BEA WebLogic Workshop Release Notes

BEA WebLogic Workshop
Date: February 2003

This document includes the following topics:

For updated release note information, go to the BEA documentation Web site at the following URL:

http://edocs.bea.com

Platform Support and System Requirements

For information on platform support, including hardware and software requirements, see the Supported Platforms page at the following location:

http://edocs.bea.com/platform/docs70/support/index.html

Upgrading from WebLogic Server 7.0 Release

This note applies if you are upgrading to the Service Pack 2 release of WebLogic Platform 7.0 from the release of WebLogic Server 7.0 (in which WebLogic Workshop was not included). You must use the upgrade installer, rather than Smart Update, in order to receive all of the components required by the upgrade release. Note that you can install Service Pack 2 without first installing Service Pack 1.

Migrating Domains Created Using the Configuration Wizard

The Configuration Wizard (introduced in WebLogic Platform 7.0) allows you to create new domains quickly and easily. If you created domains using the Configuration Wizard in WebLogic Platform 7.0, you need to migrate those domains for use with WebLogic Platform 7.0 Service Pack 2.

For most domains, migration is a three-step process:

1. Upgrade the product JAR files in the domain directory. A migration script is provided for this purpose.

Note: You can also revert a domain to its pre-migration state.

2. Update the domain to support Service Pack 2 changes. Depending on the domain template used to generate the domain, you may need to add or modify existing scripts or files.
3. If you installed WebLogic Platform 7.0 Service Pack 2 into a new directory, separate from the WebLogic Platform 7.0 installation, update the domain startup scripts and configuration files to reference the new *BEA_HOME* directory location.

   Note: If you *upgraded* your existing WebLogic Platform 7.0 installation, you can skip this step.

These steps are explained in detail in the following sections. You will need to repeat this process for *each* domain that you want to migrate.

Note: This section decribes how to migrate domains specific to WebLogic Workshop. For information about migrating other WebLogic Platform domains, see "Migrating Domains Created Using the Configuration Wizard" in the *WebLogic Platform 7.0 Service Pack 2 Release Notes* at the following URL:

http://e-docs.bea.com/platform/docs70/relnotes/relnotes.html#migration

Step 1: Upgrade Product JAR Files

To upgrade product JAR files for a domain that you generated using the Configuration Wizard to Service Pack 2, navigate to the *BEA_HOME*`\weblogic700\server\bin` directory and execute one of the following commands:

Windows: `migrate.cmd` *`domain mode`*
UNIX: `migrate.sh` *`domain mode`*

Note: You will be prompted to press any key to start processing.

The following table defines the command-line arguments.

Argument	***Description***
domain	Full pathname of the domain directory.
mode	Migration mode. The mode can be set to one of the following values: `upgrade` — Upgrade the product JAR files in the domain directory, as required. The original product JAR files are saved as *.jar.orig. If the timestamp of an existing product JAR file is more recent than the timestamp on the corresponding SP1 installation product JAR file, the file is skipped. This is the default mode. `revert` — Reverts a domain that was migrated earlier using the backup files (*.jar.orig) generated. If no *.jar.orig files exist, the command is ignore.

For example, the following command upgrades a domain called `mydomain` located in the default user projects directory (*BEA_HOME*`\user_projects`):

Windows: `migrate.cmd c:\bea\user_projects\mydomain upgrade`
UNIX: `migrate.sh c:/bea/user_projects/mydomain upgrade`

The following command reverts the changes made to `mydomain` during the migration:

Windows: `migrate.cmd c:\bea\user_projects\mydomain revert`
UNIX: `migrate.sh c:/bea/user_projects/mydomain revert`

Step 2: Update Domain to Support Service Pack 2 Changes

To update a domain that is based on the WebLogic Workshop template to support WebLogic Platform 7.0 Service Pack 2, perform the following steps.

Note: Before adding or modifying any files, it is recommended that you backup the original files.

