



Hari Venkateshaiah

WebSphere Application Server V6: Application Server Crash Problem Determination

This paper discusses application server crashes that are related to the Java™ Virtual Machine (JVM™) or Just In Time (JIT) compilers in WebSphere Application Server V6. A problem in these components is normally fatal to the application server and can cause it to crash.

High-level symptoms of issues with these components that are most often reported by users are:

- ▶ The application appears to be hung or not available, and the application is not responding to incoming requests.
- ▶ The performance of the application is degrading.

If the application server process is gone from the system or if the process ID is constantly changing (the application server is being restarted), you are probably experiencing an application server crash.

Important: We recommend that you start your problem determination process by reading *Approach to Problem Determination in WebSphere Application Server V6* at <http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf>.

Introduction

The JVM is an interpretive computing engine that is responsible for running the bytecode in a compiled Java program. The JVM translates the Java bytecodes into the native instructions of the host machine. The application server, being a Java process, requires a JVM in order to run and to support the Java applications that are running on it. JVM settings are part of an application server configuration.

The JVM provides the following:

- ▶ Class loader

As the name indicates, a class loader loads and verifies the classes. Multiple class loaders are involved in loading the required libraries for an application to run. Each class must be loaded by a classloader.

- ▶ Garbage collection

Garbage collection takes care of memory management for the entire application server. It searches memory to reclaim space from program segments or inactive data.

- ▶ Execution management

Manages the bookkeeping work for all the Java threads.

- ▶ Execution engine

Interprets the Java methods.

All JVMs use the just in time (JIT) compiler to compile heavily used Java bytecode into native instructions during server runtime to enhance performance.

JVM and JIT are platform specific. These components make use of the functionality that is provided by the operating system for enhancing WebSphere® Application Server performance.

Work the problem

You begin the problem determination process by collecting the appropriate data that is required to diagnose the problem. We first take you through a high-level analysis of your symptoms to determine if you are truly experiencing an application server crash. If so, we provide a list of all the documentation that might be required and how to collect it. You then go through the process of analyzing the data to determine the most likely source of the problem.

And lastly, we provide guidance on the next step to take for resolution, whether it be a support site, contacting IBM®, information about configuration, or some other suggestion as to how to proceed.

Solaris™ and HP-UX users:

- ▶ If the application server is running on Sun™ Solaris, go directly to “Sun Solaris” on page 16.
- ▶ If the application server is running on HP-UX, go directly to “HP-UX” on page 16.

High-level symptom analysis

Symptoms caused by an application server crash fall into the following basic categories:

- ▶ Application server stops responding

An application server that does not respond might be hung or the process might have ended. Users see hung applications or are not able to access new applications.

You should check to see if the application server process is running to determine if you are experiencing a crash. To do this, you need to know the process ID of the application server. You can find the process ID in the server name.pid file in:

```
<WAS_install_root>/profiles/<profile>/logs/<server>
```

Open the <server_name>.pid file in a text editor. The four-digit number is the process ID. You can then use the appropriate operating system command to check to see if the process is running. If it is not running, the problem is a crash.

If the application server is still running, the problem you have is most likely a hang situation. For advice on approaching hang problems, refer to *Approach to Problem Determination in WebSphere Application Server V6* at:

<http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf>

- ▶ Performance degradation

An application server can suffer performance degradation when an application server is repeatedly crashing and being restarted automatically. A quick way to tell if this is the case is to monitor the application server process ID. If it changes over time, the application server is probably crashing and being restarted.

If either is the case, proceed to “Data to collect” on page 4.

Other problems can be caused by the JVM or JIT but are not covered in this paper. Review the following to make sure that your problem does not fit into the following categories:

- ▶ If CPU activity is low but the application server has not terminated, you most likely have a hang or deadlock situation. The following are possibilities:
 - Deadlock caused by JIT generated code (for not releasing the monitor).
 - Deadlock caused by system locks (JIT and JVM internal Locks).
- ▶ If CPU activity is high and the application server is using the cycles, you most likely have a loop. The following are possibilities:
 - Looping in JIT compiled code.
 - Looping in JIT compiler code (while compiling a method).

