Oracle® JRockit

JDK Release Notes

Release R28

E15066-12

July 2011

This document contains important release information about the Oracle JRockit R28 JDK.

Oracle JRockit JDK Release Notes, Release R28

E15066-12

Primary Author: Edwin Spear, Savija T.V.

Contents

Preface

This document contains important release information about Oracle JRockit JDK R28.0.

About this Document

This document includes the following chapters:

- Chapter 1, "Changes in Supported Configurations in Oracle JRockit JDK R28", which lists the changes in the supported configurations for JRockit JDK R28.0 when compared with R27.6.6.
- Chapter 2, "New Features and Changes in Oracle JRockit JDK R28", which the new features and changes in JRockit JDK R28.0.
- Chapter 3, "Issues Resolved in Oracle JRockit JDK R28", which lists issues resolved in JRockit JDK R28.0.
- Chapter 4, "Known Issues in Oracle JRockit JDK R28", which lists issues known to exist in JRockit JDK R28.0.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Changes in Supported Configurations in Oracle JRockit JDK R28

This chapter lists the changes in the supported configurations for JRockit JDK R28.x when compared with R27.6.6.

The following are the changes in supported configurations:

- Java Version Updates
- Hardware Must Support Streaming SIMD Extensions (SSE) 2
- J2SE 1.4.2 and JVMPI Not Supported
- Itanium Platforms Not Supported

Note: JRockit JVM R28.x is not supported on Windows 2000, as was the case with R27.

For up to date supported configuration information, see *Oracle Fusion Middleware Supported System Configurations* at:
`http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html`.

1.1 Java Version Updates

The following table lists the Java versions supported by the various Oracle JRockit JDK R28.0 releases:

Table 1–1 Java Versions Supported by the Oracle JRockit JDK R28.x

JRockit JDK R28.x Release	Supported J2SE 5.0 Update	Supported Java SE 6 Update
R28.1.4	Update 30	Update 26
R28.1.3	Update 28	Update 24
R28.1.1	Update 26	Update 22
R28.1.0	Update 24	Update 20
R28.0.2	Update 24	Update 20
R28.0.1	Update 24	Update 20
R28.0.0	Update 22	Update 17

1.2 Hardware Must Support Streaming SIMD Extensions (SSE) 2

Oracle JRockit JDK R28.x does not support x87, the floating point extension to the x86 platform.

Hardware on which you intend to run the Oracle JRockit JVM must support SSE2 (Streaming SIMD Extensions): that is, Intel Pentium 4 or Pentium M; AMD Opteron or Athlon 64; or newer hardware.

1.3 J2SE 1.4.2 and JVMPI Not Supported

Oracle JRockit JDK R28.x does not support J2SE 1.4.2.

Previous JRockit JDK releases are available and supported for J2SE 1.4.2 users until all the dependent Oracle products reach end-of-life (EOL).

As a consequence of this change, JRockit JDK R28.x does not support JVMPI. Most tools partners use JVMTI, which continues to be supported.

1.4 Itanium Platforms Not Supported

Oracle JRockit JDK R28.x does not support the Itanium architecture.

Previous JRockit JDK releases are available and supported for Itanium platforms until all the dependent Oracle products reach end-of-life (EOL).

2 New Features and Changes in Oracle JRockit JDK R28

This chapter contains the following sections:

- Section 2.1, "Changes in R28.1.0"
- Section 2.2, "Changes in R28.0.1"
- Section 2.3, "New Features and Changes in R28.0.0"

2.1 Changes in R28.1.0

This section lists the changes in JRockit JDK R28.1.0. These changes are:

- Improved Garbage Collection
- Command-Line Option to Specify the Receive Buffer Size
- Enabling JVM Crash When an Out-of-Memory Error Occurs
- Collecting and Packaging Flight Recording Data from Disk Buffers

2.1.1 Improved Garbage Collection

In the `genpar` garbage collection mode, when the nursery runs out of memory in the old generation, objects that are identified for promotion to the old space are promoted within the nursery and this resulted in fragmentation of the nursery. This situation is known as promotion failure.

In R28.1, the JRockit JVM prevents promotion failure by triggering an early old collection for those young collections that are running out of memory.

2.1.2 Command-Line Option to Specify the Receive Buffer Size

When reading from network sockets, the size of the receive buffer can be limited by using the new command-line option, `-XX:MaxRecvBufferSize`.

For more information about this option, see `-XX:MaxRecvBufferSize` in the *Oracle JRockit Command-Line Reference*.

