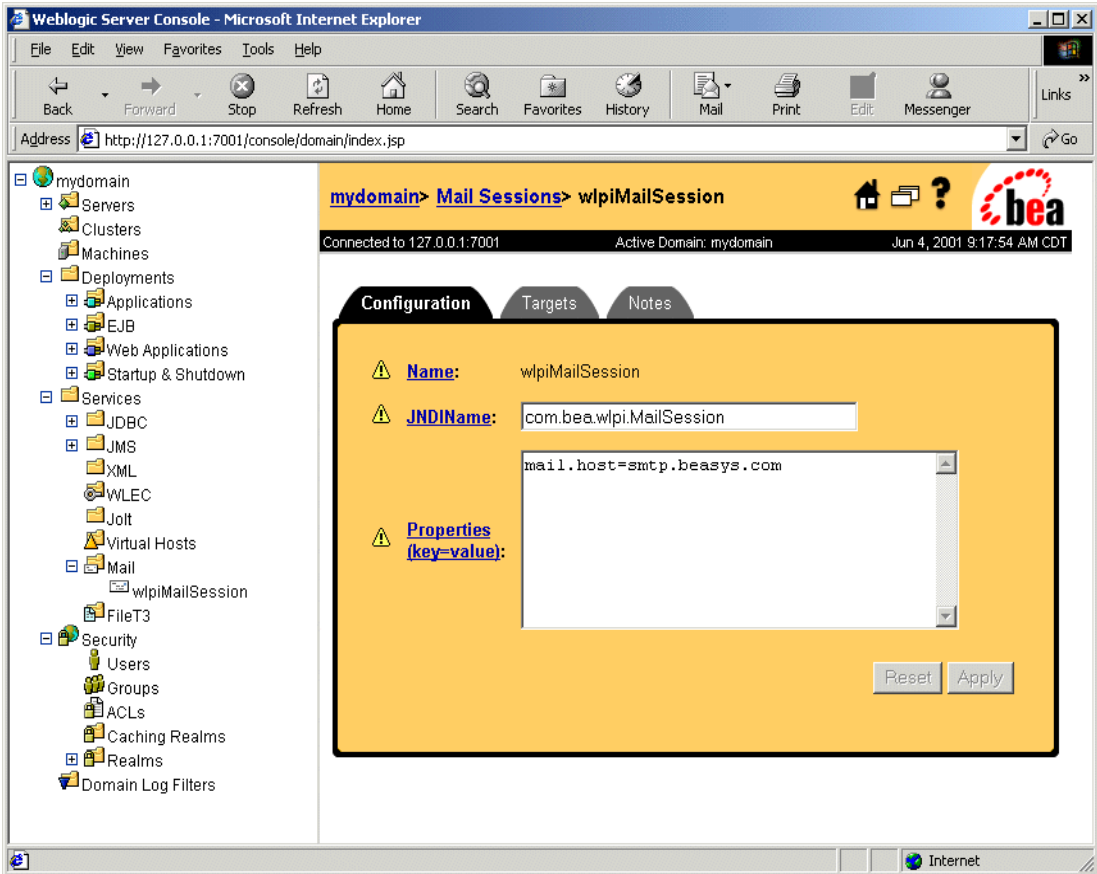
Figure 3-2 WebLogic Server Console Targets Tab