If either of these is the case, then you need to go back to *Approach to Problem Determination in WebSphere Application Server V6* at <http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf> for general guidance on these types of problems. However, note that some of the techniques in this paper might be helpful in narrowing down the failing code and in gathering documentation.

Data to collect

Diagnosing JVM and JIT problems can involve analyzing information from the following basic sources:

- ▶ Javacore files, also known as javadump files or thread dump files
- ▶ Process dumps, also known as crash dump, core file, or user dumps
- ▶ Process (native) stdout log

If the problem is difficult to recreate or disruptive to business operations, see “The next step” on page 15 for a complete list of documentation to collect before continuing.

Javacore files

A javacore is a text file that is created by an application server during a failure. Javacore is specific to the IBM JDK™. Javacore files contain diagnostic information that is related to the JVM and a Java application captured at a point during execution. For example, the information can be about the operating system, the application environment, threads, native stack, locks, and memory. The exact contents are dependent on the platform where the application server is running. By default, a javacore occurs when the JVM terminates unexpectedly. A javacore can also be triggered by sending specific signals to the JVM.

The JVM checks each of the following locations for existence and write-permission and stores the javacore in the first one available. Note that you must have enough free disk space (possibly up to 2.5 MB) for the javacore file to be written correctly.

- ▶ The location specified by the IBM_JAVACOREDIR environment variable if set.
- ▶ `<WAS_install_root>/profiles/<profile>`.
- ▶ The location that is specified by the TMPDIR environment variable, if it is set.
- ▶ The /tmp directory or on Windows® the location that is specified by the TEMP environment variable, if it is set.
- ▶ Windows only: If the javacore cannot be stored in any of the above, it is put to STDERR.

Working with environment variables

Environment variables for the server process are defined by the administrator as name/value pairs. You can manage environment variables using the following navigation path in the administrative console: **Servers** → **Application Servers** → **<server>** → **Java and Process Management** → **Environment Entries**.

See the Technote at:

<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21162255>

Environment variable settings can also be seen in the Environment Variables section of the javacore.

Process dumps

Process dumps are a complete dump of your computer virtual memory and can, therefore, be quite large. For example, if you have 4 GB of memory on your server, the dump size will also be in the GB range.

Note that you will not be analyzing the process dump but will simply note its existence. In the event that you have to call IBM support, the process dump is required as part of the documentation.

If a process dump exists, you can find it at the following location:

► On Windows

Windows has an embedded function for collecting data from processes that crash. The dumps are put into a file called `user.dmp` and are called *user dumps* for this reason.

To ensure these dumps are enabled and to find the location where they are stored:

- a. Go to **Start** → **Run**, and type `drwtsn32`.
- b. Look for the Crash Dump field to find the location of the dump file and make sure that the Create Crash Dump File option is selected.

If it is a Windows XP machine, then set the Crash Dump Type to NT4 Full Compatible in the dialog box.
- c. Click **OK**. Enabling these settings is a one time process.

► On AIX/Linux®

Process dumps are put into a file that is called `core` and are called *core file* for this reason. These core files are located in the `<WAS_install_root>/bin` directory or the `<operating_system_root>/tmp` directory.

Process (native) stdout log

Native code running in a WebSphere Application Server process can write data to the process logs (also called native logs). Native code is non-Java code typically found in files with `.dll`, `.exe`, and `.so` extensions. The process logs are named `native_stdout.log` and `native_stderr.log`. They are located in the `<WAS_install_root>/profiles/<profile>/logs/<server>` directory.

Analyze the data

The following sections describe what information you should look for. The documentation is listed in the order of importance, so do one at a time.