2.1.3 Enabling JVM Crash When an Out-of-Memory Error Occurs

Oracle JRockit R28.1 introduces the command-line option `-XX:[+|-]CrashOnOutOfMemoryError`. If this option is enabled, when an out-of-memory error occurs, the JRockit JVM crashes and produces crash files. The state of the JVM before a crash is saved to a core dump file for off-line analysis.

For more information about this option, see `-XX:+|-CrashOnOutOfMemoryError` in the *Oracle JRockit Command-Line Reference*.

2.1.4 Collecting and Packaging Flight Recording Data from Disk Buffers

This release of Oracle JRockit introduces the command-line tool `oracle.jrockit.jfr.tools.ConCatRepository`, that allows you to extract JRockit Flight Recorder data that has been written to disk, but not handled and packaged as a flight recording, and then create a flight recording from it. This feature is useful when you have flight recording buffers on disk and the JVM terminates in such a way that `.jfr` files are not assembled to a complete flight recording file.

For more information, see the *Oracle JRockit Flight Recorder Run Time Guide*.

2.2 Changes in R28.0.1

This section lists the changes in JRockit JDK R28.0.1.

2.2.1 Default MaxCodeMemory on Linux IA32 with Large Pages Increased to 64 MB

The default maximum code memory on Linux IA32 with large pages was 32 MB in R28.0.0.

In R28.0.1, the default value has been changed to 64 MB.

For more information, see `-XX:MaxCodeMemory` in the *Oracle JRockit Command-Line Reference*.

2.3 New Features and Changes in R28.0.0

The following are the new features and changes in JRockit JDK R28.0.0. These changes are:

- Change in Thread Suspension Mechanism
- Ability to Generate HPROF-Formatted Heap Dumps
- Improved Logging for Code Generation and Optimization
- Better Control Over Code Optimization Through Directives
- Garbage Collection Strategy Does Not Change at Run Time
- Large Objects Are Allocated in the Nursery
- Single Command-Line Option to Specify Compaction Behavior
- Changes in the JMX Agent
- Compressed References for Larger Heaps
- Changes in Heap Sizing
- Change in Class and Code Garbage Collection
- New Command-Line Options in R28.0
- Command-Line Options Deprecated in R28.0
- Command-Line Options Changed to the HotSpot Format in R28.0

2.3.1 Change in Thread Suspension Mechanism

The mechanism to stop threads in the JVM has been changed. In previous releases of the JRockit JVM, threads were suspended (for performing garbage collection, for example) by sending signals. In JRockit JVM R28.0, the threads check periodically whether they should self-suspend.

This change does not result in visible behavioral changes, but it makes the JRockit JVM easier to maintain and less error-prone.

2.3.2 Ability to Generate HPROF-Formatted Heap Dumps

Heap dumps can now be generated in the HPROF binary format, which can be parsed using heap analysis tools. You can use the `-XX:+HeapDumpOnOutOfMemoryError` or `-XX:+HeapDumpOnCtrlBreak` command-line options to generate Java heap dumps in HPROF binary format on `OutOfMemory` errors. You can also generate heap dumps in HPROF format by using the `hprof` diagnostic command and through JRockit Mission Control.

For more information, see "Generating Java Heap Dumps in the HPROF Binary Format" in the *Oracle JRockit Diagnostics and Troubleshooting Guide*.

2.3.3 Improved Logging for Code Generation and Optimization

The granularity of logging for code generation and optimization has been increased, to enable faster diagnostics and troubleshooting. The information in the outputs of the `-Xverbose:opt` and `-Xverbose:codegen` options is now more detailed. For more information, see the *Oracle JRockit Command-Line Reference*.

2.3.4 Better Control Over Code Optimization Through Directives

In previous releases of the JRockit JVM, you could control code optimization by specifying directives in an optfile and then using the `-Djrockit.optfile` property to indicate the name and location of the optfile.

In JRockit JVM R28.0, the format for specifying compiler-control directives in the optfile has been extended and improved to enable control over code optimization at a more detailed level. A new **diagnostic** command-line option, `-XX:OptFile`, is available for specifying the name and path of the optfile.

For more information, see "Specifying Optimization Directives" in the *Oracle JRockit Diagnostics and Troubleshooting Guide*.

2.3.5 Garbage Collection Strategy Does Not Change at Run Time

In JRockit JVM R28.0, when you specify the `throughput` or `pausetime` garbage collection mode by using the `-Xgc` command-line option, the strategy associated with the specified mode— by default, `genpar` and `gencon` respectively—is used **throughout** the run time. The garbage collector does not change between generational and nongenerational garbage collectors during the run time.