Step 3. Configure the Mail Session

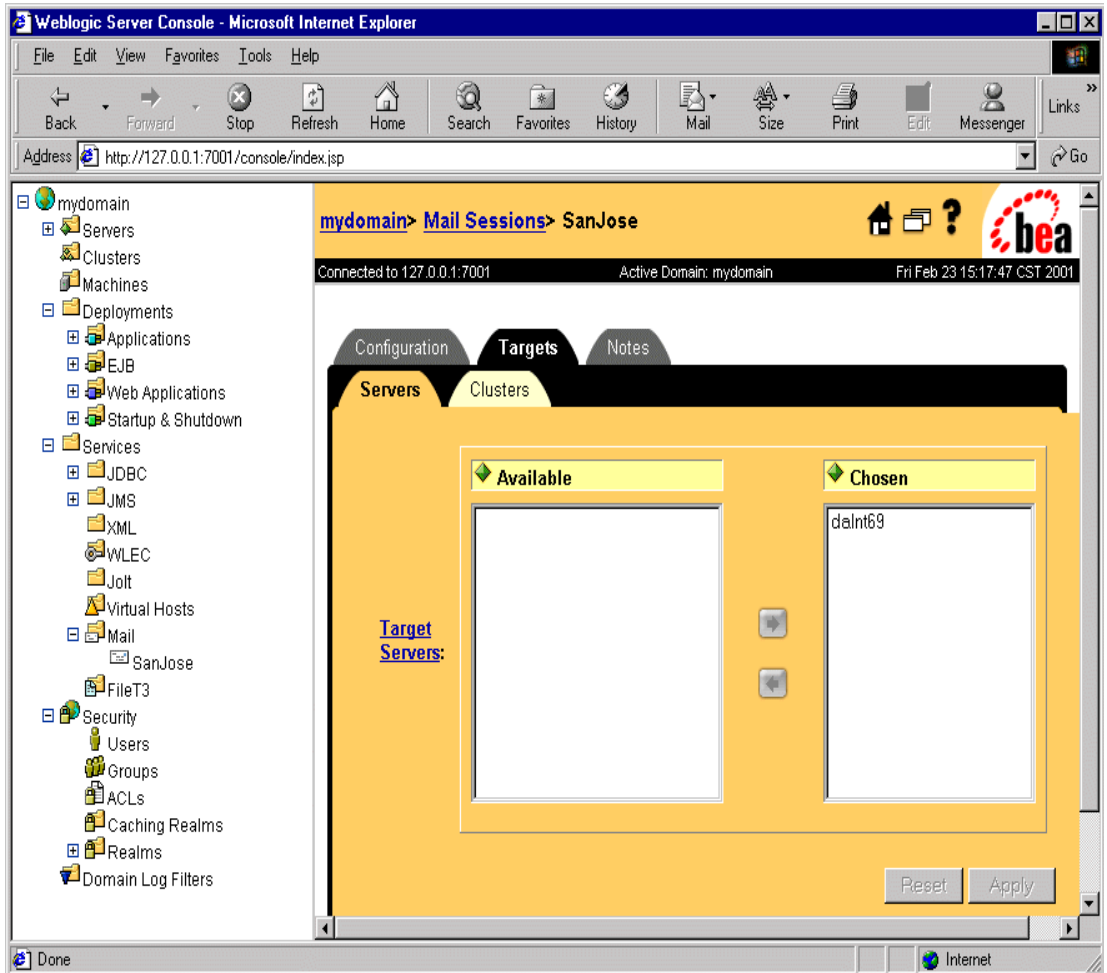
1. From the tree in the left pane, choose Mail→wlpMailSession.
2. Enter the appropriate information to configure your mail host. Figure 3-3 shows an example of the Mail Session Configuration screen.

Figure 3-3 WebLogic Server Console Mail Session Configuration Tab



3. Select the Targets tab.
4. Move your mail server name from the Available column to the Chosen column, as shown in Figure 3-4.

Figure 3-4 WebLogic Server Console Mail Session Targets Tab



Step 4. Create a New Template and Activate the Workflow

1. Start WebLogic Process Integrator Studio and log on.
2. Select Templates in the left pane, click the right mouse button and choose Create to create a new template.

3. Select the newly created template, click the right mouse button and choose Import Template Definition.
4. Select the file `\samples\wlpi\servlet\SampleWorkFlow.xml`.
5. Select the newly imported workflow, click the right mouse button and choose Open. The workflow opens.
6. Select the newly imported workflow, click the right mouse button and choose Properties. The Properties dialog displays.
7. Select Active and click OK.
8. Select the workflow, click the right mouse button and choose Save.

