



BEA WebLogic Integration™

HTTP Plug-in User Guide

Release 7.1
Release Date: June 2003
Revision Date: April 2004

Copyright

Copyright © 2004 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, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop, and How Business Becomes E-Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

BEA WebLogic Integration HTTP Plug-in User Guide

Part Number	Date	Software Version
N/A	Released: June 2003 Revised: April 2004	7.1

Contents

About This Document

What You Need to Know	v
e-docs Web Site	vi
How to Print the Document	vi
Related Information	vii
Contact Us!	vii
Documentation Conventions	viii

1. Introducing the BEA WebLogic Integration HTTP Plug-in

About the HTTP Plug-in	1-1
What the HTTP Plug-in Does	1-2
What You Need to Know	1-2
HTTP Plug-in and the Plug-in Framework	1-3

2. Deploying the HTTP Plug-in

Understanding the Representation of Paths	2-2
Deploying on WebLogic Integration 7.0 SP2	2-2
Deploying on WebLogic Integration 2.1 SP2	2-6
Precautionary Steps to Avoid Errors During Use	2-9
Updating the BEA License	2-11
Verifying Deployment	2-12

3. Using the HTTP Plug-in

Overview	3-1
Defining Workflow Variables	3-2
Setting Task Properties	3-3
Workflow Expressions	3-5

Sending an HTTP Request to a URL.....	3-6
Communicating Via a Secure HTTP (HTTPS) Connection.....	3-11
Capturing HTTP Response Data	3-14
Sending Business Data as an HTTP GET	3-15
Sending a Binary/XML Document as an HTTP POST.....	3-17
Starting a Workflow When an HTTP Request Arrives	3-19
 4. Configuring the HTTP Plug-in for a Migrated Domain	
Updating the BPM Database Table	4-1
Migrating to a Single Server Domain.....	4-2
Migrating to a Clustered Domain	4-2
 5. HTTP Plug-in Example	
Setting Up the Workflow.....	5-1
Executing the Workflow.....	5-9
Executing on WebLogic Integration 7.0	5-9
Executing on WebLogic Integration 2.1	5-11

Index

About This Document

This document explains how to deploy and use the BEA WebLogic Integration HTTP Plug-in. It is organized as follows:

- [Chapter 1, “Introducing the BEA WebLogic Integration HTTP Plug-in,”](#) provides a brief description of the HTTP Plug-in.
- [Chapter 2, “Deploying the HTTP Plug-in,”](#) contains instructions for deploying the plug-in with WebLogic Integration 2.1 Service Pack 2 (SP2) and WebLogic Integration 7.0 SP2.

Note: The plug-in is included with WebLogic Integration 7.0 SP5. If you have installed that release, the plug-in is automatically deployed on server startup.
- [Chapter 3, “Using the HTTP Plug-in,”](#) describes how to use the HTTP Plug-in features.
- [Chapter 4, “Configuring the HTTP Plug-in for a Migrated Domain,”](#) describes how to update your database schema and configure the HTTP Plug-in for a migrated domain.
- [Chapter 5, “HTTP Plug-in Example,”](#) provides a step-by-step example of using the HTTP Plug-in to send an HTTP request to a Web server.

What You Need to Know

This document is intended for Workflow Designers and System Integrators who develop applications that interact with HTTP servers.

The information provided in this document requires you to have in-depth knowledge of Workflow Design and Workflow Templates, and WebLogic Integration (WLI) Studio. Additionally, it is assumed that you know Web technologies and have a general understanding of Microsoft Windows and UNIX systems.

If you do not have the required knowledge of workflows or the WebLogic Integration Studio, see the following documents:

- *Using the WebLogic Integration Studio* at <http://edocs.bea.com/wli/docs70/studio/index.htm>
- *Learning to Use BPM with WebLogic Integration* at <http://edocs.bea.com/wli/docs70/bpmtutor/index.htm>

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://edocs.bea.com>.

How to Print the Document

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

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

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

Related Information

The following WebLogic Integration documents contain information that is relevant to using this product:

- *Using the WebLogic Integration Studio* at <http://edocs.bea.com/wli/docs70/studio/index.htm>
- *Learning to Use BPM with WebLogic Integration* at <http://edocs.bea.com/wli/docs70/bpmtutor/index.htm>

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 that you are using the documentation for the BEA WebLogic HTTP Plug-in 7.1 release.

If you have any questions about this version of HTTP Plug-in, 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
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
<i>italics</i>	Indicates emphasis or book titles.
monospace text	Indicates code samples, commands and their options, data structures and their members, data types, directories, and filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard. <i>Examples:</i> <pre>#include <iostream.h> void main () the pointer psz chmod u+w * \tux\data\ap .doc tux.doc BITMAP float</pre>
monospace boldface text	Identifies significant words in code. <i>Example:</i> <pre>void commit ()</pre>
<i>monospace italic text</i>	Identifies variables in code. <i>Example:</i> <pre>String <i>expr</i></pre>
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. <i>Examples:</i> <pre>LPT1 SIGNON OR</pre>

Convention	Item
{ }	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
[]	Indicates optional items in a syntax line. The brackets themselves should never be typed. <i>Example:</i> buildobjclient [-v] [-o name] [-f file-list]... [-l file-list]...
	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
...	Indicates one of the following in a command line: <ul style="list-style-type: none">■ 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. <i>Example:</i> 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 Introducing the BEA WebLogic Integration HTTP Plug-in

This section introduces the BEA WebLogic Integration HTTP Plug-in, and describes how it functions in a Business Process Management (BPM) workflow. It includes the following topics:

- [About the HTTP Plug-in](#)
- [What the HTTP Plug-in Does](#)
- [What You Need to Know](#)
- [HTTP Plug-in and the Plug-in Framework](#)

About the HTTP Plug-in

The HTTP Plug-in extends the functionality of the BEA WebLogic Integration BPM Studio. It enables you to access a Web application from the WebLogic Integration Studio, and the WebLogic Integration Studio from a Web application. Specifically, the HTTP Plug-in provides the ability to send HTTP requests and invoke workflow instances when HTTP requests are received.

For more information, see *Programming BPM Plug-ins for WebLogic Integration* at <http://edocs.bea.com/wli/docs70/devplug/index.htm>.

What the HTTP Plug-in Does

The HTTP Plug-in has the following functionality:

- Send an HTTP/HTTPS Request to a URL, specifically:
 - Send business data as an HTTP GET
 - Send Binary/XML document as an HTTP POST
 - Communicate via a secure HTTP (HTTPS) connection with both client-side and server-side authentication
 - Capture HTTP Header values for either a GET or POST in a variable
- Start a workflow when an HTTP Request is sent to a specified URL

What You Need to Know

This document is written for workflow designers and system integrators who develop client interfaces between file systems and other applications. The information provided in this document requires that you have in-depth knowledge of workflow design and workflow templates, Hypertext Transfer Protocol (HTTP), and WebLogic Integration Studio. Additionally, it is assumed that you know Web technologies and have a general understanding of Microsoft Windows and UNIX systems.

If you do not have the required knowledge of workflows or the WebLogic Integration Studio, see the following documents:

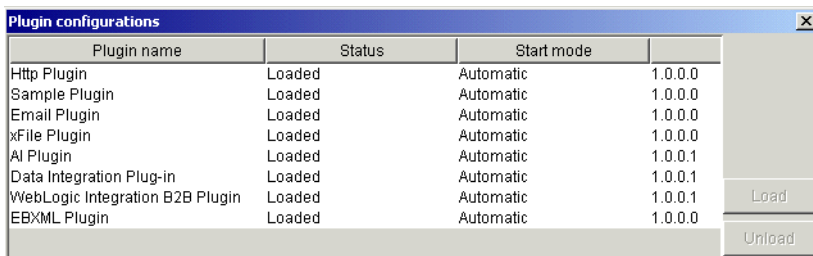
- *Using the WebLogic Integration Studio* at <http://edocs.bea.com/wli/docs70/studio/index.htm>.
- *Learning to Use BPM with WebLogic Integration* at <http://edocs.bea.com/wli/docs70/bpmtutor/index.htm>.

HTTP Plug-in and the Plug-in Framework

Like other plug-ins that extend BPM functionality, the HTTP Plug-in adheres to the BPM Plug-in specification. For more information, see *Programming BPM Plug-ins for WebLogic Integration* at <http://edocs.bea.com/wli/docs70/devplug/index.htm>. The Plug-in Configurations window displays the plug-ins and their status, as shown in the following figure.

For details on how to see the Plug-in Configurations window, see the section “Verifying Deployment,” in Chapter 2, “Deploying the HTTP Plug-in.”

Figure 1-1 Plug-in Configurations Window



The screenshot shows a window titled "Plugin configurations" with a close button in the top right corner. Inside the window is a table with four columns: "Plugin name", "Status", "Start mode", and an empty column. The table lists eight plug-ins, all with a status of "Loaded" and a start mode of "Automatic". The version numbers are 1.0.0.0 for the first four, 1.0.0.1 for the next three, and 1.0.0.0 for the last one. To the right of the table are two buttons: "Load" and "Unload".

Plugin name	Status	Start mode	
Http Plugin	Loaded	Automatic	1.0.0.0
Sample Plugin	Loaded	Automatic	1.0.0.0
Email Plugin	Loaded	Automatic	1.0.0.0
xFile Plugin	Loaded	Automatic	1.0.0.0
AI Plugin	Loaded	Automatic	1.0.0.1
Data Integration Plug-in	Loaded	Automatic	1.0.0.1
WebLogic Integration B2B Plugin	Loaded	Automatic	1.0.0.1
EBXML Plugin	Loaded	Automatic	1.0.0.0

1 *Introducing the BEA WebLogic Integration HTTP Plug-in*

2 Deploying the HTTP Plug-in

If you have installed WebLogic Integration 7.0 Service Pack 5, the plug-in is included in your installation and is deployed on server start up. See [“Verifying Deployment” on page 2-12](#) to verify deployment.