This change has been made to reduce the extent of underterministic garbage collection behavior due to strategy changes during run time.

Note that the `-XgcPrio` option continues to work in R28.0. Oracle recommends that you use the `-Xgc` option instead of using the `-XgcPrio` option.

For more information about `-Xgc`, see the *Oracle JRockit Command-Line Reference*.

2.3.6 Large Objects Are Allocated in the Nursery

Oracle JRockit JVM R28.0 allocates large objects in the nursery if the size of the object is within the limit specified by the `-XXtlaSize:wasteLimit` command-line option.

This change improves the utilization of the nursery and reduces the frequency of old-space garbage collections.

For more information about `-XXtlaSize:wasteLimit`, see the *Oracle JRockit Command-Line Reference*.

2.3.7 Single Command-Line Option to Specify Compaction Behavior

Oracle JRockit R28.0 supports a new command-line option that enables you to specify compaction-related behavior: `-XXcompaction`. This option accepts all the parameters for compaction: compaction percentage, maximum number of references, and so on.

All the other compaction-related options are deprecated in R28.0.

For more information about `-XXcompaction`, see the *Oracle JRockit Command-Line Reference*.

2.3.8 Changes in the JMX Agent

In JRockit JVM R28.0, after the local management service starts, it remains active until the JVM is terminated.

In previous releases, the performance counter memory used to leak, and new addresses of the JMX connector were written during a memory leakage. Therefore, the JMX client was reading the wrong address of the JMX connector. This issue has been fixed in R28.0.

To allow RMI communication between the JRockit JVM server and a client through a firewall, two ports (RMI Registry and RMI Server) are required to configure the firewall. In previous releases, the RMI Server port number was generated randomly on the JRockit JVM server; so it was not possible to configure the firewall in advance. In JRockit JVM R28.0, the JMX agent enables you to select the same port number for the RMI Registry and the RMI Server. Therefore, you can use the default JMX agent for RMI communication through a firewall.

You can set the JMX agent properties by using the system properties or by using the `-Xmanagement` command-line option. For more information about the JMX agent system properties, see Appendix B "JMX Agent-Related –D Options" in the *Oracle JRockit Command-Line Reference*.

2.3.9 Compressed References for Larger Heaps

Oracle JRockit JVM R28.0 supports up to 64 GB compressed references for various heap sizes. You can define the compressed reference size during the JVM startup by using the `-XXcompressedRefs` command-line option.

For more information about `-XXcompressedRefs`, see the *Oracle JRockit Command-Line Reference*.

2.3.10 Changes in Heap Sizing

In JRockit JVM R28.0, the heap grows faster than before. The JVM also ensures that the heap size grows up to the maximum Java heap size (`-Xmx`) before an OutofMemory error is thrown. In addition, the default value of the `-Xmx` option is changed from 1 GB to 3 GB on 64-bit platforms.

The JRockit JVM shrinks the heap if it is unused or if other applications require more physical memory.

In the previous releases, the JVM used to crash when the heap size reduced from a very large size to a small size. This issue has been fixed in R28.0.

2.3.11 Change in Class and Code Garbage Collection

The pause time during the old collection of code and class garbage collections has been removed in Oracle JRockit JVM R28.0. With this change, the code and class garbage collections are mostly concurrent in R28.0.

2.3.12 New Command-Line Options in R28.0

The Oracle JRockit JVM R28.0 supports several new command-line options.

For more information, see Appendix A, "Changes in Command-Line Options" in the *Oracle JRockit Command-Line Reference*.

2.3.13 Command-Line Options Deprecated in R28.0

Some command-line options have been deprecated in Oracle JRockit JVM R28.0.

For more information, see Appendix A, "Changes in Command-Line Options" in the *Oracle JRockit Command-Line Reference*.

2.3.14 Command-Line Options Changed to the HotSpot Format in R28.0

In Oracle JRockit JVM R28.0, the format of several command-line options has been changed to the HotSpot format: `-XX:+|-`*`option`* (for example, `-XX:+UseClassGC`).

For a list of the command-line options that have been changed to the HotSpot format, see Appendix A, "Changes in Command-Line Options" in the *Oracle JRockit Command-Line Reference*.

