

Oracle® JRockit

JDK Release Notes

Release R27

E29530-01

May 2012

This document contains important release information about the Oracle JRockit R27 JDK.

Oracle JRockit JDK Release Notes, Release R27

E29530-01

Copyright © 2001, 2012, Oracle and/or its affiliates. All rights reserved.

Primary Author: Savija Vijayaraghavan

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface	vii
About this Document.....	vii
Documentation Accessibility	vii
Conventions	vii
1.1 Version Naming	1-1
1.2 Java Support	1-1
1.3 Platform Support	1-2
1.4 Installation	1-3
1.5 Documentation Accompanying the Oracle JRockit JDK	1-3
2.1 Latest Release	2-1
2.2 JDK Update Versions Supported by this Release	2-1
2.3 New Features and Enhancements in the Oracle JRockit JVM R27.7	2-2
2.4 New Features and Enhancements in the Oracle JRockit JVM R27.6	2-2
2.5 New Features and Enhancements in the Oracle JRockit JVM R27.5	2-2
2.5.1 Support for Updated Java Versions	2-2
2.5.2 Eclipse Integration of JRockit Mission Control	2-2
2.5.3 Other JRockit Mission Control Updates	2-2
2.5.4 Updated Command-line Options	2-4
2.5.4.1 -Xverbose	2-4
2.5.4.2 -XpauseTarget	2-4
2.5.4.3 -XlargePages	2-4
2.6 New Features and Enhancements in the Oracle JRockit JVM R27.4	2-4
2.7 New Features and Enhancements in the Oracle JRockit JVM R27.3	2-5
2.7.1 Java Updates	2-5
2.7.2 BEA JRockit Mission Control 3.0	2-5
2.7.3 GUI Localizaton	2-5
2.7.4 Localized Documentation	2-5
2.7.5 Performance Improvements	2-5
2.7.5.1 Better Out of the Box Performance	2-5
2.7.5.2 Improved Nursery	2-6
2.7.5.3 Debugging Performance	2-7
2.8 New Features and Enhancements in the Oracle JRockit JVM R27.2	2-7
2.8.1 Java SE 6 Support	2-7
2.8.2 Java 1.4.2 and 5.0 Updates	2-8
2.8.3 New Platform Support	2-8
2.8.4 Attach API Support	2-8

2.8.5	Improved System.nanoTime Resolution	2-8
2.8.6	Performance Improvements	2-8
2.8.6.1	Improved Nursery Implementation	2-8
2.8.6.2	Improved Software Prefetching	2-8
2.8.6.3	Improved Garbage Collection Heuristics	2-9
2.8.6.4	Examples of Performance Improvements	2-9
2.8.6.4.1	SPECjbb2005	2-9
2.8.6.4.2	WLSS Benchmark	2-10
2.8.7	Supportability Features	2-11
2.8.8	JRA Improvements	2-11
2.8.9	New Command-line Options	2-11
2.8.10	Updated Command-line Options	2-11
2.9	New Features and Enhancements in the Oracle JRockit JVM R27.1	2-11
2.9.1	BEA JRockit Mission Control 2.0	2-12
2.9.2	Improved Monitoring and Diagnostics	2-12
2.9.3	Improved Supportability	2-12
2.9.4	Connect On-Demand	2-12
2.9.5	Improved Documentation	2-12
2.9.6	IPv6 Support	2-13
2.9.7	Expanded Support for Solaris	2-13
2.9.8	Performance Improvements	2-13
2.9.9	New Command-line Options	2-14
2.9.10	Updated Command-line Options	2-15
2.10	Changes in the Oracle JRockit JVM R27.7	2-16
2.11	Changes in the Oracle JRockit JVM R27.6	2-16
2.11.1	Changes in the Oracle JRockit JVM R27.6.9	2-16
2.11.2	Changes in the Oracle JRockit JVM R27.6.8	2-16
2.11.3	Changes in the Oracle JRockit JVM R27.6.7	2-16
2.11.4	Changes in the Oracle JRockit JVM R27.6.6	2-17
2.11.5	Changes in the Oracle JRockit JVM R27.6.5	2-17
2.11.6	Changes in the Oracle JRockit JVM R27.6.4	2-18
2.11.7	Changes in the Oracle JRockit JVM R27.6.3	2-19
2.11.8	Changes in the Oracle JRockit JVM R27.6.2	2-19
2.11.9	Changes in the Oracle JRockit JVM R27.6.1	2-19
2.11.10	Changes in the Oracle JRockit JVM R27.6.0	2-20
2.12	Changes in the Oracle JRockit JVM R27.5	2-21
2.13	R27.3.1 Umbrella Patch for WLS 10.0 MP1 Now Available	2-24
2.14	Changes in the Oracle JRockit JVM R27.4	2-24
2.15	Changes in the Oracle JRockit JVM R27.3	2-26
2.15.1	Known Issues in the BEA JRockit JVM R27.3.1	2-26
2.15.2	Changes in the BEA JRockit JVM R27.3	2-27
2.16	Changes in the Oracle JRockit JVM R27.2	2-29
2.17	Changes in the Oracle JRockit JVM R27.1	2-31
2.18	Known Issues	2-32
2.18.1	Workarounds for CR349882	2-40
3.1	New Features and Enhancements in JRockit R26.4	3-1
3.1.1	Performance Improvements on 64-bit Platforms	3-1

3.1.2	Performance Improvements for Low Latency Applications	3-2
3.1.3	Performance Improvements on SPARC	3-3
3.1.4	Additional Tuning Possibilities	3-3
3.2	New Features and Enhancements in JRockit R26.3	3-4
3.3	New Features and Enhancements in JRockit R26.2	3-4
3.4	New Features and Enhancements in JRockit R26.1	3-4
3.5	New Features and Enhancements in JRockit R26.0	3-4
3.6	Most Recent Changes	3-4
3.6.1	Changes in the JRockit R26.4 Release	3-4
3.6.2	Changes in the JRockit R26.3 Release	3-6
3.6.3	Changes in the JRockit R26.2 Release	3-8
3.6.4	Changes in the JRockit R26.0 Release	3-9
3.7	Known Issues	3-12

Preface

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

About this Document

This document includes the following chapters:

- [Chapter 1, "General Release Information"](#), which contains important release information for Oracle JRockit JDK.
- [Chapter 2, "R27 Release Information"](#), which describes important details about Oracle JRockit JDK R27.
- [Chapter 3, "R26 Release Information"](#), which describes release information about Oracle JRockit JDK R26.

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.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

General Release Information

This document contains important release information for Oracle JRockit JDK. It contains information on the following subjects:

- [Section 1.1, "Version Naming"](#)
- [Section 1.2, "Java Support"](#)
- [Section 1.3, "Platform Support"](#)
- [Section 1.4, "Installation"](#)
- [Section 1.5, "Documentation Accompanying the Oracle JRockit JDK"](#)

1.1 Version Naming

The Oracle JRockit JDK version number consists of two parts: the Java SE version and the Oracle JRockit JVM version. For example, Oracle JRockit JDK 6 R27.6 supports Java Standard Edition 6 and contains the JRockit JVM R27.6.

Oracle licenses and bundles the Java class libraries from Sun Microsystems. For the exact version of the Sun class libraries shipped with this version of the JRockit JDK, see [Java Support](#).