If you have an earlier release of WebLogic Integration installed and do not wish to upgrade the latest WebLogic Integration 7.0 Service Pack, you can install the plug-in as described in the following sections:

- [Understanding the Representation of Paths](#)
- [Deploying on WebLogic Integration 7.0 SP2](#)
- [Deploying on WebLogic Integration 2.1 SP2](#)
- [Updating the BEA License](#)
- [Verifying Deployment](#)

Understanding the Representation of Paths

Because the location of files in the WebLogic Integration environment depends on options selected during installation and configuration, the conventions that follow are used throughout to represent paths.

- *BEA_HOME* represents the BEA Home directory specified for your WebLogic installation.

If you install the product in the default location on a Windows system, *BEA_HOME* represents `C:\bea`.

- *WLI_HOME* represents the root of your WebLogic Integration installation.

For example:

- If you install WebLogic Integration 7.0 in the default location on a Windows system, *WLI_HOME* represents `C:\bea\weblogic700\integration`
- If you install WebLogic Integration 2.1 in the default location on a Windows system, *WLI_HOME* represents `C:\bea\wlintegration2.1`

Note: *WLI_HOME* and *BEA_HOME* also represent the corresponding Windows and UNIX environment variables. For example, the literal interpretation of *WLI_HOME* is `%WLI_HOME%` for Windows and `$WLI_HOME` for UNIX.

- *localhost* represents the IP address of the machine running the WebLogic Server.

Deploying on WebLogic Integration 7.0 SP2

The HTTP Plug-in is included with WebLogic Integration 7.0 Service Pack 5. If you have WebLogic Integration 7.0 Service Pack 5 installed, skip to [“Verifying Deployment” on page 2-12](#) to verify deployment.

If you do not wish to upgrade to the latest WebLogic Integration 7.0 Service Pack for some reason, you can deploy the HTTP Plug-in on WebLogic Integration 7.0 Service Pack 2 (SP2) by copying `.jar` files to your WebLogic installation directory, editing

the `application.xml` and `config.xml` files, copying a `.sql` script file to the appropriate location, and running a command to create the `HTTPPOLL` table in your database.

If you downloaded the HTTP Plug-in, the following files are all located in the `.zip` file you downloaded. If you received the HTTP Plug-in on a CD, they are located on the CD.

- `httpplugin-ejb.jar`
- `httpplugin.war`
- `plugin-shared.jar`
- `db2\hp_schema.sql`
- `mssql\hp_schema.sql`
- `pointbase\hp_schema.sql`
- `oracle\hp_schema.sql`
- `sybase\hp_schema.sql`
- `cloudscape\hp_schema.sql`
- `httpsetupdb.cmd`
- `httpsetupdb.sh`

To deploy the HTTP Plug-in on WebLogic Integration 7.0 SP2, do the following:

1. From the source location (the directory where you unzipped the `.zip` file or the product CD), copy the following files into the `BEA_HOME\weblogic700\integration\lib` directory:
 - `httpplugin-ejb.jar`
 - `httpplugin.war`
 - `plugin-shared.jar`
2. From the `BEA_HOME\weblogic700\integration\lib\META-INF` directory, open the `application.xml` file and add the lines in bold to the existing configuration, at the location shown here:

```
<application>
.
.
.
  <!--HTTP Plugin-->
  <module>
```

```
        <ejb>httpplugin-ejb.jar</ejb>
    </module>
    <module>
        <web>
            <web-uri>httpplugin.war</web-uri>
            <context-root>com.bea.wlpi.HttpPlugin</context-root>
        </web>
    </module>
    .
    .
    .
    <!--BPM Initialization Bean must be deployed
    after BPM plug-ins-->
    <module>
        <ejb>bpm-init-ejb.jar</ejb>
    </module>
</application>
```

3. Save the file and close it.

4. From the

BEA_HOME\weblogic700\samples\integration\config\samples directory, open the *config.xml* file and add the lines in bold to the existing configuration, at the location shown here.

Note: If you have already created a specific domain, open the *config.xml* file from that domain. The scripts given here assume that the default domain is *samples*.

```
<Application
    Deployed="true"
    Name="WLI"
    Path="D:\bea\weblogic700\integration\lib"
    TwoPhase="true">
    <EJBComponent
        Name="WLI-BMP HTTP Plug-in"
        Targets="myserver"
        URI="httpplugin-ejb.jar"/>
    <EJBComponent
        Name="Sample BPM Plug-in"
        Targets="myserver"
        URI="sampleplugin-ejb.jar"/>
    .
    .
    .
    <EJBComponent
        Name="WLI-DI Server"
        Targets="myserver"
```

```

        URI="WLXTEJB.jar" />
    .
    .
    .
    <WebAppComponent
        Name="HTTP Plug-in
        EventListener"
        Targets="myserver"
        URI="httpplugin.war" />
    .
    .
    .
    <WebAppComponent
        Name="wlai"
        Targets="myserver"
        URI="wlai.war" />
    <WebAppComponent
        Name="wlisWebApp"
        Targets="myserver"
        URI="wlisWebApp.war" />
</Application>

```

5. Save the file and close it.
6. Copy the `hp_schema.sql` file for your database from the source location (the directory where you unzipped the .zip file or product CD) to the location listed in the table that follows:

Table 2-1 Location for `hp_schema.sql` Files for Various Databases

For This Database	Copy the <code>hp_schema.sql</code> File to This Location
DB2	<code>WLI_HOME\dbscripts\db2\</code>
MS SQL	<code>WLI_HOME\dbscripts\mssql\</code>
Oracle	<code>WLI_HOME\dbscripts\oracle\</code>
Pointbase	<code>WLI_HOME\dbscripts\pointbase\</code>
Sybase	<code>WLI_HOME\dbscripts\sybase\</code>

7. Go the location where you unzipped the HTTP file and open the file that creates the HTTPPoll database required for the HTTP Plug-in.
 - If your operating system is Windows, the file is `\scripts\win32\httpsetupdb.cmd`.

- If your operating system is UNIX, the file is
/scripts/unix/httpsetupdb.sh.
8. Edit the .cmd or .sh file to modify the line that starts
- If/I "WLI_HOME" == " " call...\setEnv.cmd, to set the correct path.
 9. Execute the httpsetupdb.cmd or httpsetupdb.sh command to create the HTTPPOLL database.
 10. Start the WebLogic Integration Server.
- Note:** This step assumes that you are using the default domain. If you are using a specific domain, you must run wliconfig.cmd and restart the WebLogic Integration Server after you update the relevant schema.
- Warning:** Running wliconfig.cmd for an existing database will drop all the tables and create the tables again, losing any saved data.

Deploying on WebLogic Integration 2.1 SP2

To deploy the HTTP Plug-in, you copy .jar files to your WebLogic installation directory, edit the config.xml file, copy a .sql script to the appropriate location, and run a command to create the HTTPPOLL table in your database.

If you downloaded the HTTP Plug-in, the following files are all located in the .zip file you downloaded. If you received the HTTP Plug-in on a CD, they are located on the CD.

- httpplugin-ejb.jar
- httpplugin.war
- plugin-shared.jar
- db2\hp_schema.sql
- mssql\hp_schema.sql
- pointbase\hp_schema.sql
- oracle\hp_schema.sql
- sybase\hp_schema.sql

- httpsetupdb.cmd

To deploy the HTTP Plug-in on WebLogic Integration 2.1, do the following:

1. From the source location (the directory where you unzipped the .zip file or the product CD), copy the following files into the `BEA_HOME\wlintegration2.1\lib` directory:
 - httpplugin-ejb.jar
 - httpplugin.war
 - plugin-shared.jar
2. From the `BEA_HOME\wlintegration2.1\config\samples` directory, open the `config.xml` file and add the lines in bold to the existing configuration, at the location shown here.

Note: If you have already created a specific domain, open the `config.xml` file from that domain. The scripts given here assume that the default domain is `samples`.

```
<Application
  Deployed="true"
  Name="WLI"
  Path="D:\bea\wlintegration2.1\lib">
.
.
.
  <EJBComponent
    DeploymentOrder="1"
    Name="wlpi-ejb.jar"
    Targets="myserver"
    URI="wlpi-ejb.jar"/>
  <EJBComponent
    DeploymentOrder="10"
    Name="WLI-BPM HTTP Plugin"
    Targets="myserver"
    URI="httpplugin-ejb.jar"/>
.
.
.
  <EJBComponent
    DeploymentOrder="6"
    Name="wlxtpi.jar"
    Targets="myserver"
    URI="wlxtpi.jar"/>
  <EJBComponent
    DeploymentOrder="12"
```

2 Deploying the HTTP Plug-in

```
Name="xfileplugin-ejb.jar"
Targets="myserver"
URI="xfileplugin-ejb.jar"/>
<WebAppComponent
  Name="HTTP Plugin
  EventListener"
  Targets="myserver"
  URI="httpplugin.war"/>
<WebAppComponent
  Name="WLAIPugin"
  Targets="myserver"
  URI="wlai-plugin.war"/>
</Application>
```

3. Save the file and close it.
4. Copy the `hp_schema.sql` file for your database from the source location (the directory where you unzipped the `.zip` file or product CD) to the location listed in the table that follows:

Table 2-2 Location for `hp_schema.sql` Files for Various Databases

For This Database	Copy the <code>hp_schema.sql</code> File to This Location
DB2	<code>WLI_HOME\dbscripts\db2\</code>
MS SQL	<code>WLI_HOME\dbscripts\mssql\</code>
Oracle	<code>WLI_HOME\dbscripts\oracle\</code>
Cloudscape	<code>WLI_HOME\dbscripts\cloudscape\</code>
Sybase	<code>WLI_HOME\dbscripts\sybase\</code>

5. Go the location where you unzipped the HTTP file and open the file that creates the HTTPOLL database required for the HTTP Plug-in.
 - If your operating system is Windows, the file is
`\scripts\win32\httpsetupdb.cmd`.
 - If your operating system is UNIX, the file is
`/scripts/unix/httpsetupdb.sh`.
6. Edit the `.cmd` or `.sh` file to modify the line that starts
 - `If/I "WLI_HOME" == "" call...\setEnv.cmd`, to set the correct path.