1. Modify the `startWebLogic.cmd` (Windows) or `startWebLogic.sh` (UNIX) command to reflect the appropriate PointBase version (183 versus 172) in the JAR filenames defined in the CLASSPATH. The files for both commands are located in the following directory, by default:

   BEA_HOME\user_projects*domain*

   The following sample excerpt from the `startWebLogic.cmd` script (Windows) shows the required updates in **bold**:

   Before:

   ```
   set PB_CLASSPATH=
   %POINTBASEDIR%\eval\pointbase\lib\pbserver42ECF172.ja
   r;
   %POINTBASEDIR%\eval\pointbase\lib\pbclient42ECF172.ja
   r
   ```

   After:

   ```
   set PB_CLASSPATH=.\;
   %POINTBASEDIR%\eval\pointbase\lib\pbserver42ECF183.j
   ar;
   %POINTBASEDIR%\eval\pointbase\lib\pbclient42ECF183.j
   ar
   ```

2. Copy the following files from the *BEA_HOME*\weblogic700\samples\workshop directory to the BEA_HOME\user_projects*domain* directory of your WebLogic Workshop domain. Be careful not to overwrite any files that may have been created using one of these filenames.

   ```
   setWorkshopEnv.cmd
   setWorkshopEnv.sh
   startPointBaseConsole.cmd
   startPointBaseConsole.sh
   URLs.dat
   ```

Step 3: Update Startup Scripts and Configuration Files to Reference New BEA_HOME Directory Location (Non-Upgrades Only)

Note: This step is only required if you installed WebLogic Platform 7.0 Service Pack 2 into a new directory that is separate from the WebLogic Platform 7.0 installation. If you *upgraded* your existing WebLogic Platform 7.0 installation, you can skip this step.

The domain startup scripts (such as, `startWebLogic`) and configuration files (such as `config.xml`) define the full pathnames to files within the *BEA_HOME* directory. You need to search for and update these full pathnames to reference the new *BEA_HOME* directory location. In addition, you must update any custom scripts, such as build scripts, that define full pathnames to the files within the *BEA_HOME* directory to reflect the new *BEA_HOME* location.

Note: Many startup scripts set environment variables in your current shell, including variables that reference your *BEA_HOME* directory. After updating the *BEA_HOME* references in script files, you should open a new shell to ensure that the latest environment settings are used.

Setting the Maximum Size of Conversation IDs

Conversational web services use a database to store conversation IDs - the unique strings that identify conversations. The database table used to store conversational state is created if it doesn't exist upon the first use of a service. The maximum size for conversation IDs is built into the database schema. The default maximum size allowed for conversation IDs is 768 characters. Ordinarily, 768 works well as a maximum. However, if the design of a particular service requires a different maximum length, you can change it in two ways as described in this note.

First, you must set the weblogic.jws.ConversationMaxKeyLength property. This property can be set in either of the two following ways. Both of these require that you restart the server after making the change:

Edit the startWebLogic command file

1. Locate one of the following files, depending on which operating system you are using: startWebLogic.cmd (on Windows) or startWebLogic.sh (on UNIX or Linux).

   This file is located in the domain directory of your project. For example, for the samples project installed with WebLogic Workshop, this is the samples\workshop directory.

2. Locate the following text in the file:

- On Windows, find the line with the following text: @set JAVA_OPTIONS=%JAVA_OPTIONS% %JAVA_PROPERTIES%

- On Linux or UNIX, find the first line with the following text:
  JAVA_PROPERTIES=

3. Append the following to the line, replacing <numberOfCharacters> with the new maximum length:

```
-
Dweblogic.jws.ConversationMaxKeyLength=<numberOfCharact
ers>
```

Edit the jws-config.properties file

1. Locate the jws-config.properties file.

   This file is located in the domain directory of your project. For example, for the samples project installed with WebLogic Workshop, this is the samples\workshop directory.

2. Change the value of the weblogic.jws.ConversationMaxKeyLength property. The syntax is as follows:

```
weblogic.jws.ConversationMaxKeyLength=<numberOfCharacters>
```

After restarting the server, you must use your service (for example, by calling one of its methods) so that the conversational state table will be recreated with the new maximum conversation ID length. In the event that the conversational state tables already exist, you will need to drop them prior to referencing the service so that they will be recreated. After dropping the tables, and restarting the server, the first reference to your service will cause the tables to be recreated with the new value for the maximum conversation ID length.