Is there a javacore file?

If a javacore file exists, check the application server to see if it is still running by looking at the list of process running in the task list. If the server is running, then the application has requested the thread dump (a call to `printStackTrace`), and it is not a true failure.

If the application server stopped after receiving the javacore, go to “Analyze the javacore file” on page 7.

Is there a process dump?

If you did not get a javacore file, check to see if a process dump (core file) exists. If it does, it is still likely that you have had an application server crash.

The first action you should take is to upgrade the JDK version and test again to see if the problem still exists. If it does, go to “The next step” on page 15 for a list of documentation to gather before calling IBM support.

Upgrading the JDK

The IBM JDK has a service refresh two or three times a year. During this time, many JIT and JVM fixes are integrated and new features could be incorporated. Thus, it is advisable to use the latest service refresh before going any further with problem determination.

To determine what level you have, open a command window and issue the **java -fullversion** command.

For information about upgrading the JDK, see:

<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27004980>

JDK updates are only available in refresh packs (for example, 6.0.2), not in fix packs (for example, 6.0.1.3). When a JDK update is available separate from a fix pack, the information is included at this Web site also.

Was a stopServer command issued?

If you do not have a javacore or process dump, you probably have not had an application server crash. Look in native_stdout.log to see if the application server is being stopped with a command. If you see a command to stop the server, investigate why the command is being issued. If you see no indications in the log that the application server was deliberately stopped, go to “The next step” on page 15 for a list of documentation to gather before calling IBM support.

Analyze the javacore file

To begin the analysis of a javacore file, follow these steps:

1. Look into the TITLE tag in the javacore file to find the signal information.

Signals -1,0, OUTFOMEMORY and SIGNONE are the memory signals (see Example 1 on page 8). If you see one of these signals, go to “Out of memory error” on page 14.

Example 1 Javacore signal information

```
NULL
-----
OSECTION      TITLE subcomponent dump routine
NULL          =====
1TISIGINFO   signal 0 received
1TIDATETIME   Date:                2005/05/10 at 08:55:41
1TIFILENAME   Javacore filename:  /apps/WebSphere/wpsIn/AppServer/javacore22490.1115729741.txt
NULL
-----
OSECTION      XHPI subcomponent dump routine
NULL          =====
1XHTIME       Tue May 10 08:55:41 2005
1XHSIGRECV  SIGNONE received at 0x0 in <unknown>. Processing terminated.
1XHFULLVERSION J2RE 1.3.1 IBM AIX build ca131-20040517
```

Signals 10 and 11 indicate an application server crash. In Example 2, a signal 11 (SIGSEGV) occurred and caused the crash.

Example 2 Signal 11

```
OSECTION      TITLE subcomponent dump routine
NULL          =====
1TISIGINFO   signal 11 received
1TIDATETIME   Date:                2005/01/24 at 18:56:08
1TIFILENAME   Javacore filename:  F:\tmp\javacore.20050124.185608.1108.txt
NULL
```

2. Look for the CIJAVAVERSION tag to check the JDK version. Example 3 shows a typical entry.

Example 3 Checking JDK version

```
1CIJAVAVERSION J2RE 1.4.2 IBM Windows 32 build cn142-20041202
```

3. Look for the phrase *Fault module* in the first few lines of the javacore file (Example 4). This section gives the failing module (library) name and location.

Example 4 Javacore showing "Fault Module"

```
SECTION      TITLE subcomponent dump routine
NULL          =====
1TISIGINFO   signal 11 received
1TIDATETIME   Date:                2005/01/24 at 18:56:08
1TIFILENAME   Javacore filename:  F:\tmp\javacore.20050124.185608.1108.txt
NULL
-----
OSECTION      XHPI subcomponent dump routine
```



```
NULL          =====
1XHEXCPCODE   Exception code: C0000005 Access Violation
1XHEXCADDRESS Fault address:  100D15B0 01:000D05B0
1XHEXCPCMODULE Fault module:   D:\IBM_142\jre\bin\classic\jvm.dll
```

If the fault module is:

– The JVM module:

- Windows: JVM.dll
- AIX®: libjvm.a
- Linux: libjvm.so

Upgrade the JDK (see “Upgrading the JDK” on page 7) and retry to see if the problem goes away. If the problem still exists after the upgrade, check for a stack overflow problem (see “Stack overflow” on page 11).