7. Execute the `httpsetupdb.cmd` or `httpsetupdb.sh` command to create the HTTPPoll database.
8. From `BEA_HOME\wlintegration2.1\config\samples` directory, open the `startWeblogic.cmd` file and edit it as shown in bold, at the following location:

```
REM WLIS data directory

if not exist %WLI_SAMPLES_HOME%\data mkdir
%WLI_SAMPLES_HOME%\data || goto finish

SET SVRCP=%SVRCP%;WLI_HOME\lib\plugin-shared.jar;

REM Start weblogic
```

9. Save the `startWeblogic.cmd` file and close it.
10. Start the WebLogic Integration Server.

Note: This step assumes that you are using the default domain. If you are using a specific domain, you must run `wlconfig.cmd` and restart the WebLogic Integration Server after you update the relevant schema.

Warning: Running `wlconfig.cmd` for an existing database will drop all the tables and create the tables again, which will lose any saved data.

Precautionary Steps to Avoid Errors During Use

The following are steps you could take to avoid exception messages or errors while working with the WebLogic Integration Studio. These steps, however, are not mandatory as part of deployment.

- When you open the WebLogic Integration Studio while connected to the WebLogic Server running on either Red Hat Linux 7.2 or Red Hat Enterprise Linux AS 2.1, you may get the following exception message:

```
"ClassNotFoundException"
```

To avoid this, edit the `studio.cmd` file on the machine running the WebLogic Integration Studio, by adding the following lines in bold, at the location shown below:

- If the machine is on Windows:

2 Deploying the HTTP Plug-in

```
set CP=WLI_HOME\lib\plugin-shared.jar;
WLI_HOME\lib\wlpi-studio.jar;
%WLICP%

set CP=%CP%;
WLI_HOME\lib\ebxml-bpm-plugin.jar;
WLI_HOME\lib\wlc-wlpi-plugin.jar;
WLI_HOME\lib\xfileplugin-ejb.jar;
WLI_HOME\lib\wlai-plugin-ejb.jar;
WLI_HOME\lib\sampleplugin-ejb.jar;
WLI_HOME\lib\wlxtpi.jar;
WLI_HOME\lib\ebxml-bpm-plugin.jar;
WLI_HOME\lib\wlai-plugin-ejb.jar;
WLI_HOME\lib\httpplugin-ejb.jar;
WLI_HOME\lib\emailplugin-ejb.jar;
WLI_HOME\lib\mdb-generator.jar;
WLI_HOME\lib\wliserver.jar;
WLI_HOME\lib\wlpi-aux.jar;
WLI_HOME\lib\wlpi-ejb.jar;
WLI_HOME\lib\wlpi-mdb-ejb.jar;

start %JAVA_HOME%\bin\javaw %COMM_CLIENT_VM% -classpath
"%CP%" "-Dwli.samples=%SAMPLES_HOME%"
"-Dwli.bpm.studio.help=WLI_HOME\docs\help"
"-Durl=http://<localhost>:<port_number>"
com.bea.wlpi.client.studio.Studio
```

- If the machine is on UNIX/Red Hat Linux:

```
CP=WLI_HOME/lib/plugin-shared.jar:
WLI_HOME/lib/wlpi-studio.jar:
$WLICP
```

```
CP=$CP:
WLI_HOME/lib/ebxml-bpm-plugin.jar:
WLI_HOME/lib/wlc-wlpi-plugin.jar:
WLI_HOME/lib/wlai-plugin-ejb.jar:
WLI_HOME/lib/sampleplugin-ejb.jar:
WLI_HOME/lib/wlxtpi.jar:
WLI_HOME/lib/ebxml-bpm-plugin.jar:
WLI_HOME/lib/wlai-plugin-ejb.jar:
WLI_HOME/lib/xfileplugin-ejb.jar:
WLI_HOME/lib/httpplugin-ejb.jar:
WLI_HOME/lib/emailplugin-ejb.jar:
WLI_HOME/lib/mdb-generator.jar:
WLI_HOME/lib/wliserver.jar:
WLI_HOME/lib/wlpi-aux.jar:
WLI_HOME/lib/wlpi-ejb.jar:
WLI_HOME/lib/wlpi-mdb-ejb.jar:
```



```
start $JAVA_HOME/bin/javaw $COMM_CLIENT_VM -classpath "$CP"
"-Dwli.samples=$SAMPLES_HOME"
"-Dwli.bpm.studio.help=WLI_HOME/docs/help"
"-Durl=http://<localhost>:<port_number>"
com.bea.wlpi.client.studio.Studio
```

- To ensure that the Start node in the Workflow Design window shows the action setting icon, from the *WLI_HOME*\integration\bin directory, open the *startServer.cmd* file from *BEA_HOME*\weblogic700\samples\integration\samples\bin, and add the line in bold, at the location shown below:

```
set CP=%CP%;WLI_HOME\lib\wlpi-ejb.jar;WLI_HOME\lib\scripts.jar;
WLI_HOME\lib\wliclient.jar
set SAMPLES_DIR=D:\bea7\weblogic700\samples\integration
```

Updating the BEA License

If you have installed WebLogic Integration Service Pack 5, the plug-in is licensed as part of WebLogic Integration. No further action is required.

If you install the plug-in with an earlier release of WebLogic Integration, you must obtain a valid software license and update your *license.bea* file as described in the following procedure. If you have downloaded the plug-in for evaluation, you must obtain an evaluation license as described on the plug-in download page. If you have purchased a license for the plug-in, the license file is typically sent to you as an e-mail attachment.

When you have obtained a valid license for the plug-in, update your *license.bea* file by completing the following steps:

1. Save the license file that you obtained with a name other than *license.bea*, in the *BEA_HOME* directory. For example, save the file as *http_plugin_license.bea*. Use this file as the *license_update_file* in step 4 of this procedure.

Warning: Do not overwrite or change the name of the existing *license.bea* file.

2. Perform the step appropriate for your platform:
 - On a Windows system, open an MS-DOS session and go to the *BEA_HOME* directory.

- On a UNIX system, go to the *BEA_HOME* directory.
3. If it is not already included, add the JDK to your *PATH* variable by executing the command appropriate to your system:
 - On a Windows system:

```
set PATH=BEA_HOME\jdk131_03\bin;%PATH%
```
 - On a UNIX system:

```
PATH=BEA_HOME/jdk131_03/bin:$PATH
export PATH
```
 4. Merge the license update file into your existing license by executing the command appropriate to your system:
 - On a Windows system:

```
UpdateLicense license_update_file
```
 - On a UNIX system:

```
sh UpdateLicense.sh license_update_file
```

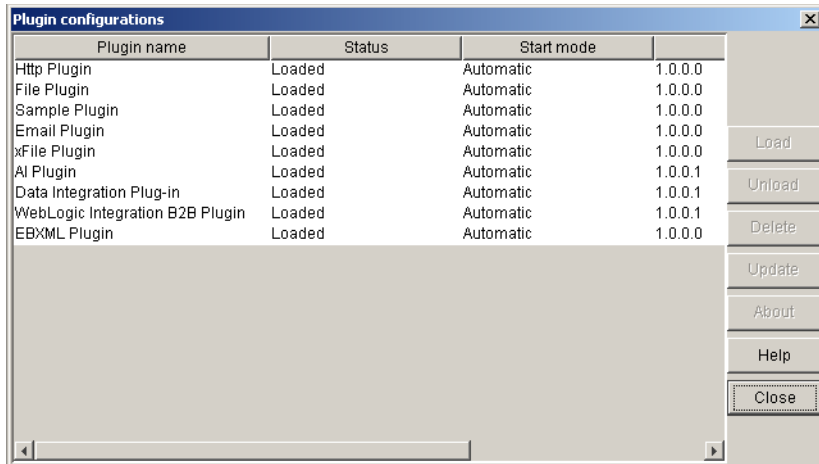
Here, *license_update_file* is the name you gave the license update file in step 1.
 5. Save a copy of your updated *license.bea* file in a safe place outside the WebLogic Integration and application installation directories.

Verifying Deployment

Once you have completed the steps for deploying the HTTP Plug-in, you need to verify whether it has been deployed correctly.

To verify the deployment, do the following:

1. Open WebLogic Integration Studio.
2. Choose Configuration→Plug-ins. The Plug-in Configurations dialog box is displayed.

Figure 2-1 Plug-in Configurations Dialog Box

3. In the Plug-in Configurations dialog box, under Plug-in name, locate HTTP Plug-in. Its presence confirms that the HTTP Plug-in has been deployed correctly.

3 Using the HTTP Plug-in

This section provides information on using the HTTP Plug-in. It includes the following topics:

- [Overview](#)
- [Sending an HTTP Request to a URL](#)
- [Starting a Workflow When an HTTP Request Arrives](#)

These topics contain step-by-step instructions for setting up the HTTP Plug-in actions. It is assumed that you already know how to design WebLogic Integration workflows. For an example of using the HTTP Plug-in to send a request to a Web server, see [Chapter 5, “HTTP Plug-in Example.”](#)

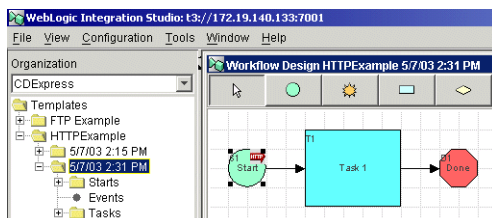
Overview

This topic provides information you need to know before using the HTTP Plug-in. It contains the following sub-topics:

- [Defining Workflow Variables](#)
- [Setting Task Properties](#)
- [Workflow Expressions](#)

You can set up the HTTP Plug-in using the Workflow Design window in the WebLogic Integration Studio, as shown in the following figure.