3 Issues Resolved in Oracle JRockit JDK R28

This chapter lists the issues resolved in Oracle JRockit JDK R28.0. It contains the following sections:

3.1 Issues Resolved in R28.1.4

The following issues have been fixed in Oracle JRockit JDK R28.1.3:

- Warnings Print When Launching Java Involving Symbolic Links on Windows
- Corrupt HPROF File

3.1.1 Warnings Print When Launching Java Involving Symbolic Links on Windows

When launching Java involving symbolic links on Windows, Oracle JRockit would sometimes print a warning, such as:

```
[WARN ][osal   ] Could not add counter \Virtual Bytes for query
[WARN ][osal   ] Failed to init virtual size counter.
```

This has been fixed.

3.1.2 Corrupt HPROF File

When Oracle JRockit was configured to dump an HPROF on an OutOfMemoryError and was receiving multiple simultaneous OutOfMemoryErrors in multiple threads, the resulting HPROF file might have been corrupt. This has been fixed.

3.2 Issues Resolved in R28.1.3

The following issues have been fixed in Oracle JRockit JDK R28.1.3:

- Deadlock Occurring in the ClassLoader (Sun Bug 7001933)
- "Peer Not Authenticated" Exception Unexpectedly Thrown (Sun Bug 6924489)
- Problem Setting SO_RCVBUF/SO_SNDBUF (Sun Bug 6984182)
- Passing Read-Only Bytebuffer to Channel Write Method Throwing Exception
- **Specific JNI API Routines Did Not Correctly Set isCopy Parameter**
- Incorrectly Optimized Methods Forcing Long Values to Become Very Large

3.2.1 Deadlock Occurring in the ClassLoader (Sun Bug 7001933)

Occasionally, if a custom file protocol handler was in place, a deadlock would occur in the ClassLoader. This has been fixed; this fix resolves Sun Bug 7001933.

3.2.2 "Peer Not Authenticated" Exception Unexpectedly Thrown (Sun Bug 6924489)

The exception `javax.net.ssl.SSLPeerUnverifiedException: peer not authenticated` was unexpectedly being thrown. This has been fixed; this fix resolves Sun Bug 6924489.

3.2.3 Problem Setting SO_RCVBUF/SO_SNDBUF (Sun Bug 6984182)

Setting `SO_RCVBUF/SO_SNDBUF` to a value larger than `tcp_max_buf` failed on Solaris 11 if the kernel parameters changed. This has been fixed; this fix resolves Sun Bug 6984182.

3.2.4 Passing Read-Only Bytebuffer to Channel Write Method Throwing Exception

Passing a read-only bytebuffer to a channel write method could throw a `java.nio.ReadOnlyByteBufferException`. This has been fixed.

3.2.5 Specific JNI API Routines Did Not Correctly Set isCopy Parameter

JNI API `Get<PrimitiveType>ArrayElements` routines did not correctly set `isCopy` parameter to `JNI_TRUE` when returning copies. This has been fixed.

3.2.6 Incorrectly Optimized Methods Forcing Long Values to Become Very Large

In rare circumstances, on instances of 32-bit JRockit, a method could be incorrectly optimized forcing long values that should be 0 to become very large. This has been fixed.

3.3 Issues Resolved in R28.1.1

The following issues have been fixed in Oracle JRockit JDK R28.1.1:

- Crashes During Concurrent Sweep JNI Object Allocation
- Silent Exit When Command-Line Options are Misspelled
- Erroneous Optimization of an arraycopy
- JDK Read Fixed Number of Bytes When Calling SecureRandom.generateSeed
- instanceof Check Failing

3.3.1 Crashes During Concurrent Sweep JNI Object Allocation

Previously, while it was conducting a concurrent sweep without a nursery, Oracle JRockit might crash if it tried to allocate an object from JNI that was the exact size of the minimum thread local area size. This has been fixed.

3.3.2 Silent Exit When Command-Line Options are Misspelled

R28.0.0 and later silently exit if a command-line option was misspelled; for example, "`-X:MaximumNurseryPercentage=80`" (note the single `X` where `XX` is required). This has been fixed in R28.1.1.

3.3.3 Erroneous Optimization of an arraycopy

In rare cases, Oracle JRockit could erroneously optimize an `arraycopy` to reuse the source array as the target array. This could lead to the wrong values being read from the source array after the `arraycopy`. This has been fixed.

3.3.4 JDK Read Fixed Number of Bytes When Calling SecureRandom.generateSeed

When calling `SecureRandom.generateSeed`, the JDK would always read 8192 bytes from the entropy pool. The JDK has been changed to only read the number of bytes requested. For more information, see:

`http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6998583`

3.3.5 instanceof Check Failing

Occasionally, the optimizer would erroneously change the behavior of an `instanceof opcode`, causing the `instanceof` check to fail. This has been fixed.