If both of these steps fail to resolve the problem, go to “The next step” on page 15.

– The JIT module:

- Windows: JITC.dll
- AIX: libjitc.a
- Linux: libjitc.so

Upgrade the JDK (see “Upgrading the JDK” on page 7) and retry to see if the problem goes away. If the problem still exists after the upgrade, go to “Finding a workaround for JIT problems” on page 9.

– A WebSphere MQ module, see the following link for fix pack and interim fix information:

<https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=wsmqcsd>

– A DB2® module, see the following link for fixpack information:

<http://www-306.ibm.com/software/data/db2/udb/support/downloadv8.html>

Finding a workaround for JIT problems

You might not be able to fix a JIT-related problem. So, the key is to find a workaround that is acceptable while you report the problem to IBM and get a solution. The best way to find a workaround is to determine the method on which the failure is occurring and have the JIT compiler skip this method. An alternative is to completely disable JIT, though this action can have performance implications.

When you have a workaround, go to “The next step” on page 15”. Meanwhile, you can use the workaround that you have identified.

Skip the failing method

Search the javacore for the phrase Current Thread Details. The method listed at the top is the failing method (Example 5).

Example 5 Find the current thread and failing method

```
1XMCURTHDINFO Current Thread Details
NULL -----
3XMTHREADINFO "Servlet.Engine.Transports : 2" (TID:0x10759F18, sys_thread_t:0x39CCF340,
state:R, native ID:0x8F4) prio=5
4XESTACKTRACE at
com.ibm.workplace.wcm.app.ui.portlet.widget.HTMLPopupMenuButtonRenderer.render(Unknown Source)
4XESTACKTRACE at
com.ibm.workplace.wcm.app.ui.portlet.widget.HTMLPopupMenuButtonRenderer.render(Unknown Source)
4XESTACKTRACE at
com.ibm.psw.wcl.core.ARendererFactory.performRender(ARendererFactory.java(Compiled Code))
```

When you have identified the failing method, set the JITC_COMPILEOPT environment variable to skip the method that is causing the crash in the thread (Example 6). See “Working with environment variables” on page 5.

Example 6 Skip the failing method

```
JITC_COMPILEOPT=SKIP{
com/ibm/workplace/wcm/app/ui/portlet/widget/HTMLPopupMenuButtonRenderer}{render}
```

It might be helpful to turn on the JIT compiling trace to provide additional documentation for IBM support and to verify that you have chosen the correct method to skip. The compiling trace logs every method that is compiled. The last method in this trace before a crash caused by the JIT is the method that should be skipped. The compiling trace is directed to the native_stderr.log for the server.

You can combine turning on the compiling trace (COMPILING) with skipping the method in the same setting, as shown in Example 7.

Example 7 Turn on the compiling trace

```
JITC_COMPILEOPT=COMPILING:SKIP{
com/ibm/workplace/wcm/app/ui/portlet/widget/renderer/HTMLPopupMenuButtonRenderer}{render}
```

Note that JITC_COMPILEOPT options uses a semicolon (;) as a separator on Windows operating system and a colon (:) as a separator on UNIX® systems.

To define an environment entry in the application server process definitions for Example 7 on page 10, use the following values:

- ▶ Name: JITC_COMPILEOPT
- ▶ Value: `COMPILING:SKIP{
com/ibm/workplace/wcm/app/ui/portlet/widget/renderer/
HTMLPopupMenuButtonRenderer}{render}`

Disable JIT

An alternative to skipping the failing method is to simply disable JIT. However, keep in mind that this action can cause performance problems and might not be a viable option.