Figure 3-1 WebLogic Integration Studio - Workflow Design Window



In the Workflow Design window, you can construct workflows and set properties that define workflows. You must also set variable properties and task properties.

Defining Workflow Variables

You must define workflow variables before defining the workflow's task properties so that you can then bind the header and body data to the variables.

Note: For more information on variables, see the “Working with Variables” section of “[Defining Workflow Templates](#)” in *Using the WebLogic Integration Studio* at <http://edocs.bea.com/wli/docs70/studio/ch5.htm>.

To define the variables used by the workflow actions, do the following:

1. In the left pane of the WebLogic Integration Studio, double-click the Templates folder, right-click the Variables node, and select Create Variable. The Variable Properties dialog box is displayed.

Note: You can create variables only for existing templates. For details, see step 5. in the section “[Setting Up the Workflow](#),” in Chapter 3, “[Using the HTTP Plug-in](#).”

Figure 3-2 Variable Properties Dialog Box

The image shows a 'Variable Properties' dialog box. It has a title bar with a close button. Inside, there is a 'Name' text box containing 'myBin'. Below it is a 'Type' dropdown menu set to 'BinaryData'. A 'Parameter' section contains three checkboxes: 'Input' (checked), 'Output' (unchecked), and 'Mandatory' (unchecked). At the bottom is a 'Notes' text area. At the very bottom are 'OK', 'Cancel', and 'Help' buttons.

2. Set the following properties:

Table 3-1 Variable Properties

Field Name	Description	Example
Name	Enter the variable name.	myBin
Type	Select the variable type from the drop-down box.	BinaryData
Parameter	Select the relevant check boxes, depending on the purpose: <ul style="list-style-type: none"> ■ Input - To create an input variable ■ Output - To create an output variable ■ Mandatory - To make the variable mandatory 	

3. Click OK. The Variable Properties dialog box closes.

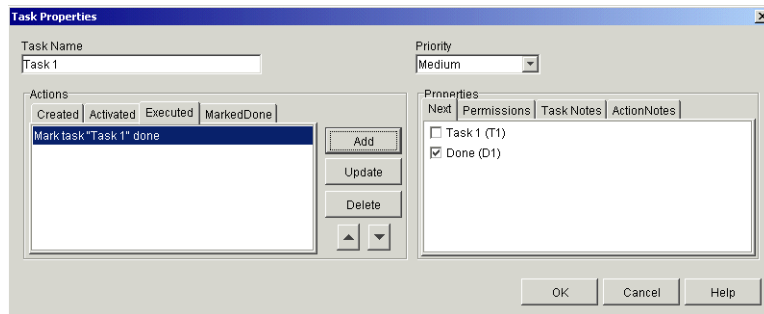
Setting Task Properties

You can set Task Properties on the Task node in the Workflow Design window.

To set workflow properties for a task, do the following:

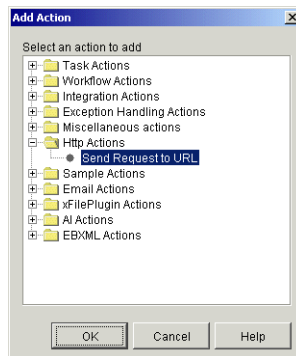
1. In the Workflow Design window, right-click a Task node, and select Properties. The Task Properties dialog box, is displayed.
2. Select the Executed tab, as shown in the following figure.

Figure 3-3 Task Properties Dialog Box



3. To add the HTTP Plug-in action, click Add. The Add Action dialog box is displayed.

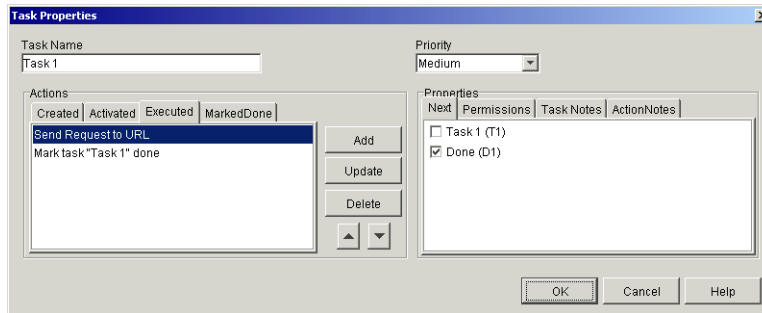
Figure 3-4 Add Action Dialog Box



4. Double-click the HTTP Actions folder, select Send Request to URL, and click OK. The Send Request to URL dialog box, where you define the properties of the action, is displayed.

Note: For details on defining the properties for the HTTP Plug-in action, see [“Sending an HTTP Request to a URL.”](#)

5. After defining the properties, click OK. The Task Properties dialog box is displayed with the plug-in action displayed on the Executed tab.
6. To make the displayed action the first to be executed, select the action and click the Up arrow. The action moves to the top position, as shown in the following figure.

Figure 3-5 Task Properties Dialog Box with Selected Action


Note: Use the Up and Down arrows to move selected tasks higher or lower in the list, depending on the workflow requirements. You can use this option when there are multiple tasks that require a sequence for their execution.

7. Click OK. The Task Properties dialog box closes.

Workflow Expressions

A workflow expression is an algebraic expression that defines a calculation that the system performs at run time, and is made up of literals, such as strings, integers and other constants, workflow variables, operators, and workflow functions. Workflow expression syntax allows you to manipulate strings, test for relationships and conditions, perform arithmetic calculations, use functions that obtain run-time information from workflows or XML messages, and so on.

The result of an expression may be a string, integer, double, date/time value, or either of the Boolean (logical) values true and false. Expressions that yield a Boolean result are referred to as conditional expressions or conditions.

The Expression button  next to a field in a Studio dialog box indicates that the field requires an entry formulated in the workflow expression language. You can either enter an expression within quotes in the field, or click the Expression button to formulate the expression.

In a Send Request to URL action, the property URL Name requires an expression.

Additionally, the following attributes may require expressions:

- Parameter Value, in Add/Delete Parameter Name and Values

- Header Value, in Add/Delete Header Name and Values

You can obtain values for these properties from the Expression Builder and XPath Wizard, which return a string value. The following values are available:

- Constant strings, such as "d:\\read\\read.xml"
- Workflow variables in String type, such as `$file_name`
- Complex expressions that return string values, such as `$a+$b+$c+"a.txt"`

For more information about expressions, see “Using Workflow Expressions” in *Using the WebLogic Integration Studio* at <http://edocs.bea.com/wli/docs70/studio/index.htm>.

Sending an HTTP Request to a URL

You can send an HTTP request to a Web server to exchange XML or non-XML/binary documents. Depending on your configuration settings, the request can be a GET or POST, it can include business data (GET) or Binary/XML data (POST), and it can include or ignore response data. You can also use variables to pass header information for POST actions or parameters values for GET actions.

In addition to instructions for sending an HTTP request, topics in this section include:

- [Communicating Via a Secure HTTP \(HTTPS\) Connection](#)
- [Capturing HTTP Response Data](#)
- [Sending Business Data as an HTTP GET](#)
- [Sending a Binary/XML Document as an HTTP POST](#)

To send an HTTP Request to a URL, do the following:

1. In the Add Action dialog box, double-click the HTTP Actions folder, select Send Request to URL, and click OK. The Send Request to URL dialog box is displayed.

Figure 3-6 Send Request to URL Dialog Box

2. Set the following properties.

Table 3-2 Send Request to URL Properties


Field Name	Description	Example
URL Name	Enter the URL name, or click  to select an expression. This field cannot be empty. The format is: <code>http://<localhost>: <port_number>/<target_url></code> For details, see “Workflow Expressions.”	<code>"http: //localhost:7001/ console"</code>
Use HTTPS Connection	Select the check box if you want to use secure HTTP connection. The SSL Parameters button will be enabled only if this is selected. This is not selected by default.	
SSL Parameters	Click the button to specify the SSL parameters. If you selected the Use HTTPS Connection check box, you must enter the SSL Parameters. For details, see “Communicating Via a Secure HTTP (HTTPS) Connection.”	

Table 3-2 Send Request to URL Properties (Continued)

Field Name	Description	Example
HTTP MODE	<p>Select POST or GET from the drop-down list.</p> <p>If you select POST, the text boxes Sending Body Data Type and Sending Body Data are enabled.</p> <p>If you select GET, the text boxes Sending Body Data Type and Sending Body Data are disabled. The button Header Values changes to Parameter Values.</p> <p>The default is POST.</p>	<p>POST</p> <p>GET</p>
Header Values	<p>Click the button to configure the header values. This opens a pop-up window where you can add or delete header names and values.</p> <p>This button changes to Parameter Values if you select GET from the HTTP MODE list.</p> <p>For details, see Step 4.</p>	<p>header name = content-type header value = text/xml</p> <p>header name = charset header value = Shift_JIS</p>
Emits through proxy server	<p>Select this check box if you are using a proxy server. The proxy server details will be enabled only if this is selected.</p> <p>This is not selected by default.</p>	
Proxy Server URL	Enter the URL of the proxy server.	172.19.144.152
Proxy Port Number	Enter the port number of the proxy server.	80
Proxy Server Userid	Enter the user ID to connect to the proxy server.	susan
Proxy Password	Enter the password to connect to the proxy server.	