1.2 Java Support

JRockit JDK Java certification depends upon the specific version of the product, as outlined here:

- Oracle JRockit JDK R27.7.2 is certified to be compatible with J2SE 5.0 Update 34, and Java SE 6 Update 31
- Oracle JRockit JDK R27.7.1 is certified to be compatible with J2SE 1.4.2_32, J2SE 5.0 Update 30, and Java SE 6 Update 26
- Oracle JRockit JDK R27.6.9 is certified to be compatible with J2SE 1.4.2_30, J2SE 5.0 Update 28, and Java SE 6 Update 24
- Oracle JRockit JDK R27.6.8 is certified to be compatible with J2SE 1.4.2_28, J2SE 5.0 Update 26, and Java SE 6 Update 22
- Oracle JRockit JDK R27.6.7 is certified to be compatible with J2SE 1.4.2_26, J2SE 5.0 Update 24, and Java SE 6 Update 20
- Oracle JRockit JDK R27.6.6 is certified to be compatible with J2SE 1.4.2_24, J2SE 5.0 Update 22, and Java SE 6 Update 17

- Oracle JRockit JDK R27.6.5 is certified to be compatible with J2SE 1.4.2_21, J2SE 5.0 Update 19, and Java SE 6 Update 14
- Oracle JRockit JDK R27.6.4 is certified to be compatible with J2SE 1.4.2_21, J2SE 5.0 Update 19, and Java SE 6 Update 14
- Oracle JRockit JDK R27.6.3 is certified to be compatible with J2SE 1.4.2_19, J2SE 5.0 Update 17, and Java SE 6 Update 11
- Oracle JRockit JDK R27.6 is certified to be compatible with J2SE 1.4.2_17, J2SE 5.0 Update 15, and Java SE 6 Update 5
- BEA JRockit JDK R27.5 is certified to be compatible with J2SE 1.4.2_16, J2SE 5.0 Update 14, and Java SE 6 Update 3
- BEA JRockit JDK R27.4 is certified to be compatible with J2SE 1.4.2_16, J2SE 5.0 Update 14, and Java SE 6 Update 3
- BEA JRockit JDK R27.4 is certified to be compatible with J2SE 1.4.2_15, J2SE 5.0 Update 12, and Java SE 6 Update 2
- BEA JRockit JDK R27.3 is certified to be compatible with J2SE 1.4.2_14, J2SE 5.0 Update 11, and Java SE 6 Update 1
- BEA JRockit JDK R27.2 is certified to be compatible with J2SE 1.4.2_13, J2SE 5.0 Update 10, and Java SE 6
- BEA JRockit JDK R27.1 is certified to be compatible with J2SE 1.4.2_12 and J2SE 5.0 Update 8
- BEA JRockit JDK R26.4 is certified to be compatible with J2SE 1.4.2_11 and J2SE 5.0 Update 6
- BEA JRockit JDK R26.3 is certified to be compatible with J2SE 1.4.2_10 and J2SE 5.0 Update 6
- BEA JRockit JDK R26.2 is certified to be compatible with J2SE 1.4.2_10
- BEA JRockit JDK R26.1 is certified to be compatible with J2SE 5.0 Update 4
- BEA JRockit JDK R26.0 is certified to be compatible with J2SE 5.0 Update 4

1.3 Platform Support

Oracle JRockit JDK is available for J2SE 1.4.2 and 5.0, and Java SE 6. Platform support varies with the J2SE version.

- Oracle JRockit JDK 6 releases are certified on the platforms listed on the Oracle JRockit JDK 6 Supported Configurations page at:
http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/suppPlat/supp_60.html
- Oracle JRockit JDK 5.0 releases are certified on the platforms listed on the Oracle JRockit JDK 5.0 Supported Configurations page at:
http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/suppPlat/supp_50.html#998926
- Oracle JRockit JDK 1.4.2 releases are certified on the platforms listed on the Oracle JRockit JDK 1.4.2 Supported Configurations page at:
http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/suppPlat/supp_142.html#1001871

1.4 Installation

Oracle JRockit JDK is included in several products, for example Oracle JRockit Mission Control, Oracle JRockit Real Time and Oracle WebLogic. For more information, see the installation guides for your specific Oracle product.

1.5 Documentation Accompanying the Oracle JRockit JDK

The documentation that is connected to a specific version of the Oracle JRockit JDK is located here:

<http://www.oracle.com/technetwork/middleware/weblogic/documentation/weblogic-jrockit-089130.html>

R27 Release Information

This section contains important details about Oracle JRockit JDK R27. It contains information on the following subjects:

- [Section 2.1, "Latest Release"](#)
- [Section 2.2, "JDK Update Versions Supported by this Release"](#)
- [Section 2.3, "New Features and Enhancements in the Oracle JRockit JVM R27.7"](#)
- [Section 2.4, "New Features and Enhancements in the Oracle JRockit JVM R27.6"](#)
- [Section 2.5, "New Features and Enhancements in the Oracle JRockit JVM R27.5"](#)
- [Section 2.6, "New Features and Enhancements in the Oracle JRockit JVM R27.4"](#)
- [Section 2.7, "New Features and Enhancements in the Oracle JRockit JVM R27.3"](#)
- [Section 2.8, "New Features and Enhancements in the Oracle JRockit JVM R27.2"](#)
- [Section 2.9, "New Features and Enhancements in the Oracle JRockit JVM R27.1"](#)
- [Section 2.10, "Changes in the Oracle JRockit JVM R27.7"](#)
- [Section 2.11, "Changes in the Oracle JRockit JVM R27.6"](#)
- [Section 2.12, "Changes in the Oracle JRockit JVM R27.5"](#)
- [Section 2.13, "R27.3.1 Umbrella Patch for WLS 10.0 MP1 Now Available"](#)
- [Section 2.14, "Changes in the Oracle JRockit JVM R27.4"](#)
- [Section 2.15, "Changes in the Oracle JRockit JVM R27.3"](#)
- [Section 2.16, "Changes in the Oracle JRockit JVM R27.2"](#)
- [Section 2.17, "Changes in the Oracle JRockit JVM R27.1"](#)
- [Section 2.18, "Known Issues"](#)

2.1 Latest Release

The latest Oracle JRockit R27 release is R27.7.2.

2.2 JDK Update Versions Supported by this Release

Oracle JRockit R27.7.2 supports these versions of the Java JDK:

- Java 6 SE Update 31
- J2SE 5.0 Update 34

2.3 New Features and Enhancements in the Oracle JRockit JVM R27.7

The Oracle JRockit JVM R27.7 is a maintenance release and contains no new features.

2.4 New Features and Enhancements in the Oracle JRockit JVM R27.6

The Oracle JRockit JVM R27.6 is a maintenance release and contains no new features. For information on resolved issues and any changes to existing functionality in this version, please see Changes in the Oracle JRockit JVM R27.7.

JRockit Mission Control Client 3.1.0 contains a large number of new features that will provide more information more seamlessly and improve the overall user experience. For descriptions of these features, please refer to New Features and Enhancements in this Release at:

http://download.oracle.com/docs/cd/E13150_01/jrockit_jvm/jrockit/tools/relnotestools/relnotestools3.html#wp1091816

2.5 New Features and Enhancements in the Oracle JRockit JVM R27.5

Oracle JRockit R27.5 includes a number of new features and enhancements to existing features. These are described here.

2.5.1 Support for Updated Java Versions

Java version updates: 1.4.2_16, J2SE 5.0 update 14, Java SE 6 update 3.

2.5.2 Eclipse Integration of JRockit Mission Control

JRockit Mission Control is now available as an Eclipse plug-in edition. The plug-in version of Mission Control provides seamless integration of BEA JRockit's application profiling and monitoring toolset with the Eclipse development platform. By integrating Mission Control with Eclipse, you will have easy access to the powerful toolset that comprises Mission Control.

When Mission Control is run within the Eclipse IDE, you have access to IDE features that aren't otherwise available in the toolset when it is run as a standalone Rich Client Platform (RCP) application. The most significant of these features is the ability to see specific code in the running application by opening it directly from Mission Control, a function called Jump-to-Source.

The other benefit of integrating Mission Control with the Eclipse IDE is that it allows you to profile and monitor an application during its development phase just as you would during its production phase. This allows you to spot potential runtime problems before you actually deploy your application to production; for example, you might, while monitoring an application during its development notice a memory leak. By catching the memory leak during development, you can correct it before you migrate your application to a production environment.

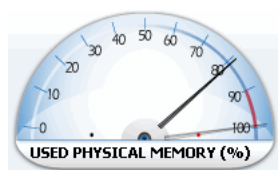
For more information, please see Integration with the Eclipse IDE or open Mission Control and launch the help system.

2.5.3 Other JRockit Mission Control Updates

The following updates have been made to JRockit Mission Control. These updates apply to both the RCP version and the Eclipse plug-in version.