To disable JIT from the from the administrative console:

1. Select **Servers** → **Application Servers**.
2. Select the server name.
3. Under Server Infrastructure, expand Java and Process Management. Click **Process Definition**.
4. Under Additional Properties, click **Java Virtual Machine**.
5. Select **Disable JIT**.
6. Restart the application server.

Analyzing problem areas

Your analysis of the javacore has most likely led you to a workaround or to one of the areas that are discussed in this section. If not, go to “The next step” on page 15.

Stack overflow

The JVM allocates a Java and native stack for each thread that is created by the application. If either of these stacks become exhausted, a stack overflow error occurs. A Java stack overflow can occur for the following reasons:

- ▶ The stack is not large enough to handle the request. Setting the following JVM parameter increases the Java stack size to 1 MB:
`-Xoss1m`
The default is 400 KB.
- ▶ The JIT compiler is causing a recursion in the application code.
- ▶ There is a recursion in the application code

A native stack overflow can occur for the following reasons:

- ▶ The stack is not large enough to handle the request. Setting the following JVM parameter increases the native stack size to 1 MB:

```
-Xss1m
```

The default is 400 KB.

- ▶ A recursion in native code (JDK or JNI code).

Setting JVM parameters: To set a JVM parameter, use the *Generic JVM arguments* field for the application server. You can find this at **Servers** → **Application Servers** → **<server>** → **Java and Process Management** → **Process Definition** → **Java Virtual Machine**.

To identify a crash that is caused by a stack overflow in a javacore file, look for a signal 11 and a current thread, similar to that shown in Example 8.