Table 3-2 Send Request to URL Properties (Continued)

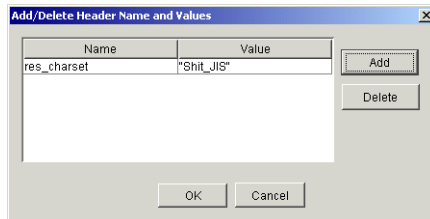
Field Name	Description	Example
Sending Body Data Type	Select XML or Binary from the drop-down list. This selection determines the type of variables displayed in the Sending Body Data drop-down list.	XML
Sending Body Data	Lists all the variables of the type available in the workflow template, depending on the selection in Sending Body Data Type. Select a variable from the drop-down list. This is enabled only if you select POST from the HTTP Mode list.	xml1
Is Response Required	Select the check box if you would like responses to HTTP requests. The response data fields are enabled only if this check box is selected. For more information, see “Capturing HTTP Response Data.”	

3. To use an HTTPS connection, select the Use HTTPS Connection check box and click SSL Parameters to set the verification and authentication parameters.

For more information, see [“Communicating Via a Secure HTTP \(HTTPS\) Connection.”](#)

4. To pass header values, in the Send Request to URL dialog box, click Header Values. The Add/Delete Header Name and Values dialog box is displayed.

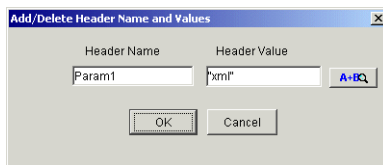
Figure 3-7 Add/Delete Header Name and Values Dialog Box



Note: If you select GET in the HTTP MODE list, the Header Values button becomes Parameter Values, and the Add/Delete Parameter Name and Values dialog box is displayed. You add a parameter name-value pair the same way you add a header name-value pair.

5. To add a header name and value, click Add. In the dialog box, enter Header Name and Header Value, and click OK. You can also click the Expression button to enter an expression.

Figure 3-8 Add/Delete Header Name and Values



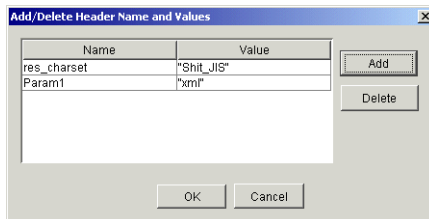
The dialog box titled "Add/Delete Header Name and Values" contains two input fields: "Header Name" with the text "Param1" and "Header Value" with the text "xml". To the right of the "Header Value" field is a button labeled "A+EQ". Below the input fields are two buttons: "OK" and "Cancel".

The entry is added to the list, as shown in the following figure.

Note:

- For a POST action where you are sending an XML document, you must define a header value that specifies the content type. The header name is Content-Type and the header value is text/xml.
- For a POST action where you are sending a Japanese XML document with Shift_JIS encoding, you must add a header that specifies the character set. The header name is charset, and the header value is Shift_JIS.

Figure 3-9 Add/Delete Header Name and Values with Added Entry



The dialog box titled "Add/Delete Header Name and Values" shows a table with two columns: "Name" and "Value". The table contains two entries: "res_charset" with value "Shift_JIS" and "Param1" with value "xml". To the right of the table are two buttons: "Add" and "Delete". Below the table are two buttons: "OK" and "Cancel".

Name	Value
res_charset	"Shift_JIS"
Param1	"xml"

6. To delete a header name and value, select it in the Name list, and click Delete.
7. Click OK when you have completed adding or deleting header names and values. The Add/Delete Header Name and Values dialog box closes.

8. To capture response data, select the Is Response Required check box and specify the variables to hold header and body data.

For more information, see [“Capturing HTTP Response Data.”](#)

9. Click OK in the The Send Request to URL dialog box.

Communicating Via a Secure HTTP (HTTPS) Connection

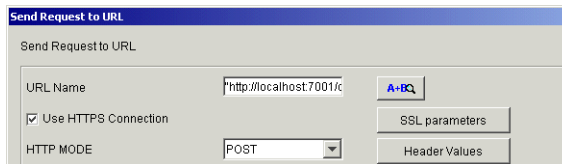
The HTTP Plug-in lets you establish a secure HTTPS connection that provides both client-side and server-side authentication. You can set verification and authentication parameters when you define the properties for an action in the Send Request to URL dialog box.

Note: To use client-side authentication, you must have a client certificate chain file and a client key file.

To communicate via a secure HTTPS connection, do the following.

1. In the Send Request to URL dialog box, select the Use HTTPS Connection check box.

Figure 3-10 Send Request to URL Dialog Box

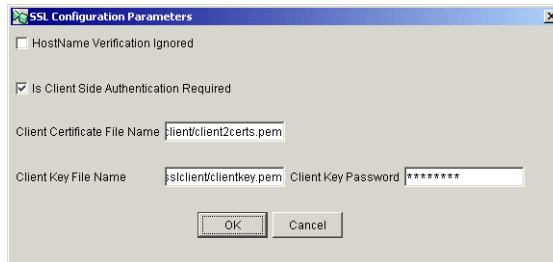


The screenshot shows a dialog box titled "Send Request to URL". Inside, there is a section titled "Send Request to URL". It contains a "URL Name" field with the text "http://localhost:7001/c". To the right of this field is a button labeled "A+BC". Below the URL field is a checkbox labeled "Use HTTPS Connection" which is checked. Below the checkbox is a dropdown menu labeled "HTTP MODE" with "POST" selected. To the right of these fields are two buttons: "SSL parameters" and "Header Values".

2. Click SSL Parameters to set the verification and authentication parameters.

The SSL Configuration Parameters dialog box is displayed.

Figure 3-11 SSL Configuration Parameters Dialog Box



3. Set the following parameters:

Table 3-3 SSL Configuration Parameters

Field Name	Description	Example
HostName Verification Ignored	Select this check box if you want to disable host name verification. By default, host name verification is enabled.	
Is Client Side Authentication Required	Select this check box if you want two-way SSL connection. The client authentication detail fields are enabled when you select this check box.	
Client Certificate File Name	Enter the path and file name of the client certificate chain file.	D:\bea\weblogic700\samples\server\src\examples\security\sslclient\client2certs.pem
Client Key File Name	Enter the path and file name of the private client key.	D:\bea\weblogic700\samples\server\src\examples\security\sslclient\clientkey.pem
Client Key Password	Enter the client key password. This field is enabled only if you have selected the Is Client Side Authentication Required check box.	(Specific to user)

4. Click OK.

Run-Time Exception for Two-Way SSL

If you encounter a run-time exception for two-way SSL, follow these instructions.

Table 3-4 Run-Time Exception for Two-Way SSL

Exception Message	Solution
<p>The server was unable to complete your request. Error in "com.bea.wlpi.HttpPlugin" plugin: "Write Channel Closed, possible SSL handshaking or trust failure". Error in "com.bea.wlpi.HttpPlugin" plugin: "Write Channel Closed, possible SSL handshaking or trust failure".</p> <p>com.bea.wli.jsp.worklist.WorklistSession\$WorklistException: The server was unable to complete your request. Error in "com.bea.wlpi.HttpPlugin" plugin: "Write Channel Closed, possible SSL handshaking or trust failure". Error in "com.bea.wlpi.HttpPlugin" plugin: "Write Channel Closed, possible SSL handshaking or trust failure".</p>	<p>By default, WebLogic Server contains a Host Name Verifier that compares the subject DN's of digital certificates and host names. When you establish an SSL connection, the subject DN of the digital certificate must match the host name of the server initiating the SSL connection. If you use the demonstration certificates, the host names will not match.</p> <p>To avoid this, edit the <code>startWeblogic.cmd</code> file as shown in bold, at the given location. This sets the flag to disable the Host Name Verifier.</p> <pre>-Dweblogic.servlet.ClasspathServlet.disableStrictCheck=true -Dweblogic.security.SSL. ignoreHostnameVerification=true</pre> <p>This solution is recommended only for development environments. A more secure solution is to obtain a new digital certificate for your WebLogic client.</p> <p>Sometimes, the WebLogic server is unable to locate the Trusted CA key store. In this situation, edit the <code>startWeblogic.cmd</code> file, as shown in bold, right after the Host Name Verification flag:</p> <pre>-Dweblogic.security.SSL. trustedCAKeyStore=WLI_HOME\lib\cacerts :finish endlocal</pre>

Capturing HTTP Response Data

When you send a request to a URL, the response contains both header data and body data. If you wish, you can ignore response data. However, if you choose to capture it, you must specify separate variables where the header and body data are to be stored. Header data is always XML; body data can be either binary or XML.

To capture HTTP response data, do the following:

1. In the Send Request to URL dialog box, select the Is Response Required check box. The response data fields are enabled.

Figure 3-12 Send Request to URL Dialog Box

The screenshot shows a dialog box titled 'Send Request to URL'. It has a checkbox labeled 'Is Response Required' which is checked. Below this, there are two rows of configuration options. The first row is 'Receiving Header Data' with a dropdown menu showing 'xml2'. The second row is 'Receiving Body Type' with a dropdown menu showing 'XML'. To the right of these, there is a label 'Receiving Body Data' followed by a dropdown menu showing 'xml3'.

2. Set the following parameters:

Table 3-5 HTTP Response Configuration Parameters

Field Name	Description	Example
Receiving Header Data	Lists all variables of type XML available in the workflow template. Select a variable from the drop-down list. You cannot choose the same variable for Receiving Body Data.	xml2
Receiving Body Type	Select XML or Binary from the drop-down list. This selection determines the type of variables displayed in the Receiving Body Data drop-down list.	XML

Table 3-5 HTTP Response Configuration Parameters (Continued)

Field Name	Description	Example
Receiving Body Data	<p>Lists all the variables of the type available in the workflow template, depending on the selection in Receiving Body Type.</p> <p>Select a variable from the drop-down list. You cannot choose the same variable for Receiving Header Data.</p> <p>Note: If the Receiving Body Data consists of foreign language characters, set the corresponding charset header in the response.</p>	<p>xml3</p> <p>If the foreign language is Japanese, set the corresponding charset header in the response as follows:</p> <p>header name = charset</p> <p>header value = Shift_JIS</p>

Note: The HTTP Plug-in stores the Header Data Content in the workflow variable in the following XML format:

```
<HEADERDATA>
<HEADER>
  <NAME></NAME>
  <VALUE></VALUE>
</HEADER>
.
.
.
</HEADERDATA>
```

Sending Business Data as an HTTP GET


You can send a GET request to a URL that contains business data by specifying the URL, selecting GET as the HTTP MODE, and specifying parameters. If you wish, you can capture response data in variables. You use the Send Request to URL dialog box to specify the settings.