Step 5. Store the SampleData.mfl File in the Repository

There are two methods for storing files in the repository. Both methods are described below.

Using Format Builder

1. Start XML Translator Format Builder.
2. Open the `samples\wlpi\servlet\SampleData.mfl` file.
3. Log in to the repository.
4. Choose Repository→Store As to store the sample file in the repository.

Using the command line

At the console command prompt, invoke the Batch Import Utility using the following command.

```
java com.bea.wlxt.repository.Import [-v] [-n] [-t type] [-f folder] files...
```

The following information describes the commands and their options.

`-v`

specifies that verbose mode is on. This switch may appear anywhere within the command line and affects all operations that follow. Verbose mode is disabled by default.

- n specifies that verbose mode is off. This switch may appear anywhere within the command line and affects all operations that follow. Verbose mode is disabled by default.
- f Optional switch specifying the parent folder of all the following files. Multiple -f switches may be specified to change folders during an import execution. By default, documents are imported into the root folder of the repository. A special -f switch argument of @ may be used to specify the root folder.

Folder names specified in the -f switch are always absolute pathnames from the repository root folder. Folder names within a path should be separated by a forward slash.
- t Optional switch specifying the default type of all the following files. The default type is assigned to documents when the document type cannot be determined by the file extension. Valid values are .mfl, .dtd, .class, .xsl, and .xsd.
- files specifies one or more filenames to be imported. Wildcards may be used based on the current command line shell.

Step 6. Generate the XML Data and Send the Message

1. Using a text editor, open the file `\samples\wlpi\servlet\SampleData.data`. Replace the text `user@bea.com` with a valid email address. This is the address the workflow uses to deliver the email message.
2. Open a browser and go to the following URL:
`http://<weblogic server/port>/WLPI_sample/WLXTTest`
3. Enter `SampleData` into the MFL text field.
4. Browse to the following data file:
`samples\wlpi\servlet\SampleData.data`
5. Select the option to invoke WebLogic Process Integrator and click Submit. A short email message is sent to the address you supplied in the data file.

Running the WebLogic Process Integrator EJB Sample

This sample simulates a dataflow from an HR system to a payroll system, initiated by the entry of payroll data. The employee data is obtained from a legacy payroll system that uses binary data. The data is translated to XML in order to perform a calculation to determine the employee's pay information. The result of the calculation is translated back to binary and sent on to the payroll system.

What is Included in the EJB Sample

The following table provides a listing and description of the files included in the WebLogic Process Integrator EJB sample application. This sample application can be found in the `samples\wpli\ejb` directory.

Table 3-2 List of EJB Sample Application Files

Directory	File	Description
\ejb	Makefile	Make file for building the sample source to a .jar file.
\ejb	WLXTEexample.jar	Exported sample workflow from WebLogic Process Integrator.
\ejb	HR.mfl	MFL file for binary data returned from the Sample HR Bean.
\ejb	Payroll.mfl	MFL file for binary data passed to the Sample Payroll Bean.
\ejb	Autopay.cmd	Windows NT command script to initiate the workflow from the command line.
\ejb	Autopay.sh	Unix shell script to initiate the workflow from a command prompt.
\ejb	build.cmd	Builds wlxtejb.jar from source.

3 *Running the WebLogic Process Integrator Sample Applications*

Table 3-2 List of EJB Sample Application Files

Directory	File	Description
\ejb\lib	WLXTEJB.jar	Executables for the sample application.
\ejb\source	Payroll.java	Sample EJB to represent legacy payroll system.
\ejb\source	PayrollHome.java	Sample EJB to represent legacy payroll system.
\ejb\source	PayrollBean.java	Sample EJB to represent legacy payroll system.
\ejb\source	HR.java	Sample EJB to represent legacy HR system.
\ejb\source	HRHome.java	Sample EJB to represent legacy HR system.
\ejb\source	HRBean.java	Sample EJB to represent legacy HR system.
\ejb\source	AutoPay.java	Program to place a pre-formatted message on the WLPI Event Topic to start the sample workflow.
\ejb\source	HexDump.java	Utility class used by the sample EJBs.
\ejb\source	EmployeeRecord.java	Employee data class used by the sample HR EJB.