- The JRockit Runtime Analyzer now shows the number of bytes of objects allocated by each Java thread.
- Three sample files that demonstrate the features of the Latency Analysis Tool have been added. The files are located at `JROCKIT_HOME/missioncontrol/samples/jrarecordings/`. The files are:
 - `pricing_server_logging_on.jra`
 - `pricing_server_logging_off.jra`
 - `java2d_demo.jra` (This file is a recording of the demo located at `JROCKIT_HOME/demo/jfc/Java2D`. The Java2D demo folder contains the source, allowing this recording to demonstrate Jump-to-Source (Jump-to-Source is only available when you are running Mission Control within Eclipse, as described in Eclipse Integration of JRockit Mission Control).
- Small adjacent Latency Analysis Tool (LAT) events of the same type are now clearly marked to make them easier to distinguish.
- Configurable velocimeters (Figure 2-1) have been added to the Console

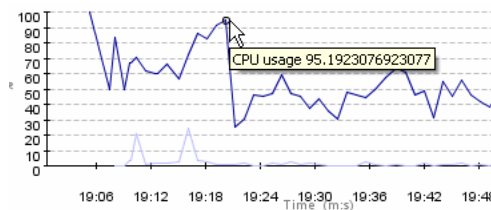
Figure 2-1 Configurable Velocimeter



- You can see the exact numerical value for a point in a graph in a tooltip by hovering your mouse pointer at the point.

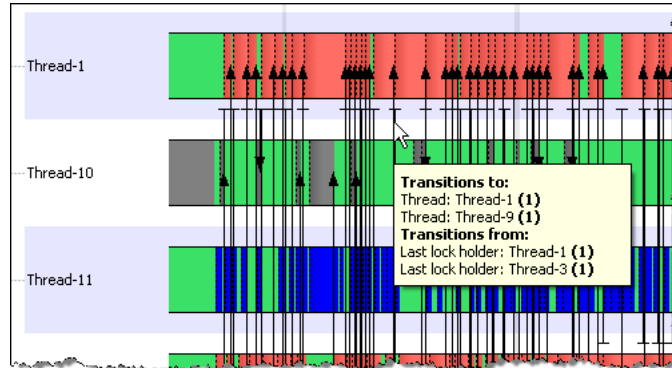
Note: This feature is only available when the graph is frozen.

Figure 2-2 Displaying the Value for a Point on a Graph



- The time ranges of graphs shown on the same page can be synchronized.
- You can filter attributes by name in the attribute browser when you select attributes to add to a graph or similar.
- Thread transitions—a latency event in one thread that is associated with another thread—are now displayed as small black arrows on the Latency Graph, as shown in Figure 2-3. By hovering your pointer over a transition arrow, a tooltip will appear, describing the transition.

Figure 2–3 Arrows Depict Thread Transitions in LAT; Tooltip Describes Selected Transition



2.5.4 Updated Command-line Options

This version of Oracle JRockit includes updates to some of the command-line options used at startup.

2.5.4.1 -Xverbose

A new verbose logging module, `-Xverbose:refobj` has been added. At info level this module provides low overhead information on `java.lang.ref.Reference` objects at each garbage collection. At the debug level, this module prints out information equivalent to the info level printouts from the old `-Xverbose:referents` module. For more information about this command, see:

http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/refman/optionX.html#wp999543

2.5.4.2 -XpauseTarget

The `-XpauseTarget` value can now be set as low as 1 ms. The real minimum pause target still depends on the application size and behavior and the hardware. For more information, see http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/refman/optionX.html#wp999535.

2.5.4.3 -XlargePages

By default the JVM will continue running without large pages if large pages cannot be acquired when `-XlargePages` is enabled. This option now can use the parameter `exitOnFailure=false` to override this behavior and force the JVM to exit if enough large pages cannot be acquired; for example:

```
-XlargePages:exitOnFailure=false
```

For more information, see http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/refman/optionX.html#wp999525.

2.6 New Features and Enhancements in the Oracle JRockit JVM R27.4

Oracle JRockit R27.4 is a maintenance release and contains no new features. New features available in the associated version of JRockit Mission Control (JRockit Mission Control 3.0.1) are described in that product's Release Notes at:

http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/tools/relnotestools/index.html

2.7 New Features and Enhancements in the Oracle JRockit JVM R27.3

This section describes new features and enhancements released in this version of BEA JRockit. It includes information on the following subjects:

- [Java Updates](#)
- [BEA JRockit Mission Control 3.0](#)
- [GUI Localizaton](#)
- [Localized Documentation](#)
- [Performance Improvements](#)

2.7.1 Java Updates

BEA JRockit R27.3 has been updated to use J2SE 1.4.2_14, J2SE 5.0 Update 11, and Java SE 6 Update 1.

2.7.2 BEA JRockit Mission Control 3.0

An updated version of BEA JRockit Mission Control is bundled with BEA JRockit R27.3. For a full description on what the release contains, see the BEA JRockit Mission Control Release Notes at: http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/tools/relnotestools/index.html.

2.7.3 GUI Localizaton

The Oracle JRockit Mission Control GUI is now also available in Japanese and simplified Chinese.

2.7.4 Localized Documentation

Later in the summer of 2007, documentation also will be available in Japanese and simplified Chinese.

2.7.5 Performance Improvements

Performance has improved in the following areas:

2.7.5.1 Better Out of the Box Performance

This version of BEA JRockit includes improvements to the following areas, resulting in better out of the box performance:

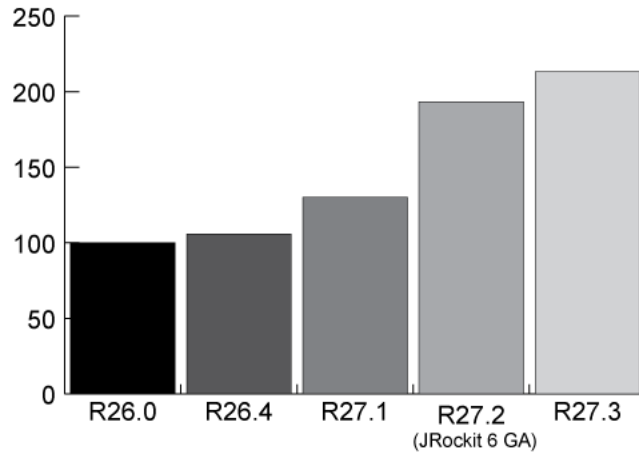
- TLA size
- TLA handling
- Nursery resizing

Note: This means that you rarely need to tune `-XXtlaSize`, `-XXlargeObjectLimit`, and `-Xns` (nursery size). For more information about these commands, see:

http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/refman/optionXX.html

Figure 2-4 demonstrates the release-to-release improvements to out of the box performance. The large increase between R27.1 and R27.2 coincides with the introduction of JRockit for Java SE 6. R27.3 contains further enhancements.

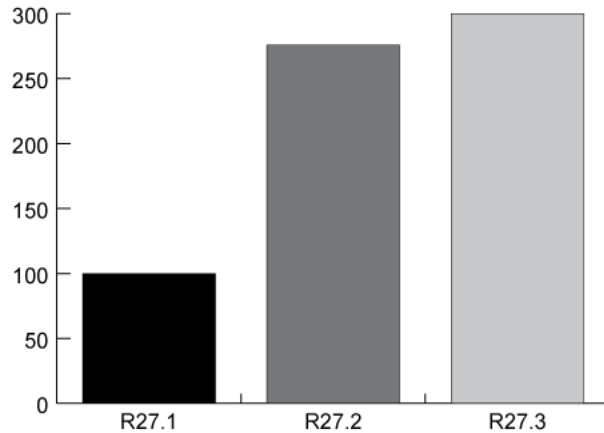
Figure 2-4 *SPECjbb2005 Out of the box improvements from R26.0 to R27.3*



2.7.5.2 Improved Nursery

A new nursery implementation was introduced in R27.2, providing significant enhancements to application throughput as well as garbage collection pause times. This implementation has been refined in R27.3, leading to further improvements in performance (see Figure 2-5).

Figure 2-5 *SIP benchmark results—further improvements from R27.2*



2.7.5.3 Debugging Performance

Single-stepping in debug mode on single-CPU machines is now significantly faster than in previous JRockit versions.

2.8 New Features and Enhancements in the Oracle JRockit JVM R27.2

This section describes new features and enhancements released in this version of BEA JRockit It includes information on the following subjects:

- [Java SE 6 Support](#)
- [Java 1.4.2 and 5.0 Updates](#)
- [New Platform Support](#)
- [Attach API Support](#)
- [Improved System.nanoTime Resolution](#)
- [Performance Improvements](#)
- [Supportability Features](#)
- [JRA Improvements](#)
- [New Command-line Options](#)
- [Updated Command-line Options](#)

2.8.1 Java SE 6 Support

BEA JRockit is now available for Java SE 6.