These are the settings for sending business data as an HTTP GET.

Figure 3-13 Send Request to URL Settings for HTTP GET

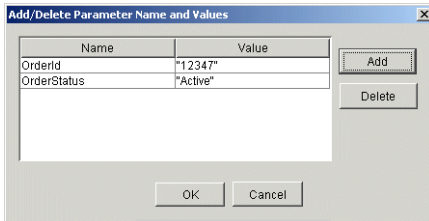
Set the following properties.

Table 3-6 Send Request to URL Properties

Field Name	Description	Example
URL Name	Enter the URL name, or click  to select an expression. The format is: <code>http://<localhost>: <port_number>/<target_url></code> For details, see “Workflow Expressions.” .	<code>“http: //localhost:7001/ console”</code>
Use HTTPS Connection	Select the check box if you want to use secure HTTP connection. For details, see “Communicating Via a Secure HTTP (HTTPS) Connection.” .	
HTTP MODE	Select GET from the drop-down list.	GET

These are sample parameter settings for sending business data in an HTTP GET. The parameters you enter depend on your business data. To set parameters, click Set Parameter Values and see Step 4. in the section [“Sending an HTTP Request to a URL.”](#)

Figure 3-14 Sample HTTP GET Parameters



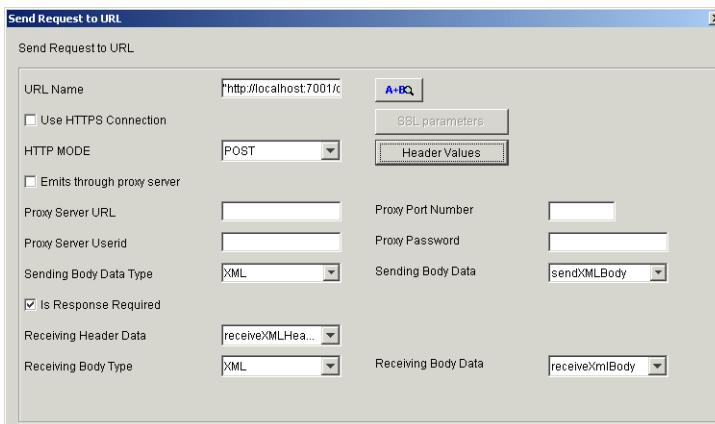
If you wish to capture response data, see [“Capturing HTTP Response Data.”](#)

Sending a Binary/XML Document as an HTTP POST

You can send a POST request to a URL that contains a binary/XML document by specifying the URL, selecting POST as the HTTP MODE, specifying header names and values, and specifying the body data type and variable. If you wish, you can capture response data in variables. You use the Send Request to URL dialog box to specify the settings.


These are the settings for sending binary/XML document as an HTTP POST.

Figure 3-15 Send Request to URL Settings for HTTP POST



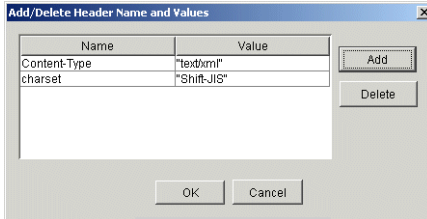
Set the following properties.

Table 3-7 Send Request to URL Properties

Field Name	Description	Example
URL Name	Enter the URL name, or click  to select an expression. The format is: <code>http://<localhost>: <port_number>/<target_url></code> For details, see “Workflow Expressions.”	<code>"http: //localhost:7001/ console"</code>
Use HTTPS Connection	Select the check box if you want to use secure HTTP connection. For details, see “Communicating Via a Secure HTTP (HTTPS) Connection.”	
HTTP MODE	Select POST from the drop-down list.	POST
Sending Body Data Type	Select XML or Binary from the drop-down list. This selection determines the type of variables displayed in the Sending Body Data drop-down list.	XML
Sending Body Data	Lists all the variables of the type available in the workflow template, depending on the selection in Sending Body Data Type. Select a variable from the drop-down list.	xml1

These are sample header settings for sending a binary or XML document in an HTTP POST.

Figure 3-16 HTTP Post Parameters



Note:

- If you are sending an XML document, the Content-Type header is required.
- If you are sending a Japanese document with Shift_JIS encoding, the charset header is required.

To set headers, click Set Header Values and see Step 4.

If you wish to capture response data, see [“Capturing HTTP Response Data.”](#)

Starting a Workflow When an HTTP Request Arrives

You can start a workflow whenever an HTTP request arrives by defining properties for the Start node, and sending a parameter Header Id to the event listener. The event listener, a servlet, takes the request, checks for the Header Id, and starts the workflow when it finds a match.

Notes:

- You must define at least two workflow variables before defining the workflow’s Start properties. This is because the header data and body data content are bound to workflow variables.
- Each request can start only one workflow instance.

To start a workflow when an HTTP request arrives, do the following:

1. In the Workflow Design window, right-click the Start node, and select Properties. The Start Properties dialog box is displayed.
2. Click the Event option button, and select Start Http from the Event drop-down list. The fields pertaining to the Start Http properties appear, as shown in the following figure.

Figure 3-17 Start Properties Dialog Box

3. Set the following properties:

Table 3-8 Start Properties

Field Name	Description	Example
Header Id	Enter the unique header ID, based on the business data being sent.	126
Header Data Value	Select the name of the variable to hold the header data content from the drop-down list. The list contains all variables defined for this header ID. For details on the header data content format, see the following note.	xml2
Body Data Type	Select the body data type from the drop-down list.	XML/Binary
Body Data Value	Select the name of the variable to hold the body data content from the drop-down list.	xml3

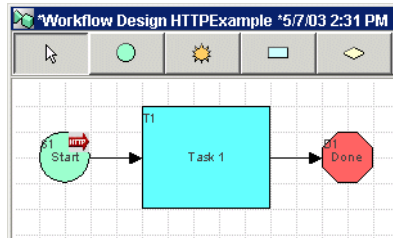
Note: The HTTP plug-in stores the Header Data Content in the workflow variable in the following XML format:

```
<HEADERDATA>
  <HEADER>
    <NAME></NAME>
    <VALUE></VALUE>
  </HEADER>
...
</HEADERDATA>
```

4. From the Start Organization drop-down list, select the relevant organization and click OK. The Start Properties dialog box closes.

After the properties are set, the Start node on the Workflow Design window indicates the action setting, as shown in the following figure.

Figure 3-18 Start Node with HTTP Action



5. To start the workflow, send the HTTP request through the Send Request to URL action. For details, see [“Sending an HTTP Request to a URL” on page 3-6](#).

For example, enter the following in the URL Name field to send the request with the Header Id of 126:

```
http://host:port/context_root/HttpEventListener?HEADERID=126
```

Here, *host* is the host name or IP address of the server, *port* is the listen port, and *context_root* is the context root defined for `httpplugin.war` in the `WLI_HOME/lib/META-INF/application.xml` file. In WebLogic Integration 7.0 SP5, *context_root* is `HttpPlugin`. If you installed the plug-in with an earlier release, *context_root* is `com.bea.wlpi.HttpPlugin`.

Note: Do not select the Is Response Required check box.

4 Configuring the HTTP Plug-in for a Migrated Domain

This section describes how to update your database schema and configure the HTTP Plug-in for a single server domain and a clustered domain. It includes the following topics:

- [Updating the BPM Database Table](#)
- [Migrating to a Single Server Domain](#)
- [Migrating to a Clustered Domain](#)

Updating the BPM Database Table

The HTTP Plug-in uses a new database table called `HTTPOLL`. To update the BPM database with this table, edit the following file by appending the contents of `hp_schema.sql`:

```
BEA_HOME\weblogic700\integration\dbscripts\<database_type>\  
migrate\BPM_70-70SP2.sql
```

Note: The contents of `hp_schema.sql` will depend on your database.

Migrating to a Single Server Domain

To configure the HTTP Plug-in for a single server domain, you must deploy both `httpplugin-ejb.jar` and `httpplugin.war` as components of the WebLogic Integration application.

1. To deploy `httpplugin-ejb.jar`, edit the domain's `config.xml` file by adding the following lines:

```
<EJBComponent
  Name="HTTP BPM Plug-in"
  Targets="<myserver>"
  URI="httpplugin-ejb.jar"/>
```

2. To deploy `httpplugin.war`, edit the domain's `config.xml` file by adding the following lines:

```
<WebAppComponent
  Name="Http BPM Plug-in Help"
  Targets="myserver"
  URI="httpplugin.war"/>
```

Migrating to a Clustered Domain

To configure the HTTP Plug-in for a clustered domain, you must deploy both `httpplugin-ejb.jar` and `httpplugin.war` on the cluster server. You must also provide the URL for the managed servers.

Note: This example refers to a cluster system called `mycluster`.