How to Run the EJB Sample

Follow the steps below to run the EJB sample. For specific instructions on performing the tasks in WebLogic Process Integrator and WebLogic Server, please refer to the documentation that accompanies those applications.

Step 1. Configure and Run WebLogic Process Integrator

1. Copy the file `WLXTEJB.jar` from the `\samples\wlpiejb\lib` directory where you have XML Translator installed to the `\lib` directory where you have WebLogic Process Integrator installed.
2. Open the file `config.xml` in a text editor. This file can be found in the `config` directory where you have BEA WebLogic Process Integrator installed.

Note: The `config` directory contains separate subdirectories for each domain you have created. Each of these subdirectories contains its own `config.xml` file. Make sure you open the file under the correct domain.

3. Add the following lines to the end of the WebLogic Process Integrator Application section of the file:

```
<EJBComponent
  Name="wlxt-sample"
  Targets="<your_machine_name>"
  URI="WLXTEJB.jar"
/>
```

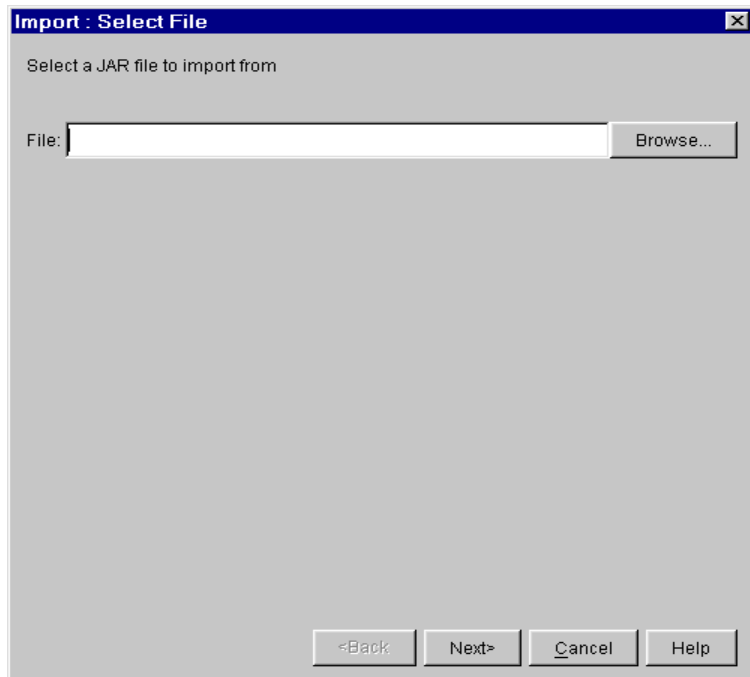
4. Start the WebLogic Process Integrator Server.

Step 2. Import the Workflow Definition

To import the workflow definition:

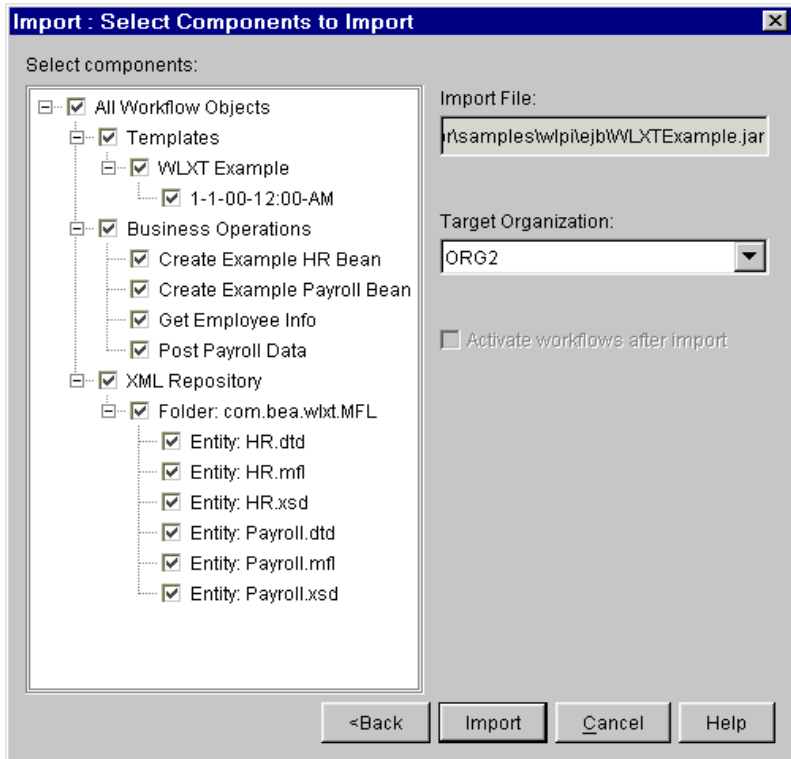
1. Run WebLogic Process Integrator Studio.
2. Choose Tools→Import Package. The Import: Select File dialog opens (Figure 3-5).

Figure 3-5 Import: Select File Dialog



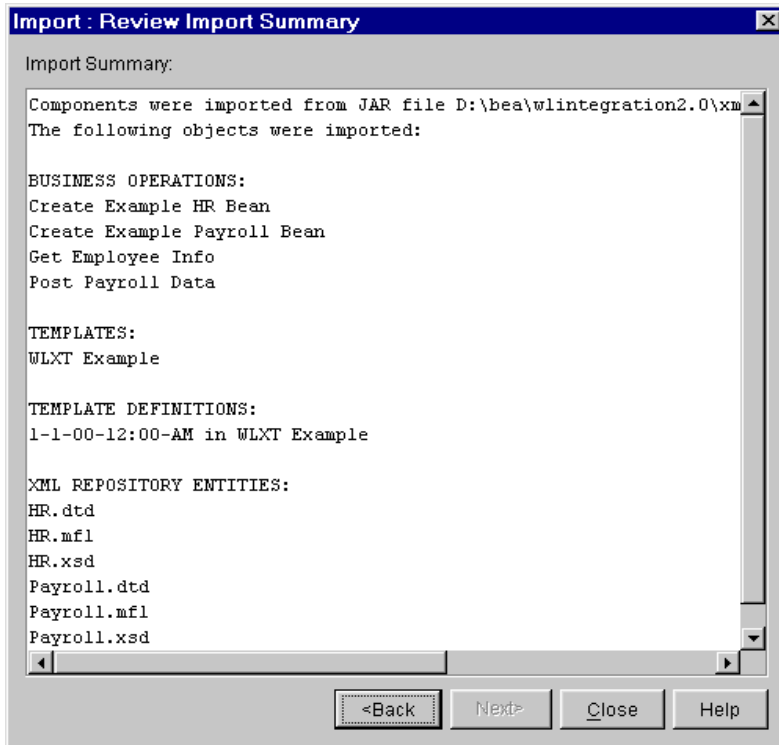
3. Click Browse, select the definition file `WLXTExample.jar`, and click Open. Click Next, the Import: Select Components to Import dialog opens (Figure 3-6).

Figure 3-6 Import: Select Components to Import



4. Make sure all components are selected and click Import. The Import: Review Import Summary dialog opens (Figure 3-7).

Figure 3-7 Import: Review Import Summary



5. Confirm that the correct components are listed. If not, click Back and select the components again. If so, click Close. You are now ready to open the template.