3.4 Issues Resolved in R28.1.0

The following issues have been fixed in Oracle JRockit JDK R28.1.0:

- Oracle JRockit Hangs when used with Application Management Solutions
- Memory Leakage in the JMX Implementation
- Oracle JRockit Exits when Aborting an Optimization

- Oracle JRockit Heap Dumps Do Not Open in Eclipse Memory Analyzer
- Exceptions Thrown Without InvocationTargetException Wrapping

3.4.1 Oracle JRockit Hangs when used with Application Management Solutions

Oracle JRockit would hang during startup when Oracle WebLogic Server was started with an application management solution such as CA Wily Introscope.

This issue has been fixed in JRockit R28.1.0.

3.4.2 Memory Leakage in the JMX Implementation

A memory leakage used to occur in the JMX implementation of JDK 6. This issue has been fixed by Sun in JDK 6 Update 22.

The fix is included in JRockit R28.1.0.

3.4.3 Oracle JRockit Exits when Aborting an Optimization

If the `-XX:+|-ExitOnOutOfMemoryError` option was enabled, JRockit would exit when aborting an optimization due to the compiler memory limit.

This issue has been fixed in JRockit R28.1.0.

3.4.4 Oracle JRockit Heap Dumps Do Not Open in Eclipse Memory Analyzer

When the heap size was 2 GB or higher, JRockit would write segmented heap dumps that Eclipse Memory Analyzer (MAT) could not parse.

This issue has been fixed in JRockit R28.1.0.

3.4.5 Exceptions Thrown Without InvocationTargetException Wrapping

When you invoke methods using the class reflection feature, JRockit would sometimes throw an exception type that is different from the signature of the `Method.invoke()` method.

This issue has been fixed in JRockit R28.1.0. Invoking methods using reflection now throws the correct `InvocationTargetException` type.

3.5 Issues Resolved in R28.0.2

The following issues have been fixed in Oracle JRockit JDK R28.0.2:

- Oracle JRockit Starts Slowly on Some Solaris Machines
- IO Exceptions in Epoll Socket Muxer Would Throw NoClassDefFoundErrors
- Oracle JRockit Crashing While Pruning References to Obsoleted Code
- Oracle JRockit Could Not Open JAR or ZIP Files Larger Than 2GB
- Xalan and Xerces Versions Updated

3.5.1 Oracle JRockit Starts Slowly on Some Solaris Machines

On some Solaris machines, Oracle JRockit would start slowly, printing warnings; for example:

```
[WARN ][osal   ] Failed to initialize kstat for CPU 0, ignoring
```

This issue has been fixed in release R28.0.2.

3.5.2 IO Exceptions in Epoll Socket Muxer Would Throw NoClassDefFoundErrors

IO exceptions originating from the epoll socket muxer would occasionally throw NoClassDefFoundErrors when trying to find the `java/lang/IOException` class. This issue has been fixed in release R28.0.2.

3.5.3 Oracle JRockit Crashing While Pruning References to Obsoleted Code

Occasionally, while pruning references to obsoleted code, Oracle JRockit would crash in `Code_and_classgc_background_task`. This issue has been fixed in release R28.0.2.

3.5.4 Oracle JRockit Could Not Open JAR or ZIP Files Larger Than 2GB

In previous Oracle JRockit releases, the JVM could not open JAR or ZIP files larger than 2GB. This issue has been fixed in release R28.0.2.

3.5.5 Xalan and Xerces Versions Updated

The Xalan and Xerces versions were updated to fix a functional regression found in JDK 1.6.0_18. This fix is included in the Oracle JRockit R28.0.2 JVM.

3.6 Issues Resolved in R28.0.1

The following issues have been fixed in Oracle JRockit JDK R28.0.1.

- JVM Crashes on Encountering Non-UTF8 Characters in Compiler Directives
- Null-Check Incorrectly Optimized or Proved as Always Failing
- Linux Systems Crash at Startup when libjsig.so is Set to be Preloaded
- NIO Selector Functionality Failure
- Deprecated Flag -XXExternalCompactRatio Gives Incorrect Warning
- ZipEntry Initialization Error
- Crash in ZLIB Code While Running Finalizer
- Undeterministic Behavior on x86_64 Machines
- JVM Spins Forever When Compiling JavaFX Classes
- Descriptions Not Intuitive for Compaction JFR Events
- WLS NIOSocketMuxer Occasionally Loses Sockets On Windows

3.6.1 JVM Crashes on Encountering Non-UTF8 Characters in Compiler Directives

The JVM crashes when it encounters non-UTF8 characters in a compiler control directives file.