JRockit for Java SE 6 provides all current JRockit capabilities, including:

- Industry leading performance
- Advanced monitoring and diagnostics capabilities
- Full support for JRockit Mission Control 2.0

In addition, the Java SE 6 version of JRockit includes all generic Java SE 6 features, such as:

- XML and Web Services enhancements
 - Java Architecture for XML Binding (JAXB) 2.0
 - Java API for XML-Based Web Services (JAX-WS) 2.0
 - Streaming API for XML (StAX)
 - Web-Services Metadata
 - XML Digital-Signature APIs
- Annotations
 - Common Annotations
 - Pluggable Annotation-Processing API
- JDBC 4.0
- Scripting
- Java Compiler API

For more information about Java SE 6, please see the following web page:

<http://java.sun.com/javase/6/docs/index.html>

BEA JRockit for Java SE 6 is current available on x86 and 64-bit Xeon/AMD64 platforms.

2.8.2 Java 1.4.2 and 5.0 Updates

BEA JRockit R27.2 has been updated to use J2SE 1.4.2_13 and J2SE 5.0 Update 10.

2.8.3 New Platform Support

BEA JRockit is now supported on Windows Vista and Red Hat Enterprise Linux 5.0.

2.8.4 Attach API Support

Oracle JRockit now supports Sun Microsystem's Attach API, a Java extension that provides a way to attach tools written in Java to BEA JRockit JVM. For details, please refer to Attach API Support at:

http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/aboutjrockit.html

2.8.5 Improved System.nanoTime Resolution

The System.nanoTime() method has been improved to always use the best time resolution available on each platform. For more information about the System.nanoTime() method, see Timing with nanoTime() and CurrentTimeMillis() in the BEA JRockit Diagnostics Guide at http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/slow_start.html#wp1083972.

2.8.6 Performance Improvements

This version of JRockit includes numerous performance enhancements, including an improved nursery implementation, software prefetching, and garbage collection heuristics. These enhancements will improve performance by an average of 10% over a broad range of applications, with the largest benefits expected for memory intensive applications and out-of-the-box configurations.

The following performance improvements can be noticed for this release:

- [Improved Nursery Implementation](#)
- [Improved Software Prefetching](#)
- [Improved Garbage Collection Heuristics](#)
- [Examples of Performance Improvements](#)

2.8.6.1 Improved Nursery Implementation

The R27.2 release includes a new nursery implementation, which yields better application throughput and shorter nursery garbage collection pause times.

2.8.6.2 Improved Software Prefetching

Software prefetching, previously enabled with the options -XXallocPrefetch and -XXallocRedoPrefetch, is now enabled by default. This can improve performance by up to 40%.

Note: To fully benefit from this feature on Intel Xeon servers, it is recommended that you disable hardware prefetching in the computer's BIOS.

2.8.6.3 Improved Garbage Collection Heuristics

Nursery sizing heuristics have been improved for the default garbage collection algorithm (-Xgcprio:throughput), leading to better application throughput.

The default configuration of the -XXgcThreads option has been improved, resulting in better out-of-the-box behavior for latency sensitive applications. In most cases, there is no longer a need to tune this option manually.

2.8.6.4 Examples of Performance Improvements

To demonstrate the performance improvements in JRockit R27.2, see the following benchmark results:

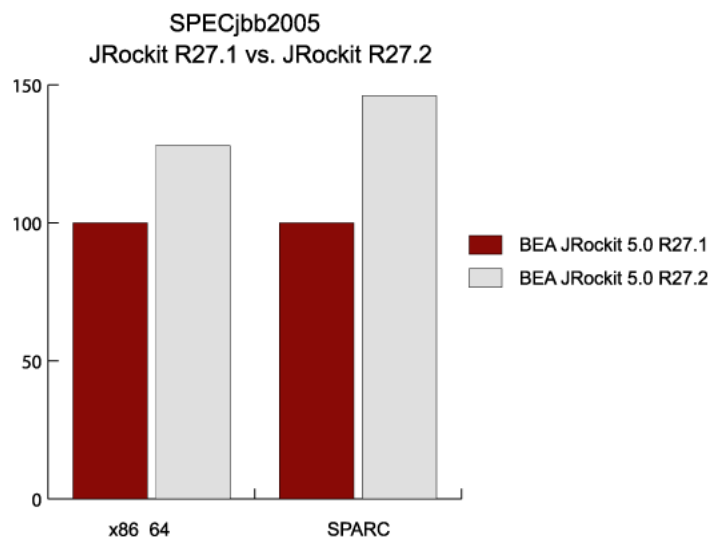
- [SPECjbb2005](#)
- [WLSS Benchmark](#)

2.8.6.4.1 SPECjbb2005

The results on the SPECjbb2005 benchmark clearly shows the performance improvements that an upgrade from R27.1 to R27.2 can bring.

In Figure 2-6, a benchmark comparison between the BEA JRockit R27.1 and R27.2 releases is shown, where the JVMs have been tuned for optimal performance with various start-up options.

Figure 2-6 SPECjbb2005 performance improvements using tuned JRockit

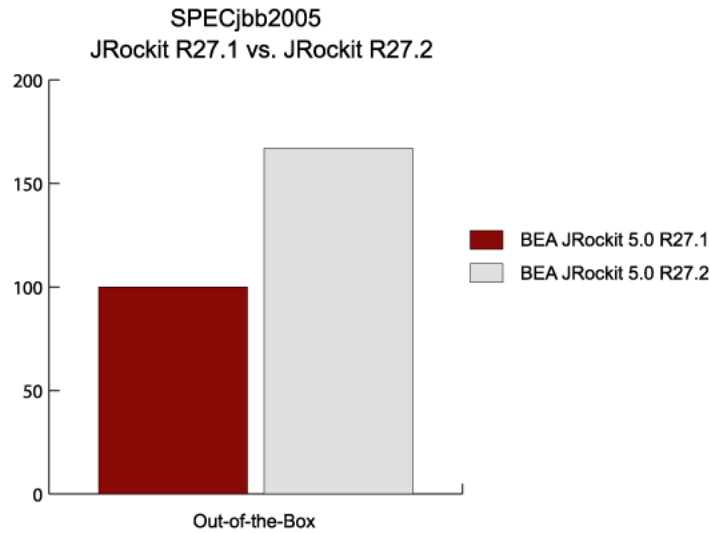


As you can see, R27.2 shows a performance increase of more than 30% compared with the R27.1 release.

The most impressive SPECjbb2005 benchmark result was generated when R27.1 and R27.2 were compared completely out of the box, without any performance tuning. Figure 2-7 demonstrates the improvements in performance that the new and improved

out-of-the-box behavior provides. The out of the box performance improvement is almost 70%.

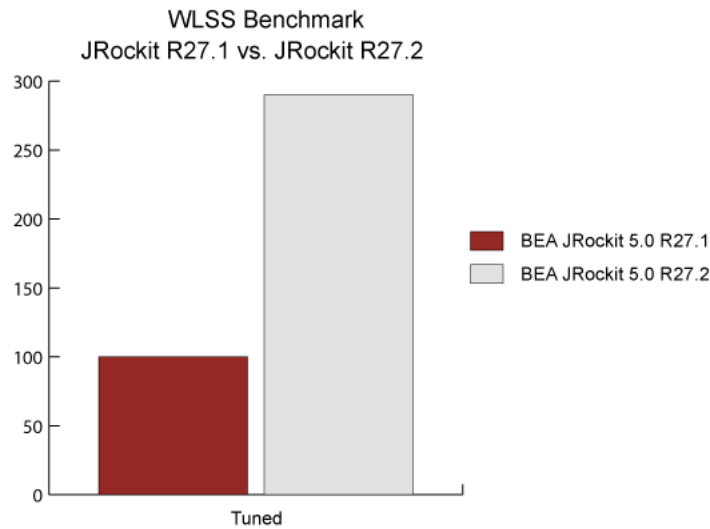
Figure 2-7 Out of the box (OOTB) performance improvements



2.8.6.4.2 WLSS Benchmark

This is a benchmark that demonstrates the performance benefits of the new nursery implementations. The application is characterized by a large number of short lived sessions, leading to high memory allocation and short lived objects. Performance is measured in calls set up per second, under the boundary condition that 95% of the call setups should be done within 50 milliseconds. These requirements imply that the application benefits from an efficient nursery. Figure 2-8 visualizes the improvement.