Note that a different table is used for each service.

Known Limitations

Working with Databases

Out of Memory Error for Large Result Sets

You may receive an "Out of memory" error when retrieving a large ResultSet using a Database control. This is because the query result is converted to HTML, which more than doubles its size.

Workaround: Try to use queries that return smaller ResultSets or return an Iterator.

"Invalid Transaction State" When Using Pointbase

If you are running Weblogic Workshop with the PointBase database (the default configuration for Workshop), and you encounter an error stating that the database is in an "invalid transaction state", your PointBase database files have become corrupted and you must replace them.

Workaround: Replace cajun.dbn and cajun$1.wal, two PointBase files included with Weblogic Workshop samples. These are located in the WebLogic Workshop samples domain directory. Usually this directory is at /bea/weblogic700/samples/workshop. If you have made back up copies of these two files, you can replace the two files in use with your clean back ups.

You can also download clean copies of the files from the bug fixes section of the resource library at dev2dev Online. On the bug fixes page, look for change request (CR) number CR080632, then click the name of the bug to reach a page from which you can download the files. Note that any information you had stored in your PointBase database will be lost if you replace the files with the clean files downloaded from the web.

Editing Code

CPU Peaks and Computer Hangs When Using the Find Feature

If you are using the Find command to search a file that is 50 KB or larger, you may find that your computer hangs. This can occur if the file contains a very long line of text. For example, if you create a CTRL file from a WSDL file that lacks line breaks (as some do), the WSDL text embedded in the CTRL file will result in a very long line. Using the Find feature on such a file may hang your computer.

Workaround: Ensure that lines of code are properly broken before using the Find feature on a file.

Testing and Debugging

Problems When Using Netscape 6.2

- If you test a JWS file and your browser is Netscape 6.2, the browser may hang in the splash screen when the browser is launched (as when you select "Start" or "Start and Debug" from WebLogic Workshop). This is a bug in Netscape 6.2 that is tracked in Bugzilla bug numbers 54701 and 54716.
- If you test a JWS file in a Netscape 6.2 browser window and try to use the Callback WSDL link on the Overview page, you will get a blank page instead of a generated WSDL.

Workaround: Use a different browser.

WebLogic Server Does Not Respond While Debugging Multiple Service Instances

When debugging a web service using the breakpoints and the Step Into command, WebLogic Server can reach a state in which it no longer responds to commands. This can happen if you run multiple instances of your web service with breakpoints set. For example, you might inadvertently double-click the Test View button corresponding to a method of your service, then use the Step Into command twice (through the Debug menu or toolbar button).

Workaround: If you reach a state in which your project will no longer build, stop and restart WebLogic Server.

Execution Sometimes Moves to Unexpected Locations When Stepping Through Code

In the following situations, you might find that the debugger takes execution to unexpected lines in your code:

- When using the Step Out command, execution moves to the first line of the calling procedure, rather than the line following the call.
- When stepping through a try/catch/finally block, execution moves to the try statement after the catch and finally statements finish executing.
- When stepping through inner class constructors, execution moves to the declaration of the web service class. Doing this repeatedly may cause the debugger to stop responding.

Delay When Attempting to Examine Values While Debugging

You might experience delay when you try to examine variable values while debugging code with large objects. In these cases, WebLogic Workshop will display ... or no value until the value appears.

Cannot Step into Service Control Instance

When debugging, you can't step into a Service control instance. This version of WebLogic Workshop does not allow you to "step into" on a call to a Service control method.

Workaround: If the called web service's source code (JWS file) is available, you can set a breakpoint there.

Test View Does Nothing When Testing with a Large XML Document

When you have a parameter map for a method, the instance of the XML document you enter in the text area for a parameter can exceed the maximum allowable HTTP GET string. When this occurs the browser will appear to accept the button press but will do nothing.

Workaround: Submit a smaller test document or use the Test XML page instead of the Test Form page.