Example 8 Identifying a stack overflow

```
1XMCURTHDINFO Current Thread Details
NULL -----
3XMTHREADINFO "Servlet.Engine.Transports : 1" (TID:0x30AED438, sys_thread_t:0x7DCB59A0,
state:R, native ID:0x4553) prio=5: pending=java.lang.StackOverflowError
4XESTACKTRACE at
org.apache.xpath.XPathContext.popSubContextList(XPathContext.java:949)
4XESTACKTRACE at
org.apache.xpath.axes.PredicatedNodeTest.executePredicates(PredicatedNodeTest.java(Compiled
Code))
4XESTACKTRACE at
org.apache.xpath.axes.PredicatedNodeTest.acceptNode(PredicatedNodeTest.java(Compiled Code))
4XESTACKTRACE at org.apache.xpath.axes.AxesWalker.nextNode(AxesWalker.java(Compiled
Code))
4XESTACKTRACE at
org.apache.xpath.axes.WalkingIterator.nextNode(WalkingIterator.java(Compiled Code))
4XESTACKTRACE at
org.apache.xpath.axes.NodeSequence.nextNode(NodeSequence.java(Compiled Code))
4XESTACKTRACE at org.apache.xpath.objects.XNodeSet.compare(XNodeSet.java(Compiled
Code))
4XESTACKTRACE at org.apache.xpath.objects.XNodeSet.greaterThan(XNodeSet.java:667)
4XESTACKTRACE at org.apache.xpath.operations.Gt.operate(Gt.java:45)
4XESTACKTRACE at org.apache.xpath.operations.Operation.execute(Operation.java(Compiled
Code))
4XESTACKTRACE at org.apache.xpath.Expression.bool(Expression.java(Compiled Code))
4XESTACKTRACE at org.apache.xpath.operations.And.bool(And.java:70)
4XESTACKTRACE at org.apache.xpath.operations.And.bool(And.java:70)
4XESTACKTRACE at org.apache.xpath.operations.And.bool(And.java:70)
4XESTACKTRACE at org.apache.xpath.XPath.bool(XPath.java(Compiled Code))
4XESTACKTRACE at
```

```
org.apache.xalan.templates.ElemChoose.execute(ElemChoose.java(Compiled Code))
4XESTACKTRACE          at
org.apache.xalan.transformer.TransformerImpl.executeChildTemplates(TransformerImpl.java(Compiled Code))
4XESTACKTRACE          at
org.apache.xalan.transformer.TransformerImpl.transformToRTF(TransformerImpl.java(Compiled Code))
4XESTACKTRACE          at
org.apache.xalan.transformer.TransformerImpl.transformToRTF(TransformerImpl.java(Compiled Code))
4XESTACKTRACE          at
org.apache.xalan.templates.ElemVariable.getValue(ElemVariable.java(Compiled Code))
4XESTACKTRACE          at
org.apache.xalan.templates.ElemVariable.execute(ElemVariable.java(Compiled Code))
4XESTACKTRACE          at
org.apache.xalan.transformer.TransformerImpl.executeChildTemplates(TransformerImpl.java(Compiled Code))
4XESTACKTRACE          at
org.apache.xalan.templates.ElemTemplate.execute(ElemTemplate.java:394)
4XESTACKTRACE          at
org.apache.xalan.templates.ElemCallTemplate.execute(ElemCallTemplate.java(Compiled Code))
4XESTACKTRACE          at
org.apache.xalan.transformer.TransformerImpl.executeChildTemplates(TransformerImpl.java(Compiled Code))
4XESTACKTRACE          at org.apache.xalan.templates.ElemIf.execute(ElemIf.java(Compiled Code))
4XESTACKTRACE          at
org.apache.xalan.transformer.TransformerImpl.executeChildTemplates(TransformerImpl.java(Compiled Code))
4XESTACKTRACE          at
org.apache.xalan.templates.ElemTemplate.execute(ElemTemplate.java:394)
4XESTACKTRACE          at
org.apache.xalan.templates.ElemCallTemplate.execute(ElemCallTemplate.java(Compiled Code))
4XESTACKTRACE          at
org.apache.xalan.transformer.TransformerImpl.executeChildTemplates(TransformerImpl.java(Compiled Code))
4XESTACKTRACE          at org.apache.xalan.templates.ElemIf.execute(ElemIf.java(Compiled Code))
4XEMORENOTSHOWN      ... (more frames not shown)
```

You can tell that this crash was caused by a stack overflow because of the following clues in the trace:

- ▶ The following entry:

```
pending=java.lang.StackOverflowError
```

- ▶ The stack appears to loop in the following code:

```
org.apache.xalan.templates.ElemCallTemplate.execute(ElemCallTemplate.java(Compiled Code))
```

```
org.apache.xalan.transformer.TransformerImpl.executeChildTemplates(TransformerImpl.java(Compiled Code))
```

If the current thread contains `pending=java.lang.StackOverflowError` but the Java stack does not appear to be in a loop, the stack overflow is probably thrown from the native stack.

To resolve a crash that is caused by a stack overflow, do the following:

1. In the example above, try increasing the Java stack (`-Xoss1m`). For stack overflows that are caused by native stack exhaustion, use `-Xss1m`. If the server still crashes with an identical current thread, then the stack overflow is caused by a recursion in either the JIT or the application code.
2. Upgrade the JDK. If the stack overflow is caused by JIT, upgrading the JDK might resolve the issue.
3. Skip the methods that appear to be looping (see “Skip the failing method” on page 10). In the example above, we would try skipping the following:

```
org.apache.xalan.templates.ElemCallTemplate.execute  
org.apache.xalan.transformer.TransformerImpl.executeChildTemplates
```

4. If these actions do not resolve the problem, then the problem is most likely a recursion in the application code. The owners of the code should be asked to review their code for the cause of the loop.