Figure 2-8 SIP benchmark results



2.8.7 Supportability Features

There is a new verbose module for referents. Use the start-up option `-Xverbose:referents` or the `jrcmd` parameter `verbosity set=referents=info` to make the JRockit JVM print verbose information on reference objects.

2.8.8 JRA Improvements

In BEA JRockit Runtime Analyzer, you can now see how long JRockit had been running before the start of the JRA recording. In addition, a list of all processes running on the host is included in JRA recordings.

2.8.9 New Command-line Options

The following command-line options have been added to Oracle JRockit R27.2. For a description of any option, see the Oracle JRockit JVM Reference Manual at http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/refman/index.html.

- `-XlargePages` (was `-XXlargePages`)
- `-XX:MaximumNurseryPercentage`

2.8.10 Updated Command-line Options

The command-line options listed below have been updated. For full descriptions on all command-line options, please refer to the Oracle JRockit Reference Manual at http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/refman/index.html.

-X Options:

- `-Xgc`

-XX Options:

- `-XXlargeObjectLimit`
- `-XXminBlockSize`
- `-XXtlaSize`

2.9 New Features and Enhancements in the Oracle JRockit JVM R27.1

This section describes new features and enhancements released in this version of BEA JRockit. It includes information on the following subjects:

- [BEA JRockit Mission Control 2.0](#)
- [Improved Monitoring and Diagnostics](#)
- [Improved Supportability](#)
- [Connect On-Demand](#)
- [Improved Documentation](#)
- [IPv6 Support](#)
- [Expanded Support for Solaris](#)
- [Performance Improvements](#)
- [New Command-line Options](#)

- [Updated Command-line Options](#)

2.9.1 BEA JRockit Mission Control 2.0

A completely new version of BEA JRockit Mission Control is bundled with JRockit R27.1. This version contains a large set of usability enhancements, online documentation, and even more detailed diagnostics data, see the BEA JRockit Mission Control Release Notes at http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/tools/relnotestools/relnotestools2.html.

2.9.2 Improved Monitoring and Diagnostics

The verbose logging framework in JRockit has been completely reworked. It now provides fine granular control over a large number of JVM subsystems, such as memory and threads, and it allows you to specify the amount of log data from each subsystem, log destination, and a variety of decorations, such as configurable time stamps.

Verbose logging can be controlled in several different ways:

- `-Xverbose`, `-XverboseLog`, and `-XverboseDecorations` command-line options.
- `jrcmd` tool, see Using `jrcmd` in the JRockit Diagnostics Guide.
- JRockit Management API, see the JMAPI Javadocs.

The Java management (JMX) implementation in BEA JRockit 5.0 R27.1 has been changed to require security to be enabled by default. It also requires you to specify the IP port for the management server explicitly. To revert to the old behavior, use: `-Xmanagement:ssl=false,authenticate=false,port=7091`. See `-Xmanagement` in the Reference Manual for details. This change does not affect JRockit 1.4.2.

2.9.3 Improved Supportability

This version of JRockit also includes several supportability enhancements, including improved crash files, JVM self-checks, and other features. While these features are not intended for end-users, they will facilitate communication with BEA Support and speed up problem resolution.

2.9.4 Connect On-Demand

JRockit 5.0 and later now supports the Attach API, see:

<http://docs.oracle.com/javase/6/docs/technotes/guides/attach/index.html>. This means you can connect on-demand to:

- JMX
- JVMTI, which enables you to connect on-demand from any JVMTI client supporting this functionality.

2.9.5 Improved Documentation

A completely new Diagnostics Guide has been added to the JRockit product documentation set. This guide will help you troubleshoot JRockit and your applications. For more information, see http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/index.html.

2.9.6 IPv6 Support

IPv6 is now available on all platforms supported by JRockit.

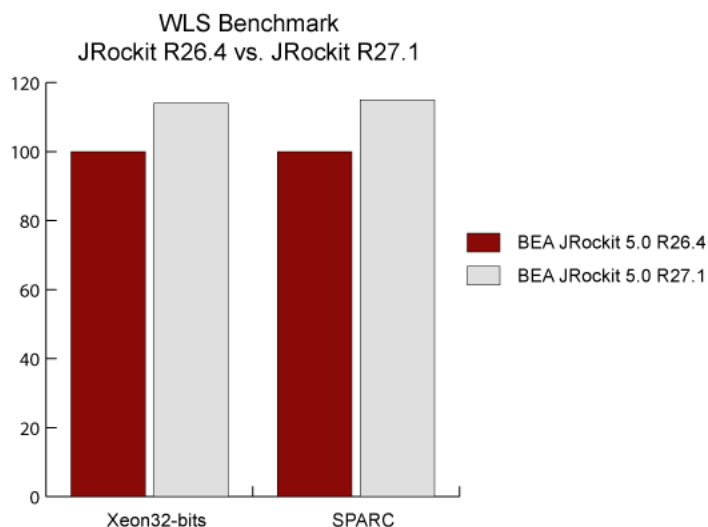
2.9.7 Expanded Support for Solaris

JRockit is now available for both J2SE 1.4.2 and 5.0 on Solaris/SPARC.

2.9.8 Performance Improvements

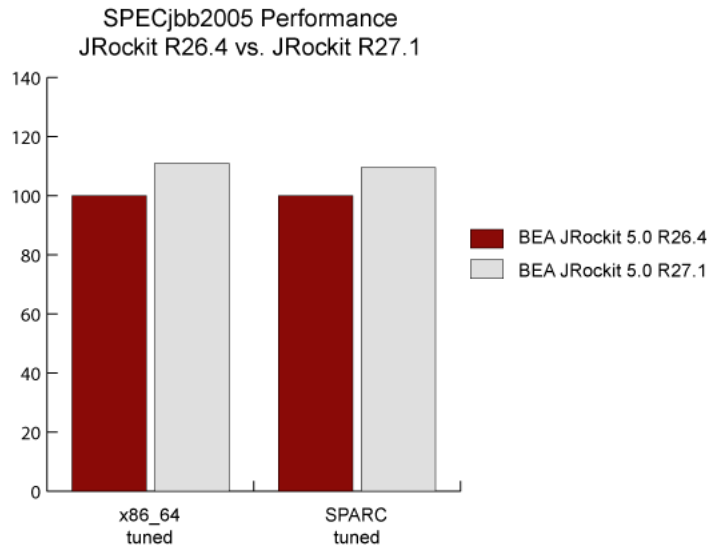
This version of JRockit includes numerous performance enhancements. One example is that performance has been improved for J2EE applications using a large number of JSP pages and servlets. These enhancements are expected to improve performance on many WLS applications by 10-15%, see Figure 2-9.

Figure 2-9 WLS Benchmark—JRockit R26.4 vs. JRockit R27.1



The release also includes a number of generic enhancements, as demonstrated by improved scores on the SPECjbb2005 benchmark. See Figure 2-10 for a comparison between the Oracle JRockit R26.4 and Oracle JRockit R27.1 releases.

Figure 2–10 SPECjbb2005 performance improvements

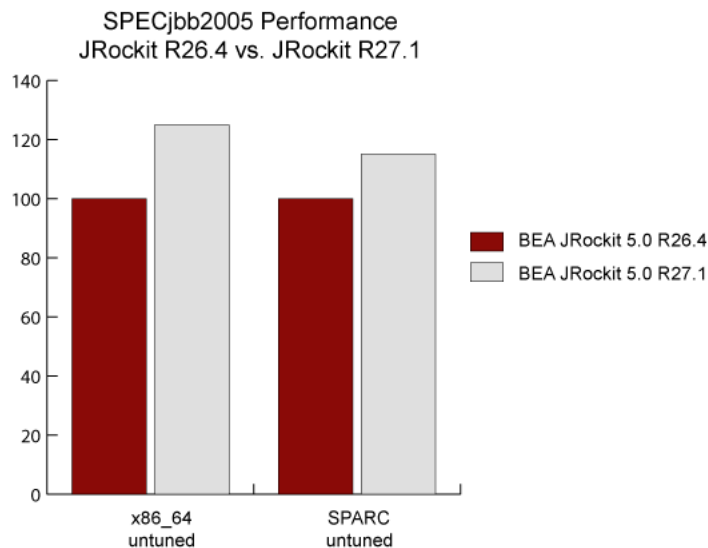


Out of the box performance has also been improved due to:

- Improved default configuration for memory allocation, see `-XXtlSize` in the Reference Manual.
- Compressed references is now available on all 64-bit platforms (x86-64, Itanium, and SPARC) and is enabled by default.