This issue has been fixed in R28.0.1.

3.6.2 Null-Check Incorrectly Optimized or Proved as Always Failing

In certain cases involving try-catch clauses, the JVM incorrectly optimizes or proves a null-check as always failing.

This issue has been fixed in R28.0.1.

3.6.3 Linux Systems Crash at Startup when libjsig.so is Set to be Preloaded

On Linux systems, the JVM crashes at startup if the user sets `libjsig.so` (the signal chaining library) to be preloaded through the `LD_PRELOAD=libjsig.so` environment variable. This prevents some third-party JNI libraries from being used with Oracle JRockit R28. To download a patch for this issue, see patch ID 9586671 for JDK 6 and patch ID 9672120 for JDK 5.

This issue has been fixed in R28.0.1.

3.6.4 NIO Selector Functionality Failure

In certain cases, the NIO selector functionality fails unless `net.dll` is loaded before `nio.dll`. This happens only when using `JAVA_HOME\bin\java` as opposed to `JAVA_HOME\jre\bin\java`. To download a patch for this issue, see patch ID 9586671 for JDK 6 and patch ID 9672120 for JDK 5.

This issue has been fixed in R28.0.1.

3.6.5 Deprecated Flag -XXExternalCompactRatio Gives Incorrect Warning

When the deprecated command-line option `-XXexternalCompactRatio` is used, the following incorrect warning is displayed.

```
[WARN ] -XXexternalCompactRatio is a deprecated option. Please use
-XXcompaction:internalPercentage instead.
```

This issue has been fixed in R28.0.1.

3.6.6 ZipEntry Initialization Error

When the `java.util.zip.ZipEntry` created for an uncompressed entry (method STORED) is initialized, the uncompressed and compressed fields are not initialized with the same value. This sometimes causes a `java.util.zip.ZipException`.

This issue has been fixed in R28.0.1.

3.6.7 Crash in ZLIB Code While Running Finalizer

Sometimes, calling `java.lang.util.zip.Deflater.deflateBytes()` after calling `java.lang.util.zip.Deflater.end()` causes the JVM to crash.

This issue has been fixed in R28.0.1 to throw a `NullPointerException`.

3.6.8 Undeterministic Behavior on x86_64 Machines

Sometimes, a method invocation with more than eleven arguments introduces undeterministic behavior in Java applications on x86_64 machines.

This issue has been fixed in R28.0.1.

3.6.9 JVM Spins Forever When Compiling JavaFX Classes

The JRockit JVM spins forever when compiling JavaFX classes that contain endless loops.

This issue has been fixed in R28.0.1.

3.6.10 Descriptions Not Intuitive for Compaction JFR Events

Some descriptions for the `GcCompaction` JFR event are not intuitive.

This issue has been fixed in R28.0.1.

3.6.11 WLS NIOSocketMuxer Occasionally Loses Sockets On Windows

At times, sockets disappear when the `NIOSocketMuxer` is used with WebLogic Server running on Windows.

This issue has been fixed in R28.0.1.

3.7 Issues Resolved in R28.0.0

The following issues have been fixed in Oracle JRockit JDK R28.0.0.

- Deadlocks On the Windows Platform When Threads Block on I/O Operations
- Issues with Nondefault Flag with -XXcallProfiling in Oracle JRockit R27.x
- Performance Issues with Windows Computers Running Many Processes
- Optimizing Compiler Producing Erroneous Results
- Broken Java Launcher Removed from Product
- JVMTI_EVENT_COMPILED_METHOD_UNLOAD Event Not Being Posted

3.7.1 Deadlocks On the Windows Platform When Threads Block on I/O Operations

This release adds a workaround for deadlocks on the Windows platform when thread(s) block on I/O operations on any standard stream (`stdin`, `stdout` `stderr`/`System.in`, `System.out`, `System.err`) while a shared library (DLL) is loading. The deadlock occurs if a process is launched such that the stream on which a call is blocked is a redirected pipe, typically the result of either a spawned process through `Process.exec` or similar; or launched through a shell with data piped to or from itself. The bug happens because any I/O operation on such a pipe holds a kernel lock during the whole call, a lock which is also required by the WinAPI function `GetFileType()`. This function in turn is called from the Microsoft CRT startup code whenever either a new CRT DLL, or a DLL with statically linked CRT functions, is loaded.

The bug only occurs on Windows releases prior to Windows Vista.