Step 3. Open the Template

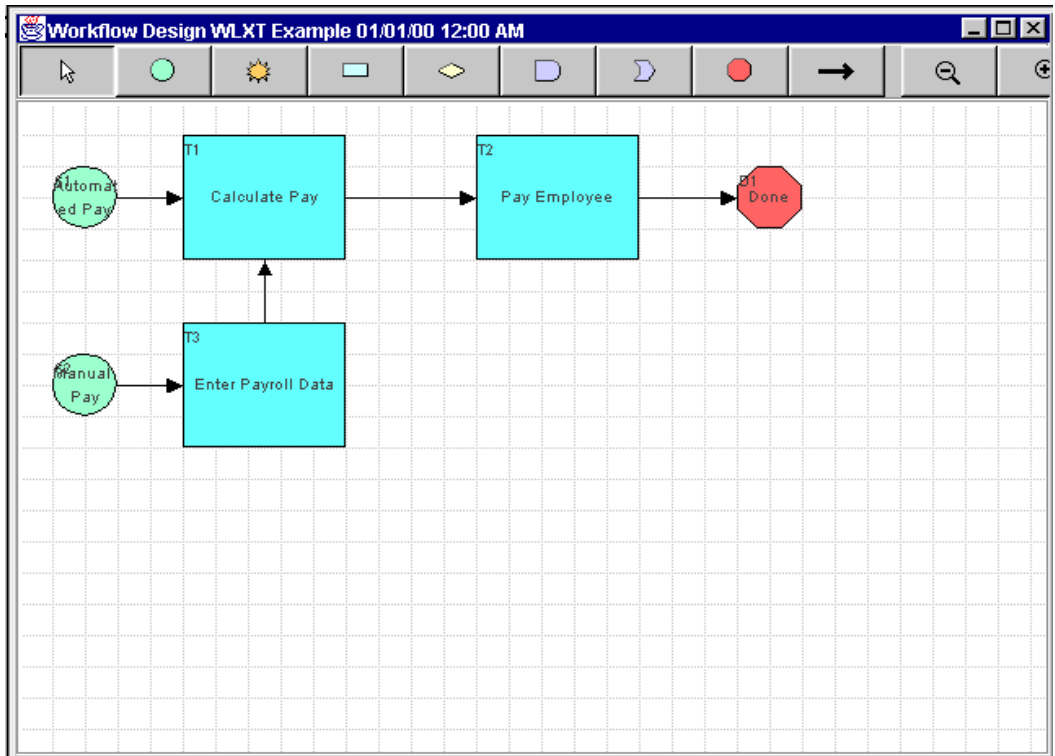
To open the template:

1. Expand the WLXT Example template imported in the previous step in the tree view. Select the template definition 1-1-00-12:00-AM and click the right mouse button.

3 *Running the WebLogic Process Integrator Sample Applications*

2. Choose Open. The workflow created for this sample application displays.

Figure 3-8 Workflow for XML Translator Example



3. Select the XML Translator Example template definition again from the tree view and click the right mouse button.
4. Choose Properties. The Template Definition properties dialog displays.

Figure 3-9 Template Definition

The screenshot shows a dialog box titled "Template Definition WLXT Example". It has two tabs: "General" (selected) and "Exception Handlers". In the "General" tab, there is an "Id" text field with an "A+BQ" button to its right. Below it is a checked checkbox for "Active". There are two date pickers: "Effective" set to "Jan 1, 2000" and "Expiry" set to "Dec 31, 2000". Below these is an unchecked checkbox for "Enable auditing". A "Notes" text area is empty. At the bottom of the main content area, there are two fields: "Last changed on" with the value "Dec 12, 2000 9:47:00 AM" and "Last changed by" with the value "joe". At the very bottom of the dialog are "OK" and "Cancel" buttons.

5. Click Active to confirm that the template is active and click OK.
6. Select the XML Translator Example template definition a third time from the tree view and click the right mouse button again.
7. Choose Save to save the template definition with the changes you made.