Plus other enhancements, see Figure 2-11 for a comparison between the Oracle JRockit R26.4 and Oracle JRockit R27.1 releases.

Figure 2–11 Out of the box performance



2.9.9 New Command-line Options

The following start-up commands have been added with Oracle JRockit R27. For a description of any new option, refer to the specific option in the Oracle JRockit

Reference Manual at http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/refman/index.html.

- -XXtsf
- -XverboseDecorations

2.9.10 Updated Command-line Options

The rules for how command-line parameters are parsed have been updated to avoid user confusion. Incompatible command-line combinations now cause Oracle JRockit to print out an error message and terminate. Please refer to the specific option in the Oracle JRockit Reference Manual for a description of the new behavior. See http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/refman/index.html.

Table 2–1 Updated Command-line Options

-X Options	-XX Options
-Xgc	-XXallocClearChunks
-XgcPrio	-XXallocClearChunkSize
-Xmanagement	-XXcompressedRefs
-Xms	-XXdisableFatSpin
-Xmx	-XXdisableGcHeuristics
-Xns	-XXexternalCompactRatio
-XpauseTarget	-XXfullCompaction
	-XXfullSystemGC
	-XXinitialPointerVectorSize
	-XXinternalCompactRatio
	-XXlargeObjectLimit
	-XXmaxPooledPointerVectorSize
	-XXminBlockSize
	-XXnoCompaction
	-XXnoSystemGC
	-XXpointerMatrixLinearSeekDistance
	-XXsetGC
	-XXallocPrefetch
	-XXallocRedoPrefetch
	-XXcompactSetLimit
	-XXcompactSetLimitPerObject
	-XXcompactRatio
	-XXstaticCompaction
	-XXthroughputCompaction
	-XXtlaSize
	-XXusePointerMatrix

2.10 Changes in the Oracle JRockit JVM R27.7

Oracle JRockit JVM R27.7.2 contains no major changes from Oracle JRockit JVM R27.6.9.

2.11 Changes in the Oracle JRockit JVM R27.6

This section describes the changes made in the Oracle JRockit JVM R27.6 releases:

- [Changes in the Oracle JRockit JVM R27.6.9](#)
- [Changes in the Oracle JRockit JVM R27.6.8](#)
- [Changes in the Oracle JRockit JVM R27.6.7](#)
- [Changes in the Oracle JRockit JVM R27.6.6](#)
- [Changes in the Oracle JRockit JVM R27.6.5](#)
- [Changes in the Oracle JRockit JVM R27.6.4](#)
- [Changes in the Oracle JRockit JVM R27.6.3](#)
- [Changes in the Oracle JRockit JVM R27.6.2](#)
- [Changes in the Oracle JRockit JVM R27.6.1](#)
- [Changes in the Oracle JRockit JVM R27.6.0](#)

2.11.1 Changes in the Oracle JRockit JVM R27.6.9

The following table lists changes in Oracle JRockit JVM R27.6.9.

Issue ID	Description
10110908	The calculated <code>LoadedClassCount</code> MBean exposed negative values (<code>TotalLoadedClassCount - UnloadedClassCount = LoadedClassCount</code>). This problem was triggered when too many classes were failing class file resolution. This has been fixed.

2.11.2 Changes in the Oracle JRockit JVM R27.6.8

Oracle JRockit JVM R27.6.8 contains no notable changes from Oracle JRockit JVM R27.6.7.

2.11.3 Changes in the Oracle JRockit JVM R27.6.7

The following table lists changes in Oracle JRockit JVM R27.6.7.

Issue ID	Description
9486409	Occasionally, calling <code>java.lang.util.zip.Deflater.deflateBytes()</code> after calling <code>java.lang.util.zip.Deflater.end()</code> would cause the Oracle JRockit JVM to crash. This has been fixed; now, instead of crashing, a proper <code>NullPointerException</code> is thrown.
9227236	When a user would initialize a <code>java.util.zip.ZipEntry</code> instance that had been created for an uncompressed entry (method <code>STORED</code>), the uncompressed and compressed fields would not be initialized with the same value, sometimes causing a <code>java.util.zip.ZipException</code> . This has been fixed.

2.11.4 Changes in the Oracle JRockit JVM R27.6.6

The following table lists changes in Oracle JRockit JVM R27.6.6.

Issue ID	Description
9108982	On rare occasions, a race condition was occurring when several threads were trying to initialize the same class. This has now been fixed.
9104797	With <code>-XX:+UseCfsAdaptedYield</code> , customers using the CFS thread scheduler on Linux might see pause time improvements in the garbage collector.
9059394	Due to an OS bug, Oracle JRockit running on Windows 2008 (and Vista) might have crashed during start-up when Terminal Services were installed. This has been fixed.
9050465	With SPARC implementations of Oracle JRockit that had numerous live variables and unchecked array index out of bounds exceptions, the contents in some registers could have been corrupted. This has now been fixed.
9000555	The Oracle JRockit optimizing compiler was, in rare cases, generating code that erroneously removed stores to copies of non-integer typed arrays, which was leading to incorrect computational results. This has been fixed.
8929771	In rare circumstances, SPARC Solaris implementations of Oracle JRockit was crashing when applications were trying to load a lot of code. This has been fixed.
8841076	When using bitwise operations and casting, sign extension could, under some circumstances, be lost, resulting in incorrect values. This has now been fixed.

2.11.5 Changes in the Oracle JRockit JVM R27.6.5

The following table lists changes in Oracle JRockit JVM R27.6.5.

Issue ID	Description
9032858	<p>Oracle JRockit R27.6.5 contains back ports of all Sun security fixes from the following Sun Java versions:</p> <ul style="list-style-type: none"> ■ 6 update 16 ■ 5.0 update 20 ■ 1.4.2 update 22 <p>For more details, see:</p> <p>http://www.oracle.com/technology/deploy/security/critical-patch-updates/cpuoct2009.html</p>
9027339	<p>Recent updates to the Sun JDK versions 1.6.0_14 and 1.5.0_19 created an incompatibility issue with the SSL implementation on Oracle JRockit R27.6.4 (1.6.0_14 and 1.5.0_19) and certain versions of Oracle WebLogic Server. Patches are now available to resolve this issue.</p> <p>For more information, see:</p> <p>https://webiv.oraclecorp.com/cgi-bin/webiv/do.pl/Get?WwwID=note:952078.1</p>
8870430	The reported heap usage after a garbage collection did not include data stored in one of Oracle JRockit's caches. This has been fixed.
8781977	Core file resource limits are now printed out correctly in the Oracle JRockit textual dump file.

Issue ID	Description
8770723	Under very rare circumstances, some optimized methods would perform faulty arithmetic. This has been fixed.
8740199	When optimizing <code>System.arraycopy()</code> the Oracle JRockit compiler was, in some cases, failing to recognize explicit types checks on the destination array, which could have caused semantic errors in the optimized code. This has been fixed.
8649273	The Oracle JRockit textual dump file now includes the value of <code>LD_PRELOAD</code> on Linux and <code>LD_PRELOAD, LD_PRELOAD_32, LD_PRELOAD_64</code> on Solaris.
8567923	Under some circumstances, when using the <code>genconcon</code> collector, Oracle JRockit would corrupt the free list. This has been fixed.
8258685	Sometimes, <code>JVMFactory.getJVM().getMachine().getPhysicalMemory().getUsedMemory()</code> would return 0. This has now been fixed.
8235000	Oracle JRockit JDK 5.0 and 1.4.2 could crash in method <code>Java_sun_dc_pr_PathFiller_writeAlpha8</code> (Sun bug #6464341). This has been fixed in Oracle JRockit 5.0 R27.6.5 and Oracle JRockit 1.4.2 R27.6.5.
8173162	Oracle JRockit JDK 5.0 and 1.4.2 was hanging when reading JPEG images with embedded ICC profiles (Sun bug #4528643, #6295525). This issue has been fixed in Oracle JRockit 5.0 R27.6.5 and Oracle JRockit 1.4.2 R27.6.5.
7690841	Threads waiting on a <code>java/nio/channels/Selector.select()</code> call on Windows platforms were not being released if interrupted. This has been fixed.