1. To deploy `httpplugin-ejb.jar`, edit the cluster's `config.xml` file by adding the following lines:

```
<EJBComponent
  Name="HTTP BPM Plug-in"
  Targets="mycluster"
  URI="httpplugin-ejb.jar"/>
```

2. To deploy `httpplugin.war`, edit the cluster's `config.xml` file by adding the following lines:

```
<WebAppComponent  
  Name="Http BPM Plug-in Help"  
  Targets="mycluster"  
  URI="httpplugin.war"/>
```

3. To provide the URL for the managed servers, add the following in the `startManagedServer.cmd` file:

```
-Durl=t3://<localhost>:<port_name>
```

4 *Configuring the HTTP Plug-in for a Migrated Domain*

5 HTTP Plug-in Example

This section gives a step-by-step example of using the HTTP Plug-in. It includes the following topics:

- [Setting Up the Workflow](#)
- [Executing the Workflow](#)

This example illustrates how to use the HTTP Plug-in to send an HTTP request to a Web server. The first part of the example tells you how to set up a workflow in the WebLogic Integration Studio, that when executed, will send an HTTP request to a URL. The second part of the example tells you how to use the WebLogic Integration Worklist to execute the workflow, sending the HTTP request to the specified URL.

Note: This example uses the WebLogic Integration `samples` Domain. For more information, see the “Configuring and Starting the Samples Domain” section of “[Getting Started](#)” in *Starting, Stopping, and Customizing BEA WebLogic Integration* at <http://edocs.bea.com/wli/docs70/config/getstart.htm>.

Setting Up the Workflow

To set up a workflow in WebLogic Integration Studio to send an HTTP request to a Web server, do the following:

1. Start the WebLogic Integration Server.
2. To open the WebLogic Integration Studio, do one of the following:
 - On Windows, for WebLogic Integration 7.0, select Start→Programs→BEA WebLogic Platform 7.0→WebLogic Integration 7.0→Studio.

- On Windows, for WebLogic Integration 2.1, select Start→Programs→BEA WebLogic E-business Platform→WebLogic Integration 2.1→Studio.

- On UNIX, for WebLogic Integration 7.0, run

```
BEA_HOME/weblogic700/integration/bin/studio.sh
```

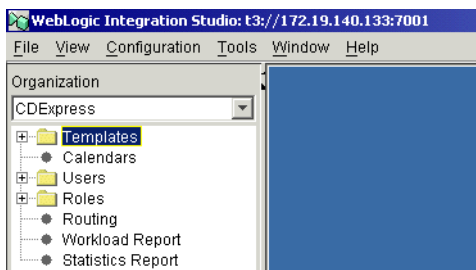
- On UNIX, for WebLogic Integration 2.1, run

```
BEA_HOME/wlintegration2.1/bin/studio.sh
```

The Logon to WebLogic Integration dialog box is displayed.

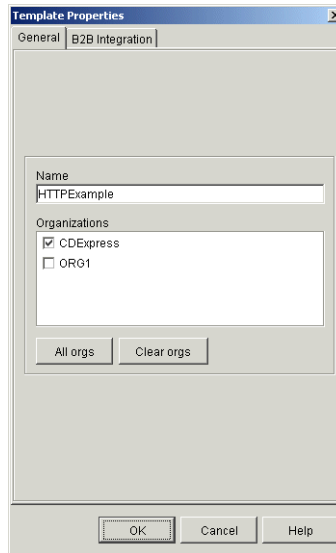
3. Enter the User Name, Password, and Server URL, and click OK. You are connected to the WebLogic Server, and the WebLogic Integration Studio is displayed, as shown in the following figure.

Figure 5-1 WebLogic Integration Studio



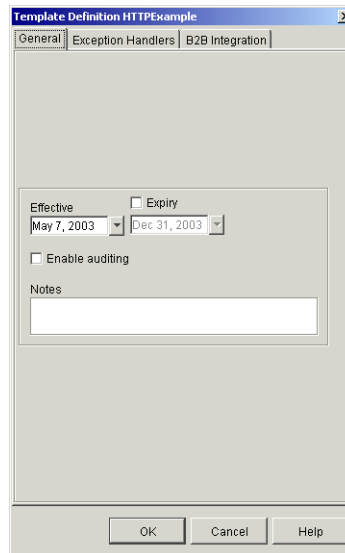
4. In the left pane of WebLogic Integration Studio, select CDEExpress from the Organization drop-down list.
5. In the left pane, right-click the Templates folder and select Create Template. The Template Properties dialog box is displayed.

Figure 5-2 Template Properties Dialog Box



6. On the General tab, in the Name field, enter HTTPExample and click OK. The Template Properties dialog box closes. The new template HTTPExample is added to the Templates folder in the WebLogic Integration Studio.
7. In the left pane of the WebLogic Integration Studio, double-click the Templates folder, right-click the HTTPExample Template, and select Create Template Definition. The Template Definition HTTPExample dialog box is displayed.

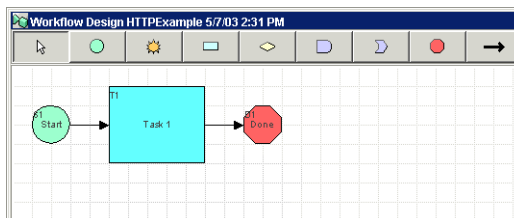
Figure 5-3 Template Definition HTTPExample Dialog Box



8. On the General tab, do one of the following:
 - To specify a different expiry date for the workflow, select the Expiry check box, and select the desired date from the calendar drop-down list.
 - To retain the default expiry date, click OK.

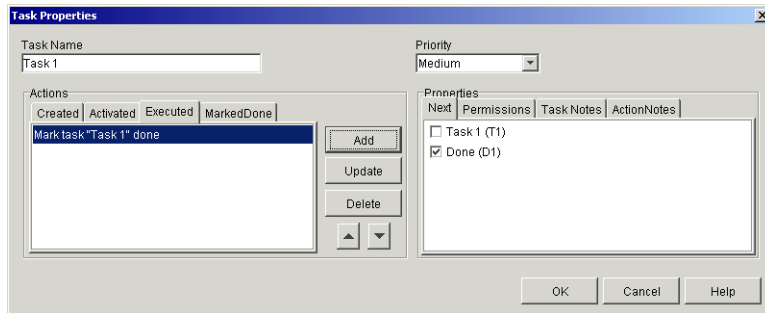
The Template Definition appears inside the HTTPExample folder, displaying the creation date and time. The Workflow Design window is displayed in the right pane, as shown in the following figure.

Figure 5-4 Workflow Design – HTTPExample Window



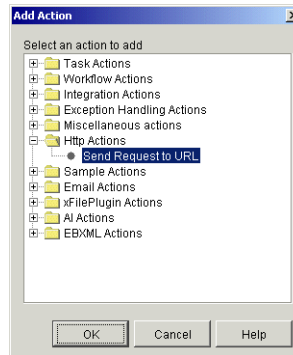
9. Right-click Task 1 and choose Properties. The Task Properties dialog box is displayed. Select the Executed tab, as shown in the following figure.

Figure 5-5 Task Properties Dialog Box



10. Click Add. The Add Action dialog box is displayed.

Figure 5-6 Add Action Dialog Box



11. Double-click the HTTP Actions folder, select Send Request to URL, and click OK. The Send Request to URL dialog box, where you define the properties of the action, is displayed.

Figure 5-7 Send Request to URL Dialog Box

12. Set the following properties:

Table 5-1 Send Request to URL Properties

Field Name	Description	Example
URL Name	Enter the URL name. This field cannot be empty.	"http://localhost:7001/console"
HTTP MODE	Select POST or GET. If you select POST, the text boxes Sending Body Data Type and Sending Body Data are enabled. If you select GET, the text boxes Sending Body Data Type and Sending Body Data are disabled. The button Header Values changes to Parameter Values. The default is POST.	POST
Header Values	Click the button to configure the header values. This opens a pop-up window where you can add or delete header names and values. This button changes to Parameter Values if you select GET from the HTTP MODE list.	

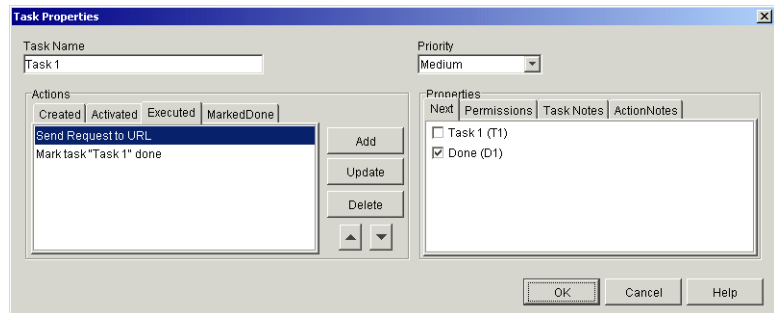
Table 5-1 Send Request to URL Properties (Continued)

Field Name	Description	Example
Sending Body Data Type	Select XML or Binary from the drop-down list. This selection determines the type of variables displayed in the Sending Body Data drop-down list.	XML
Sending Body Data	Lists all the variables of the type available in the workflow template, depending on the selection in Sending Body Data Type. Select a variable from the drop-down list. This is enabled only if you select POST from the HTTP Mode list.	xml1
Is Response Required	Select the check box if you would like responses to HTTP requests. The response data fields are enabled only if this check box is selected.	Select the check box.
Receiving Header Data	Lists all variables of type XML available in the workflow template. Select a variable from the drop-down list. You cannot choose the same variable for Receiving Body Data.	xml2
Receiving Body Type	Select XML or Binary from the drop-down list. This selection determines the type of variables displayed in the Receiving Body Data drop-down list.	XML
Receiving Body Data	Lists all the variables of the type available in the workflow template, depending on the selection in Receiving Body Type. Select a variable from the drop-down list. You cannot choose the same variable for Receiving Header Data.	xml3

13. Click OK. The Task Properties dialog box is displayed with the Send Request to URL action displayed on the Executed tab.

14. To make the Send Request to URL action the first to be executed, select it and click the Up arrow. The action moves to the top position, as shown in the following figure.