Workaround

Detect whenever the JVM process is started with its `stdin` redirected to a pipe. Then intercept all `FileInputStream.read()` calls to it and prevent them from blocking, essentially doing polling I/O. The workaround prevents a thread reading from `System.in` from blocking any `System.loadLibrary` calls. This workaround might add some latency to reading from `System.in`, but should be invisible for most applications.

This workaround can be controlled by using these diagnostic options (unlock using `-XX:+UnlockDiagnosticVMOptions`):

- `-XX:+|-UseStdinPipeReadWorkaround`, which enables the workaround (default is on).
- `-XX:StdinPipeReadWorkaroundPollPeriod=<`*`millis`*`>`, which polls the period for reads from `stdin` (default 1)

Notes:

- This fix does not handle NIO reads from `System.in`. Using these might still cause deadlocks of the JVM.
- This fix is only activated if redirection of standard in is detected.

3.7.2 Issues with Nondefault Flag with -XXcallProfiling in Oracle JRockit R27.x

This release resolves the following issues, which arose when using the nondefault flag, `-XXcallProfiling` in Oracle JRockit R27.x.

- Deadlock between compiler and code garbage collection.
- Memory leak of call profiling data from invalidated methods.

3.7.3 Performance Issues with Windows Computers Running Many Processes

This release fixes the issue of slow performance on Windows machines running with many (more than 40) processes with the same name whenever access was needed to the process PDH header or slow startup when running with the JRockit Flight Recorder.

3.7.4 Optimizing Compiler Producing Erroneous Results

The Oracle JRockit optimizing compiler was, in rare cases, producing erroneous results from computations that included a narrowing of primitive types. This issue has been fixed.

3.7.5 Broken Java Launcher Removed from Product

The broken Java launcher java-rmi.exe in JRockit 6 on Windows is no longer shipped with the product. See also Sun Bug 6512052 at:`http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6512052`

3.7.6 JVMTI_EVENT_COMPILED_METHOD_UNLOAD Event Not Being Posted

Oracle JRockit R27.6.0 was not posting the `JVMTI_EVENT_COMPILED_METHOD_UNLOAD` event when unloading obsolete method code. This issue has been fixed.

4

Known Issues in Oracle JRockit JDK R28

This chapter describes issues known to exist in the Oracle JRockit JDK R28.

4.1 HPROF Heap Dump Might be Corrupt When Multiple OOMs Thrown

If Oracle JRockit encounters several Java Out of Memory exceptions while writing an HPROF heap dump, the contents of the heap dump might be corrupt.

4.2 JVM Crashing During GC When Running With -Xdebug or -agentlib:jdwp

Under some circumstances, Oracle JRockit might crash in the garbage collector when running with `-Xdebug` or `-agentlib:jdwp`. When running in debugging mode, if variables are defined in a try-block and the JVM stops the thread at certain safepoints within the try/catch/finally block, the JVM will crash. The root cause for this problem is that some class files generated by Oracle javac contain incorrect debugging information, information the Oracle JRockit JVM depends on when stopping the threads.

Workaround:

Do not use `-Xdebug` or `-agentlib:jdwp`. The bug in javac will be fixed in an upcoming release of the JDK. If you have additional questions, contact Oracle Support.

4.3 java.math.BigDecimal Objects Cannot be Serialized Over IIOP Between Releases

Serialization of `java.math.BigDecimal` objects over IIOP between the JRockit JVM and other JVMs throws an IOException. This incompatibility has been fixed; the JRockit JVM R28 is now compatible with other JVMs but not with older R27 releases. As a consequence, `java.math.BigDecimal` objects cannot be serialized over IIOP between the R27 and R28 releases of the JRockit JVM.

4.4 Timing Stability Issue When "Fast Time" Is Enabled on Intel Systems

Timing stability issues might occur on modern x86 systems (for instance Nehalem-EX) with more than two CPU sockets.

Disabling fast time (by using the `-XX:-UseFastTime` command-line option) could solve the issue.

4.5 JMAPI Method Changed to Throw an UnapplicableMethodException

In R28.0 the JMAPI method `com.bea.jvm.ProfilingSystem.newConstructorProfileEntry()` was changed to throw an `UnapplicableMethodException`. This exception is never thrown in practice but the addition causes compilation errors for old code. Removing the exception declaration will also cause problems compilation, thus the exception will remain in future versions.

4.6 Error Message for CPU Load Counters for JRockit JVM Running on Windows

At times, the following message might be displayed when running the JRockit JVM on Windows.

```
[WARN ][osal   ] could not enumerate processors (1) error=-1073738824
[WARN ][osal   ] Failed to init system load counters
```

This issue occurs when the Windows processor performance counter (`PerfOS`) is disabled.