2.11.6 Changes in the Oracle JRockit JVM R27.6.4

The following table lists changes in Oracle JRockit JVM R27.6.4.

Issue ID	Description
8583868	In rare circumstances involving malformed zip-files, Oracle JRockit could crash. This has been fixed.
8567164	Oracle JRockit was leaking memory when notification was used on <code>MemoryMXBeans</code> . This has been fixed.
8551200	Under some circumstances <code>BigDecimal.valueOf()</code> would return null instead of zero. This has been fixed.
8548547	During certain optimizations, the thread would run out of stack memory. This has been fixed.
8545562	<code>-Xverbose:stackoverflow</code> now works correctly when printing full stack traces upon Stack Overflow Errors.
8479419	Oracle JRockit was crashing on Windows platforms when it was recursively taking spinlocks exactly 16 times on bytecode compiled by Oracle Java Compiler (OCJ). This has been fixed.
8405290	Previously, when running <code>bin/jrcmd -f <file></code> , only the first line of the file was run. This has been fixed.
8343063	On the Sparc platform, the contents of a register could be overwritten by an exception handler if the method contained many local variables. This has been fixed.

Issue ID	Description
8194509	The Oracle JRockit optimizing compiler was occasionally generating erroneous code resulting in Null Pointer Exceptions being thrown from <code>org.eclipse.jdt.internal.compiler.lookup.ParameterizedTypeBinding.getMethods()</code> of the Eclipse IDE. This has been fixed.
8178388	Oracle JRockit can now handle zip- and jar-archives larger than 65535 files.

2.11.7 Changes in the Oracle JRockit JVM R27.6.3

The following table lists changes in Oracle JRockit JVM R27.6.3.

Issue ID	Description
8282822	On 64-bit Windows Vista/2008, the 64-bit JRockit installer executable is now properly manifested to request the administrator execution level. This issue was originally tracked in CR363487.
8187650	Starting with Oracle JRockit Mission Control 3.1.0 and Oracle JRockit Real Time 3.1.0, each product installation directory is also an Oracle home, allowing additional product registration with the Oracle Central Inventory. The Central Inventory contains information about all installed Oracle products on the same host and can be managed by running the Oracle Universal Installer.
8186383	A condition in which a JRockit JDK installation running applications on compositing window managers under Linux might fail and generate the following error message has been fixed in Oracle JRockit 5.0 R27.6.3 and Oracle JRockit 6 R27.6.3: <code>xcb_xlib.c:50: xcb_xlib_unlock: Assertion 'c->xlib.lock' failed.</code> This issue was originally tracked by CR349882.
8179196	Starting with Oracle JRockit Mission Control 3.1.0 and Oracle JRockit Real Time 3.1.0, the demo and sample programs and the source code of the Java platform are no longer installed by default. They have been separated into optional components that you explicitly select in order to install.

2.11.8 Changes in the Oracle JRockit JVM R27.6.2

The following table lists changes in Oracle JRockit JVM R27.6.2

Issue ID	Description
7574848	Certain XSLT/XSL transformations could on occasion throw <code>ClassCastExceptions</code> . This has now been fixed.

2.11.9 Changes in the Oracle JRockit JVM R27.6.1

The following table lists changes in Oracle JRockit JVM R27.6.1

Issue ID	Description
8187000	The JRA could cause a JVM crash at the end of a recording, while zipping the file, if the JVM process was low on memory. This has been fixed and JRA will now output an error message instead.
8184406	3D rendering with extensive use of xmm registers could occasionally cause the JVM to crash. This has now been fixed.

Issue ID	Description
8183855	External compaction missed a part of the heap when the area to compact shrunk. This is now fixed.
8181106	Oracle JRockit versions R27.5 and R27.6 can crash when executing optimized code for the <code>java/lang/StringBuffer.lastIndexOf()</code> method on 64-bit platforms. The compiler no longer generates broken code for this method.
8180938	Oracle JRockit could, on occasion, crash while optimizing methods, resulting in stack traces pointing to <code>removePhi</code> . This has now been fixed.
8178677	Occasionally, reading the Manifest from an <code>rt.jar</code> distribution using <code>JarInputStream</code> returns NULL for certain <code>rt.jar</code> distributions. This has been fixed.
8178674	Oracle JRockit was crashing due to corruption of internal data structures while garbage collecting heaps with pinned objects allocated close to the nursery keep area. This has been fixed.
8176433	The Oracle JRockit optimizing compiler could generate faulty x86 code, leading to erroneous results for Java code operating on floating-point return values of method calls. This has been fixed.
8175793 8174814 8124284	The Oracle JRockit optimizing compiler can consume large amounts of memory when compiling large methods with complex control-flow (such as large JSPs). On 32-bit platforms this can lead to OutOfMemory errors. This has been fixed.
8097260	JRockit Mission Control was unable to open LAT recordings that contained threads that didn't have names. This has been fixed.

2.11.10 Changes in the Oracle JRockit JVM R27.6.0

The following table lists changes in Oracle JRockit JVM R27.6.0.

Issue ID	Description
CR371396	When a full compaction was issued, all references in the heap wanted to be stored in the compact set. If a limit was set on the compact set, it would likely skip the compaction due to too many pointers. This has been fixed.
CR366936	In previous releases of the JRockit JVM, unloading class hierarchies that had many unloaded types sharing a common superclass or implementing the same interface could cause very long pause times. This has been improved in R27.6 .
CR366265	Technical license checks have been removed.
CR366238	The JRockit JVM running with WebLogic Event Server on Sparc systems was crashing in <code>cgStoreMetaInfo</code> , indicating a known issue in delay slot scheduling. The delay slot scheduling bug has been fixed and the system is no longer crashing.
CR364912	In previous versions of the JRockit JVM, a thread suspended in a debugger in a call chain for a static initializes could cause a crash while reading local variable information on the receiving type for the method initializing the class. This has been fixed.
CR361911	Calls to <code>java.lang.management.CompilationMXBean.getTotalCompilationTime()</code> previously returned 0. This has been fixed.

Issue ID	Description
CR361457	Connecting the Memory Leak Detector to a running JRockit JVM built for Linux IA32 could cause the JVM to crash if the JVM process used more than about 1020 file descriptors at that time. This has now been fixed.
CR360910	The JRockit JVM did not properly handle error codes returned from Agent_OnAttach. If an error was returned, the JVM was aborted. This has been fixed.
CR359309	The ACLs (Access Control Lists) on the per-user <TEMP>\hspcrfdata_<user> directory are set up such that users with administrator privileges have the rights to delete the directory. This only affects Windows versions of the JRockit JVM. For more information, please see: http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=5073453 .
CR358260	When a large number of classes (~10,000) implementing the same interface were unloaded at the same time, unregistering them from the Interface could take a long time. This has been fixed
CR357526	The maximum number of active Object monitors has been increased to 4,194,304.
CR357397	The nursery size was slowly shrinking until it was almost zero. Eventually, the young collection stops reclaiming any space at all, although no old collection is triggered. Thus, the JRockit JVM just continues to do young collections. This has been fixed.
CR355985	After upgrading from an earlier version of the JRockit JVM to R27.4, some users experienced a change in class loader behavior: a NoClassDefFoundError was thrown if the \$Inner class was not present in the CLASSPATH. This has been fixed.
CR355465	An operating system bug in some operating systems caused signals to get lost if they were sent at the exact time of a call to pthread_create. This could cause the JRockit JVM to hang. This has been solved by blocking those signals when calling pthread_create.
CR355117	On previous versions of the JRockit JVM, setting back the system clock would cause calls to Thread.sleep to sleep longer than intended. This has now been fixed.
CR330541	The new fontconfig.properties file for CJK (Chinese, Japanese, Korean) locales compatible with RHEL 5 and Asianux 3.0 has been fixed by backporting the fix from Oracle JRockit JDK 6 to Oracle JRockit JDK 5.0. This is a backport of a fix for the problem reported by Sun in http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=2149631 .
CR328979	In versions R27.1 to R27.5 of JRockit JVM, delay slot scheduling issues on SPARC resulted in crashes during garbage collection. This has been fixed.
CR236722	The JRockit JVM is more robust when handling JNI threads which have terminated without calling DetachCurrentThread. The JVM will now try to detect these threads and ignore them. The JRockit JVM will print a warning message when this happens to alert developers to the problem. However, note that by not calling DetachCurrentThread, the JNI API is violated, and even though The JRockit JVM is now more robust, it can never guarantee the stability when the API is violated in this way.