Figure 5-8 Task Properties Dialog Box with Send Request to URL Action



15. Click OK. The Task Properties dialog box closes.
16. In the left pane of WebLogic Integration Studio, right-click the Template Definition folder, and choose Properties. The Template Definition HTTPExample dialog box, with the Active check box, is displayed.

Figure 5-9 Template Definition HTTPExample Dialog Box with Active Check Box



17. Select the Active check box and click OK.

18. In the left pane, right-click the Template Definition and select Save.

Note: An asterisk before the definition name indicates that the changes to that folder have not been saved.

Executing the Workflow

In this part of the example, the WebLogic Integration Worklist executes the workflow, sending the HTTP request to the specified URL. This part of the example provides information on the following:

- [Executing on WebLogic Integration 7.0](#)
- [Executing on WebLogic Integration 2.1](#)

For more information about the WebLogic Integration Worklist, see “Using the WebLogic Integration JSP Worklist” at <http://edocs.bea.com/wli/docs70/jspwlist/index.htm>.

Executing on WebLogic Integration 7.0

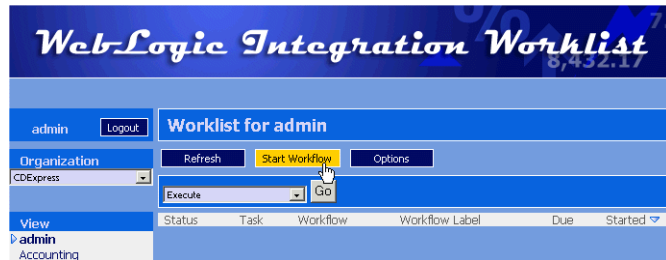
1. To start the WebLogic Integration Worklist, do one of the following:
 - On a Windows system, select Start→Programs→BEA WebLogic Platform 7.0→WebLogic Integration 7.0→Worklist.
 - On a UNIX system, open a browser and enter the following URL:

`http://localhost:7001/worklist`

The WebLogic Integration Worklist Login window is displayed.

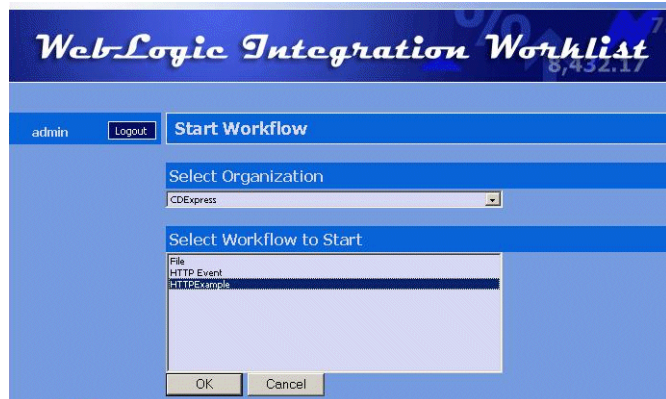
2. Enter the User Name and Password, and click OK. The WebLogic Integration Worklist Main window, is displayed.

Figure 5-10 WebLogic Integration Worklist Main Window



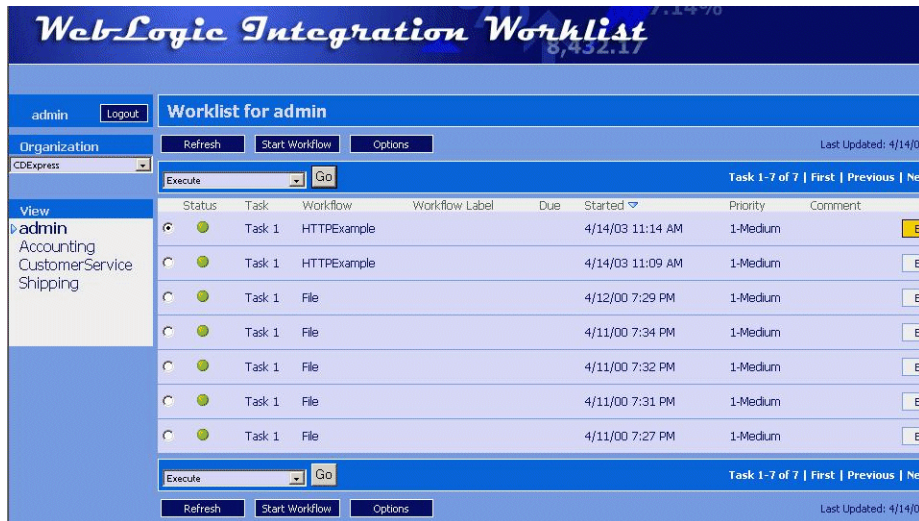
3. Click Start Workflow. The WebLogic Integration Worklist Start Workflow window is displayed.

Figure 5-11 WebLogic Integration Worklist Start Workflow Window



4. From the Organization drop-down list, select CDExpress.
5. In Select Workflow to Start list, select HTTPExample, and click OK. The WebLogic Integration Worklist Window opens with the new task, as shown in the following figure.

Figure 5-12 WebLogic Integration Worklist Window with New Task



6. Click the option button next to the new task and click Execute. The request is sent to the specified URL and the task disappears from the worklist.

Executing on WebLogic Integration 2.1

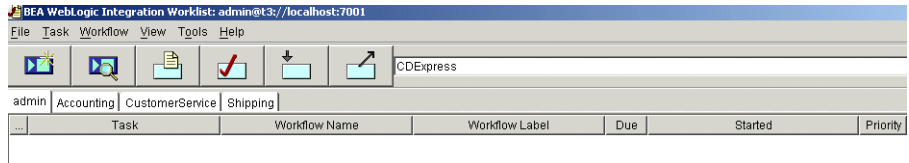
1. To start the WebLogic Integration Worklist, do one of the following:
 - On a Windows system, select Start→Programs→BEA WebLogic E-Business Platform→WebLogic Integration 2.1→Worklist.
 - On a UNIX system, run

```
BEA_HOME/wlintegration 2.1/bin/worklist
```

The Logon to WebLogic Integration dialog box is displayed.

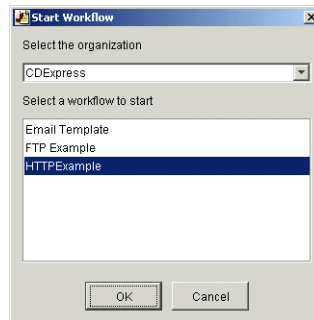
2. Enter the User Name, Password, and Server URL, and click OK. The WebLogic Integration Worklist window is displayed.

Figure 5-13 WebLogic Integration Worklist Window



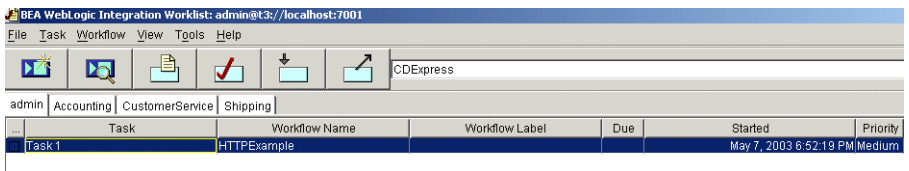
3. From the Organization drop down list, select CDEExpress.
4. Choose Workflow→Start a Workflow. The Start Workflow dialog box is displayed.

Figure 5-14 Start Workflow Dialog Box



5. In the Select a Workflow to Start list, select HTTPExample, and click OK. The WebLogic Integration Worklist window is displayed with the task, as shown in the following figure.

Figure 5-15 WebLogic Integration Worklist Window with Task



6. Right-click the task and select Execute. The request is sent to the specified URL and the task disappears from the worklist.

Index

A

- About the BEA WebLogic HTTP Plug-in 1-1
- about this document v
- add action 3-4

B

- BEA
 - contacting -vii
 - license, updating 2-11
 - Product Documentation vi
 - WebSupport vii
- BPM database table, updating 4-1

C

- clustered domain, migrating to 4-2
- conventions, documentation viii

D

- database schema, updating 4-1
- deploy HTTP Plug-in
 - on WebLogic Integration 2.1 2-6
 - on WebLogic Integration 7.0 2-2
- document, printing vi
- documentation conventions viii
- domains, migrating to 4-2

E

- e-docs web site vi
- exception, two-way SSL 3-13

- executing, workflow 5-9
- Expression button 3-5
- expressions, workflow 3-5

H

- header name and values
 - adding 3-9
 - deleting 3-9
- HTTP Plug-in
 - configuring for a migrated domain 4-1
 - deploying 2-2
 - introducing 1-1
 - overview 3-1
 - send HTTP request to a URL 3-6
 - service 3-6
 - setting up 3-1
 - using 3-1
 - verifying deployment 2-12

L

- license, BEA, updating 2-11

M

- migrate to
 - clustered domain 4-2
 - domains 4-2
 - single server domain 4-2

P

- plug-in configurations window 1-3

- prerequisites 1-2
- print product documentation vi
- properties
 - task 3-3
 - variable 3-2

R

- related documents vii

S

- send HTTP request 3-6
- single server domain, migrating to 4-2
- start node, with HTTP action 3-21
- start workflow 3-19
- support, technical vii

T

- task properties, setting 3-3
- template
 - creating 5-2
 - properties 5-2
- template definition, creating 5-3
- template, creating 5-2
- two-way SSL, run-time exception 3-13

V

- variables, workflow 3-2
- verify deployment 2-12

W

- WebLogic Integration Samples Domain 5-1
- WebLogic Integration Server, starting 5-1
- WebLogic Integration Studio
 - opening 5-1
 - Workflow Design window 3-1
- workflow
 - executing 5-9
 - executing on WebLogic Integration 2.1 5-11

- executing on WebLogic Integration 7.0 5-9
- expressions 3-5
- instance 3-19
- setting up 5-1
- start node 3-21
- Workflow Design window 3-3
- workflow variables, defining 3-2
- worklist 5-9

X

- XPath Wizard 3-6