Out of memory error

During compilation of a method, JIT uses native memory as its work buffer. It also uses native memory to store the compiled code address. This memory is in use as long as those methods are in use by the application. When the classes from the application are garbage collected, this memory is released to the operating system. If the available memory is too small and methods are getting compiled in large numbers, there is a chance that the application will receive an `OutOfMemory` fatal error. In this case, you might need to increase the physical memory of the system.

For information about how to address memory problems, see *Approach to Problem Determination in WebSphere Application Server V6* at:

<http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf>

The next step

The symptoms and problem areas included in this paper are some that you are more likely to experience. However, there are other things that can go wrong, or the cause of the problem might be related to a component other than the JVM or JIT.

If, after going through this process, you still have an undiagnosed problem, it is recommended that you go back to *Approach to Problem Determination in WebSphere Application Server V6* at:

<http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf>

Review the problem classifications to see if there are any other components that might be causing the problem.

If you feel sure you have a JVM or JIT related problem, there are things you can do before contacting IBM support. First, review the documentation that you have gathered for errors related to the problem that were not addressed in this paper, and search support sites for information or fixes.

The following diagnostic guides could be of use:

- ▶ *IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.2 Diagnostics Guide, SC34-6358:*

<http://www-106.ibm.com/developerworks/java/jdk/diagnosis>

- ▶ *Introduction to IBM JVM for Linux JIT diagnostics*

<http://www-128.ibm.com/developerworks/eserver/library/es-JITDiag.html>

Next, collect all of the data that is outlined in the MustGather documentation for JVM and JIT problems and raise a problem record with IBM. Be sure to spell out all of the diagnostic work that you have done so far to minimize the time it takes IBM Support to assist you in resolving your problem.

The following URL lists the MustGather documents for JVM and JIT related problems:

<http://www-1.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCYP8L&q=MustGatherDocument>

Sun Solaris

IBM does not supply a software developer kit or runtime environment for the Sun Solaris platform. However, IBM does make strategic products, such as the WebSphere Application Server, for this platform. WebSphere Application Server contains an embedded copy of the Sun Solaris JVM along with some IBM add-ons, such as security, XML, and ORB packages. The WebSphere Application Server Solaris SDK is, therefore, a hybrid of Sun and IBM products. However, the core JVM and JIT are Sun Solaris. Thus, this book is not appropriate for diagnosis on Sun Solaris.

IBM does service the Sun Solaris SDK, but only when it is an embedded part of IBM middleware, for example, WebSphere Application Server. If you get a Java problem on Solaris as a result of using an IBM middleware product, then follow these steps:

- ▶ Check the following URL for the latest operating system patches. (You should be always on the latest patches.)

<http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/J2SE>

- ▶ MustGather information for crashes on Solaris

<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21049530>

HP-UX

IBM does not supply a software developer kit or runtime environment for HP platforms. However, IBM does make strategic products, such as the WebSphere Application Server, for this platform.

In this case, WebSphere Application Server contains an embedded copy of the HP JVM alongside some IBM add-ons, such as security packages. The WebSphere Application Server HP SDK is, therefore, a hybrid of HP and IBM products. However, the core JVM and JIT are HP software. Thus, this book is not appropriate for diagnosis on HP platforms.

IBM does service the HP SDK, but only when it is an embedded part of IBM middleware (for example, WebSphere Application Server). If you get a Java problem on an HP platform as a result of using an IBM middleware product, check the following URL for latest OS patches:

<http://www.hp.com/products1/unix/java/patches/index.html>

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

This document created or updated on September 30, 2005.



Send us your comments in one of the following ways:

- ▶ Use the online **Contact us** review redbook form found at:
ibm.com/redbooks
- ▶ Send your comments in an email to:
redbook@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662, P.O. Box 12195
Research Triangle Park, NC 27709-2195 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®


AIX®

Redbooks™

@server®

DB2®

WebSphere®

Redbooks (logo) ™

IBM®

The following terms are trademarks of other companies:

Java, JDK, JVM, Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.