Step 4. Start the Workflow

There are two ways to start the workflow created in the sample:

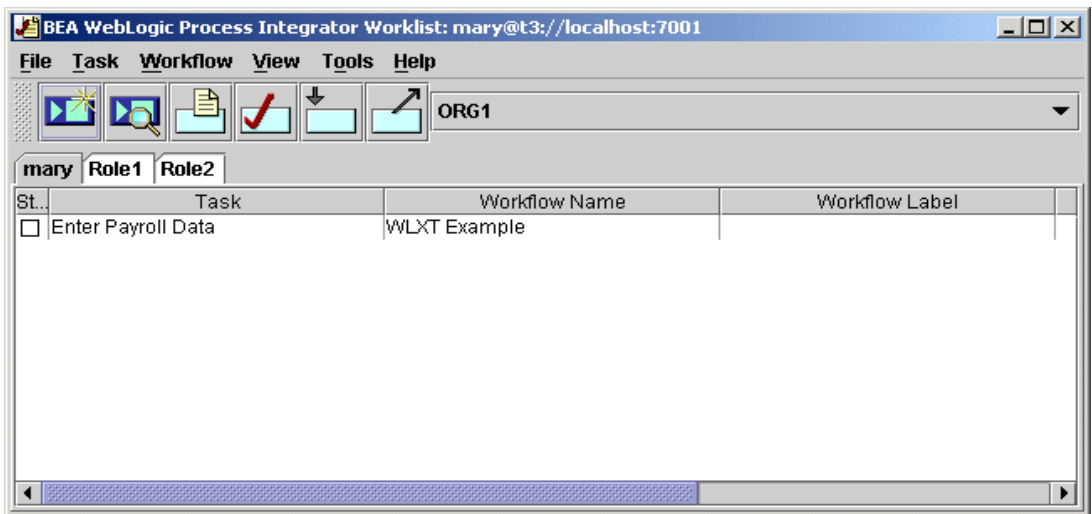
- From the WebLogic Process Integrator Worklist
- From the Command Line

From the WebLogic Process Integrator Worklist

To start the sample workflow from the Weblogic Process Integrator Worklist:

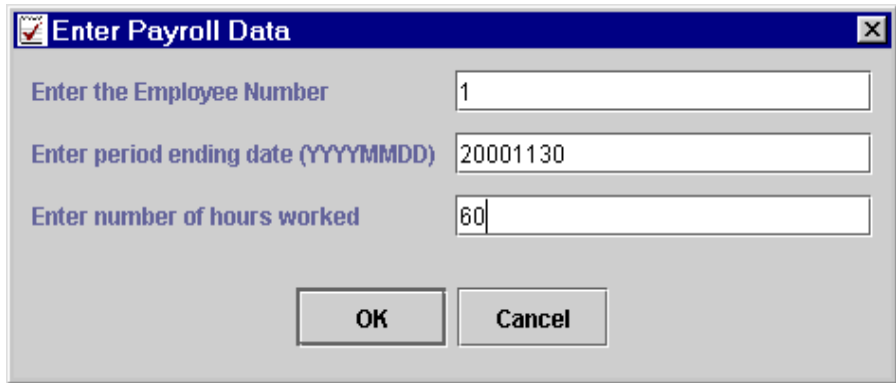
1. Start WebLogic Process Integrator Worklist and choose Workflow→Start a Workflow.
2. Select WLXT Example. Click OK.

Figure 3-10 XML Translator Example Worklist



3. Select the Enter Payroll Data task and click the right mouse button.
4. Choose Execute. The Enter Payroll Data dialog displays.

Figure 3-11 Enter Payroll Data



The screenshot shows a standard Windows-style dialog box titled "Enter Payroll Data". It features three text input fields. The first field is labeled "Enter the Employee Number" and contains the number "1". The second field is labeled "Enter period ending date (YYYYMMDD)" and contains the date "20001130". The third field is labeled "Enter number of hours worked" and contains the number "60". Below the input fields are two buttons: "OK" and "Cancel".

5. Enter the payroll data and click OK. The task is started and the workflow runs.

Note: For this example, the employee numbers 1 through 4 are valid. You can enter any period ending date and any number of hours worked.