Stopping WebLogic Server While Debugging Can Peak CPU Use

If you are debugging with breakpoints set, then stop WebLogic Server before ending your debugging session, you may find that your CPU usage peaks.

Workaround: End your debugging session by clicking the Stop button or closing the Test View browser window. To avoid this condition, be sure to end your debugging session before stopping WebLogic Server.

Out of Memory Error While Developing on HP-UX

When running WebLogic Workshop for extended periods of time on HP-UX, you may receive an out-of-memory error from WebLogic Server.

Workaround: Increase the maximum Java heap memory allowed. To do this, open startWebLogic.sh and find the section beginning "HP-UX)". In that section edit the JAVA_OPTIONS variable to increase the maximum Java heap memory to 256 megabytes (the default on installation is 128 megabytes). The following illustrates the line in the script you must edit to increase memory. The edited value is in bold text.

JAVA_OPTIONS="-Xms64m -Xmx**256**m $JAVA_OPTIONS"

WebLogic Workshop on RedHat Linux

Required Environment Setting on Linux 7.1 and Earlier

In order to use WebLogic Workshop on on versions of RedHat Linux prior to 7.2, you must include the following among your environment settings:

```
export LD_ASSUME_KERNEL=2.2.5
```

Asynchrony and Callbacks

Client Callbacks Can Fail Silently

When a callback is invoked on a client that has not provided a callback URL, a RuntimeException is generated but no error is reported automatically in Test View or in workshop.log. You are not required to handle RuntimeExceptions, so the error is completely silent by default. If you want to catch occurrences of this situation, you must implement a handler for the JwsContext.onException callback in your JWS file. For more information on the callback, see How Do I: Handle Errors In a Web Service? in the WebLogic Workshop documentation.

Client Proxy Code

Proxy Support JAR Must Be Manually Added to the Classpath In Order to Call Proxy Classes from WebLogic Workshop

If you are writing code in WebLogic Workshop that uses the Java proxy class to call a web service, WebLogic Workshop does not automatically put the proxy support JAR on the classpath.

Workaround: Add the proxy support JAR to your classpath manually in order to use the Java proxy from a class you are developing in WebLogic Workshop.

XML Maps Must Be Namespace-Qualified In Order to Produce Valid Client Proxy JAR Files

If your web service has XML maps that do not have namespace declarations, the client proxy JAR file generated for that service will be empty. This is due to constraints of the underlying proxy generation code.

Workaround: Add a namespace declaration to the root element of the XML maps, as with the red text in the following example:

```
/**
* @jws:operation
* @jws:parameter-xml xml-map::
* <person xmlns="http://www.openuri.org/">
* <lastname><xm:value obj="String lastName"/></lastname>
* <firstname>{firstName}</firstname>
* </person>
* ::
*
* @return the string "Received person: '{name}'"
*/
public String acceptPerson(String lastName, String firstName)
```

Deployment

WebLogic Workshop Uses a Single JMS Server for Certain Messaging Needs

WebLogic Workshop applications can easily be deployed on WebLogic Server cluster environments using the instructions provided in the WebLogic Workshop documentation in Clustering Workshop Web Services.

In WebLogic Workshop, however, applications that use a JMS message queue rely on the services of a single JMS server and connection factory. This includes applications that use JMS as a transport protocol, the message-buffer property, or Timer controls. Future versions of WebLogic Workshop will use the distributed JMS destination capability in WebLogic Server to reduce the dependency on a single JMS server in a clustered environment.

"Unexpected Exception" When Deploying Secure Web Service

When developing a secure web service, you may see the following error message:

```
An unexpected exception occurred while attempting to deploy the
Enterprise JavaBean for this Web Service.
```

This can happen if there are syntax errors in the web.xml or weblogic.xml files edited to include security information. To confirm that this is the cause of the error message, look in the workshop.log file, which contains exception information logged by a web service. There, you may find specific information about configuration errors.

The workshop.log file is located in the root of the domain in which your web service is deployed. For example, for samples installed with WebLogic Workshop, this is the samples\workshop folder.