The message also indicates that CPU load events are not recorded in JRockit Flight Recorder and not shown in the JRockit Mission Control console.

Workaround: Enable the `PerfOS` counter in Windows by using the Microsoft Extensible Counter List tool (`exctrlst.exe`). You can download the tool from `http://download.microsoft.com/download/win2000platform/exctrlst/1.00.0.1/nt5/en-us/exctrlst_setup.exe`.

4.7 ACopyRemoval Breaks Explicit Typechecks

Inner type checks on arrays might show the wrong type in optimized code.

4.8 Oracle JRockit Hangs On OEL/OVM Combination

When Oracle JRockit is running on OEL on OVM, a fix is required in OEL. Listed below are the minimum requirements for running JRockit on OEL on OVM:

- OVM 2.1.2
- OEL 4.7 ia32

  Patch required. The version of the para-virtualized kernel for OEL must be 2.6.9-78.0.13.0.1.1.ELxenU or later.
- OEL 4.7 x64

  GA bits works fine
- OEL 5.3 ia32 and x64

  GA bits works fine

Oracle JRockit supports both hardware and para-virtualized versions and both OEL 4 and OEL 5.

4.9 Triggering Young Collections if the Nursery is Too Small

If the nursery is too small, Oracle JRockit might begin triggering young collections, "back to back", without promoting anything. This appears in the `-Xverbose:memdbg` output as repeated young collections where the number of promoted objects is zero. It can also be seen as very short times between the young collections (close to 0 ms).

Workaround:

Increase the nursery size. If nursery size has been set automatically by `-Xgcprio:throughput`, it can be overridden by manually setting `-Xns` to a higher value.

4.10 SSE2 Registers Might Not be Restored Correctly After Return from Signal Handler

Due to a Linux kernel bug, certain SSE2 registers might not be restored correctly after returning from a signal handler. This issue manifests itself as such undefined behavior as erroneous floating values in Java code and crashes.

Workaround:

This problem is fixed in mainline kernel version 2.6.xx. You can obtain patches for OEL 4 and OEL 5 as RPM'S from the Unbreakable Linux Network. Follow normal kernel upgrade procedure to obtain the fix.

For older kernels, use the command-line option `-XX:+UseMembarForTransitions`.

4.11 System Crashing when Stack Expansion Uses Randomized Address Spaces

On some newer Linux systems (for example, SLES 11) you might experience crashes related to stack expansion when using randomized address spaces.

Workaround:

In Linux, you might be able to eliminate these crashes by using the kernel configuration command `sysctl -w kernel.randomize_va_space=0`.

In Oracle JRockit, you can eliminate these crashes by using the JVM command-line option `-XX:+TrustPThreadStackInfo`. The flag defaults to false.

4.12 Large Pages on Solaris Might Cause Long Pauses

Using large pages on Solaris might occasionally cause long pauses. These pauses happen when a page is accessed for the first time.

Workaround:

Disable large pages by using the command-line option `-XX:-UseLargePagesForHeap`.

4.13 Calculation-Intensive Applications Returning Corrupt Register Values

Floating point calculation-intensive programs run on top of the Oracle JRockit JVM might result in bogus results. This happens because a bug in the Linux kernel does not preserve some CPU registers when switching between tasks.

Oracle makes a patch available for OEL 4 and 5. For OEL 4 you need OEL 4.8 with updated kernel (2.6.9-89.0.18.0.1.EL or later). For OEL 5 you need an updated kernel (2.6.18-164.9.1.0.1.el5 or later). The fix is included in OEL 5.5.

Novell also makes a fix available (BugZilla number 573478). The fix is available for SLES 9 SP4; SLES 10 SP2 and SP3; and SLES 11.

The issue has also been reported to RedHat (BugZilla number 547893). Contact RedHat support for access to this fix.

4.14 R28 Not Supported On Windows 2008 With More Than 64 Processors

Oracle JRockit does not support more than 64 logical processors on Windows Server 2008 and Windows Server 2008 R2.

4.15 Out of Memory Error Occurs When Classblock Memory Runs Low

On Solaris/SPARC, due to the way classblock memory is reserved, Oracle JRockit might occasionally run out of memory when a large number of classes are loaded (in the order of 100000).

Workaround:

Use the following command-line options:

```
-XX:+UnlockDiagnosticVMOptions  -XX:MaxClassBlockMemory=xxM
```

The default value of `-XX:MaxClassBlockMemory` is 50 MB, and a reasonable value is around 75 MB.