From the Command Line

To start the sample workflow from a command line prompt:

1. Open the script file (`Autopay.cmd` on Windows NT systems; `Autopay.sh` on Unix systems) in a text editor and check the location of the WebLogic Process Integrator Server. By default, the location is `localhost` and `port:7001`.
2. Change the location information to match the host and port for your system.
3. Set the environment variable `WL_HOME` to the home directory for WebLogic Server on your system. For example:

```
set WL_HOME=c:\bea\wlserver6.0
```

4. Run the command scripts for your system (Windows NT or Unix), passing the same parameters shown in Figure 3-11. For example:

```
Autopay 1 2000-11-30 60
```

Figure 3-12 shows the WebLogic Process Integrator output from executing the workflow.

Figure 3-12 WebLogic Process Integrator Server Console

```

Command Prompt - server
-----
HR getEmployeeInfo called with EmpNum 1
HR returning:
00000001456477617264732020202020      ...Edwards
2020202020202020205065746520202020      Pete
20202020202020202020202020202036313220
4d61696e2053742e202020202020202020      Main St.
202020202020202020202020202020202020      612
202020202020202020202020202020202020      Purchase
2020202020202020205075726368617365
202020202020202020202020202020202020      NY..?.Ah..
202020202020202020202020202020202020
4e5900003fcd4168cccd

XMLnonXMLTranslator: binary2XML called with:
binaryData:
00000001456477617264732020202020      ...Edwards
202020202020202020202020202020202020      Pete
20202020202020202020202020202036313220
4d61696e2053742e202020202020202020      Main St.
202020202020202020202020202020202020      612
202020202020202020202020202020202020      Purchase
2020202020202020205075726368617365
202020202020202020202020202020202020      NY..?.Ah..
202020202020202020202020202020202020
4e5900003fcd4168cccd

MFL File:
HR.mfl

XMLnonXMLTranslator: Returning data from binary2XML:
<?xml version="1.0"?><EmployeeRecord><EmpNum>1</EmpNum><LastName>Edwards</LastNa
me><FirstName>Pete</FirstName><Address>612 Main St.</Address><City>Purchase</Cit
y><State>NY</State><Zip>16333</Zip><HourlyWage>14.55</HourlyWage></EmployeeRecor
d>

XMLnonXMLTranslator: xml2Binary called with:
xmlData:
<?xml version="1.0" encoding="UTF-8"?>
<PayData>      <EmpNum>1</EmpNum>      <PeriodEnding>20001130</PeriodEnding>      <
HoursWorked>60</HoursWorked>      <GrossEarnings>873.0</GrossEarnings>      </PayDat
a>

MFL File: Payroll.mfl

XMLnonXMLTranslator: Returning data from xml2Binary:
0000000132303030313133300000003c      ...20001130...<
445a4000      DZE.

Payroll postPayrollData called:
EmpNum: 1
PeriodEnding: 20001130
HoursWorked: 60
GrossEarnings: 873.0
    
```

Index

B

BinaryData variable 2-12

C

cache hits 2-12

clustering

- configuring XML translator plug-in 2-16
- WebLogic Server 2-16

com.bea.wlxt.cluster.BroadcastTopic 2-17

config.xml file 2-17

current size 2-11

D

data translation 2-2

debug messaging 2-12

design-time component 1-4

E

EJB sample

- files 3-11

event data

- processing 2-8

H

high water mark 2-12

hit ratio 2-12

I

import

- repository 2-15

M

mail session

- configuring 3-6

message format language (MFL) 1-4

mfl requests 2-12

P

performance 2-9

plug-in prerequisites 1-7

preferred pool size 2-11

prerequisites

- plug-in 1-7

processing event data 2-8

R

refresh 2-12

repository

- using 1-6

repository input utility 2-15

run-time component 1-5

run-time plug-in to WebLogic Process
integrator 1-5

S

servlet sample

- included files 3-2

- running 3-3

U

user defined data types

- configuring 2-13

W

WebLogic Server clustering 2-16

WLPI_sample.war file 3-5

WLXTEExample.jar file 3-14

wlxt-repository.properties file 2-15

workflow

- starting 3-18

workflow definition

- importing 3-13