Callback Interface Must Be Public on Production Server

Callback interfaces must be declared as "public" in production mode. If you don't declare callback interfaces as public, dynamic proxy generation fails with "IllegalAccessException" in production mode.

UDDI Explorer

UDDI Overview Page Not Rendered in Mozilla on Linux

On Linux, when using the UDDI Explorer to browse for web services, the Mozilla web browser may not render the content of the service's overview page. Note that the page's source code is there, but does not render in Mozilla.

Documentation

Netscape 4.76 May Not Launch from WebLogic Workshop

If your browser is Netscape 4.76, launching Help from WebLogic Workshop may not work. This is a bug in Netscape 4.76 related to launching the browser from another application.

Workaround: Upgrade to at least Netscape 4.78.

Documentation Unusable in Mozilla Version 1.0 on Linux

In the Mozilla 1.0 browser, script used to present navigation panes does not work correctly.

Workaround: Use a different browser.

Resolved Issues

CR085822	JMS-enqueued messages now participate in transactions.
No CR	A Database control no longer throws an exception if it is passed a null complex object.
CR081858	Command length restrictions limiting the number of EJB jars in

	the WEB-INF/lib during a the build of a webservice have been removed.
No CR	Viewing a functions map in auto-generated code no longer updates the file if no changes are made.
CR087532	Bounded repeating elements from a WSDL are now correctly generated and handled as arrays in the Java code.
CR089329	The time *12:00:00* is now correctly treated as noon.
CR084651	**Date** types are now treated differently than **DateTime** types.
No CR	Logging is now supported for SOAP requests.
CR091000	The correct JNDI entries are now displayed in the jndi-name browser for the EJB control in the case where there is an EJB with a descriptor containing the entry <*local-jndi-name*>.
CR084943 **CR086593**	Numerical type mapping has been updated:

xsd:decimal	java.math.BigDecimal
xsd:integer	java.math.BigInteger
xsd:nonPositiveInteger	java.math.BigInteger
xsd:long	long
xsd:nonNegativeInteger	java.math.BigInteger
xsd:negativeInteger	java.math.BigDecimal
xsd:int	int
xsd:unsignedLong	java.math.BigInteger
xsd:positiveInteger	java.math.BigInteger
xsd:short	short
xsd:unsignedInt	long
xsd:byte	byte
xsd:unsignedShort	int
xsd:unsignedByte	short

CR086065	The <*SOAP-ENV:Fault*> tag now allows <*detail*> sub-elements, per the SOAP spec.
CR086582	Inherited members are now correctly referenced in generated XML maps.
CR086911	The DTD reference is now correct in new domains created by the domain wizard.

CR086913	The *jws.ProductionMode* property no longer generates an error message.
No CR	Corrected control generation to include restricted simple types. This adds additional support for enumerations and control file generation. Workshop now correctly identifies decimal data types in schema to generate appropriate Java types.
CR089165	WebLogic Workshop now correctly prints an XML stream passed to a web service.
CR087191	Fixed *types* namespace to qualified all arrays within soap RPC.
CR087789	WebLogic Workshop now processes large ECMA script files in such a way as to avoid errors with methods over 64k.
No CR	If a WSDL contains multiple service definitions, WebLogic Workshop will now generate a Service control for the first one.
CR089740	WebLogic Workshop now supports the importing of a WSDL file from within another WSDL file.
No CR	WebLogic Workshop now correctly handles changing case only on a case insensitive file system when renaming and rebuilding a .jws file.
CR091464	Added *jwErrorDetail* element to *detail* element in the soap fault. The new element is namespace-qualified, with "http://www.bea.com/2002/04/jwErrorDetail/" as the namespace.
CR091612	WebLogic Workshop now displays the correct failure when a WSDL file contains <xsd:anyAttribute>.
No CR	WebLogic Workshop no longer drops top-level namespaces.
CR092775	WebLogic Workshop no longer displays incorrect error when generating *http-xml* methods returning **void**.
CR094366	Removed incorrect space in soap-fault text to conform to SOAP spec.
CR089875	Eliminated the need to restart the server when redeploying a WLW .ear file.