2.12 Changes in the Oracle JRockit JVM R27.5

The following table lists changes in this version of the BEA JRockit JVM.

Issue ID	Description
CR359828	In earlier versions of Oracle JRockit, the value for Heap Usage Before for a garbage collection in a JRA recording was incorrect, as it actually showed the Heap Usage After for the proceeding collection instead. This is now fixed.
CR358662	Oracle JRockit would occasionally crash during optimization of methods containing two or more different chains of String object concatenations. This has been fixed.
CR356984	A bug was causing code optimization to crash in function <code>_irTypesIsExactClass</code> . This has been fixed.
CR354539	Oracle JRockit was occasionally leaving sockets in a <code>CLOSE_WAIT</code> state on Windows when using <code>java.nio</code> channel selectors. This has been fixed.
CR354463	Two threads calling <code>getThreadInfo</code> or <code>getStackTrace</code> on each other was causing Oracle JRockit to deadlock. This has now been fixed.
CR352232	The command line option <code>-XlargePages</code> now can use the parameter <code>exitonfailure=false</code> to override its default behavior and force the JVM to exit if enough large pages can't be acquired; for example: <code>-XlargePages:exitOnFailure=true</code> .
CR350569	On some occasions, calling <code>inflate</code> on a closed <code>Inflater</code> would make BEA JRockit crash, creating a core file. Now, JRockit will instead throw a <code>NullPointerException</code> .
CR350009	The verbose module <code>referents</code> (with the alias <code>verboserefs</code>) has been replaced by the new <code>refobj</code> module. Using <code>-Xverbose:refobj</code> will provide simpler but, performance-wise, cheaper statistics about reference objects. An output similar to the detailed output from the old <code>referents</code> module is provided by <code>-Xverbose:refobj=debug</code> . Please note that this output, like the old <code>referents</code> module, is performance-wise very costly. If you use the old <code>referents</code> module, it will be converted into <code>refobj=debug</code> .
CR349794	The crash dump summary now includes more detailed information on the memory system.
CR348007	A mistake in the heuristics for heap expansion that caused Oracle JRockit to throw an Out of Memory error when it should have expanded the heap has been fixed.
CR347294	A direct link has been added from crash files to the Oracle JRockit documentation. Now, if Oracle JRockit crashes, you can click this link to open the troubleshooting documents.
CR346988	A bug in the stack trace printing code caused Oracle JRockit to crash in some circumstances when printing stack traces. This happened regularly when starting WebLogic Server with <code>-Xverbose:exceptions=debug</code> . This has been fixed.
CR345875	Calling <code>RetransformClasses</code> via <code>JVMTI</code> would sometimes fail with error code <code>JVMTI_ERROR_INVALID_CLASS_FORMAT</code> . This condition has been fixed.
CR345588	The <code>JMAPI</code> call <code>com.bea.jvm.GarbageCollector.hasCompaction()</code> returned incorrect values. This has been fixed.
CR345574	If Oracle JRockit was started with <code>-Xcheckjni</code> a thread reference passed into a <code>JVMTI</code> callback function was used in a <code>JNI</code> call, the JVM would exit with an error saying <code>[ERROR][native] Invalid reference</code> . This has been fixed.

Issue ID	Description
CR344773	In some reports generated by the <code>-Xgcreport</code> flag, garbage collection pauses would sometimes appear to be longer than the garbage collection itself. This has been resolved.
CR342358	The minimum pause target limit has been lowered to less than 5 ms. The actual minimum target depends upon the application you are running.
CR340660	Calling the methods <code>isLoading()</code> , <code>load()</code> or <code>force()</code> in <code>java.nio.MappedByteBuffer</code> on an empty buffer would throw an <code>IOException</code> . This has been fixed.
CR340016	In R27.5 the generation of exception stacktraces has been changed to always include the full stacktrace, regardless of whether the exceptions were generated asynchronously (<code>NullPointerException</code> , <code>StackOverflowError</code> , etc) or lazy stacktraces is on. As a consequence of this, the verbose module <code>stackoverflow</code> has been deprecated and is now a no-op. This module will be fully removed in the next major release.
CR339531	For some customers using BEA JRockit R27.3.1, <code>AssertionError: unused</code> was thrown. This happened because the <code>ClassLoader</code> was fed class/package names in the wrong format. Instead of receiving <code>com.bea.MyClass</code> it received <code>com/bea/MyClass</code> . This has been fixed.
CR339281	BEA JRockit 5.0 Update 14 R27.5.0 adds support for Sun PKCS#11 provider (http://java.sun.com/j2se/1.5.0/docs/guide/security/p11guide.html) for Linux on Itanium in addition to Linux on 64-bit Xeon/AMD64 (Sun Bug ID 6467921).
CR338872	In R27.1, the class bytes preprocessing facility was changed to allow for recursive preprocessing. This meant that a classpreprocessor instance that was currently doing class preprocessing and through this caused a new class to be loaded would be recursively called with the new classbytes. This caused failures in some existing preprocessor implementations that relied on the old pre-27.1 behavior. In R27.5, this has been reverted. A thread doing class preprocessing will now silently refuse to preprocess any types created by executing the preprocessor itself.
CR335834	Because of the much larger amount of suspensions that lazy unlocking introduces, Oracle JRockit users running on Windows IA64 were often bumping into the <code>GetThreadContext</code> bug. Lazy unlocking is now enabled by default in the Java 6 version of BEA JRockit R27.5 on all platforms except IA64 and for all garbage collection strategies except the deterministic garbage collector. In older releases you can enable lazy unlocking with the command line option <code>-XXlazyUnlocking</code> .
CR335688	The garbage collection strategy is now correctly reported when a static concurrent garbage collector uses a parallel mark or sweep phase as an emergency action when the heap has become full before or during garbage collection. <code>-XXdisableGcHeuristics</code> now disables all strategy changes, including emergency parallel mark or sweep in the static concurrent garbage collectors.
CR333688	In R27, a bug was introduced that would cause memory leaks whenever a JVMTI agent with the <code>can_retransform_classes</code> capability replaced byte code of a class being loaded. This would also impact byte code preprocessing done through the JRockit Management API (JMapi). This bug has been fixed in R27.5.
CR329800	JRockit Mission Control 3.0 did not properly detect license errors from JRockit R27.3.0 when starting a JRA/LAT recording or opening Memleak. This has been fixed.

Issue ID	Description
CR328964	-XXcompactSetLimit is now always respected. Note however, that this only applies to references outside the compaction area. The number of references inside the compaction area is not limited by this flag.
CR325082	A rare occurrence in the register allocation code was causing Oracle JRockit to crash. This has been fixed.
CR306735	Oracle JRockit will no longer accept memory sizes that are larger than the address space on the current platform. In practice, this means that on a 32-bit system, the value given to -Xmx, -Xms and -Xns cannot be larger than 4 GB, or Oracle JRockit won't start.

2.13 R27.3.1 Umbrella Patch for WLS 10.0 MP1 Now Available

An umbrella patch of Oracle JRockit JVM R27.3.1 has been built for distribution with WLS 10.0 MP1. You can download this patch as a zip (Windows) or tar.gz (Linux) file from commerce.bea.com.

The patch include fixes for the CRs listed in the following table.

Issue ID	Description
CR344232	On some rare occasions, Oracle JRockit would crash when allocating memory. This only happened when a call to <code>mmAllocObjectOrArray</code> tried to allocate <code>largeObjectLimit</code> bytes that were exactly the same size as the TLA fetched.
CR339531	Disabling assertions did not work for ClassLoader-managed assertions. This could result in Assertion Errors when starting WLS in debug mode. The reason for this was that the ClassLoader was fed wrong class/package names. This has been fixed. This issue has been resolved in Oracle JRockit R27.3.1 CP but not in Oracle JRockit R27.4. It will be fixed in Oracle JRockit R27.5, scheduled for release in early 2008.
CR336511	A patch has been built and backported for all publicly known security issues in Oracle JRockit R27.3.1. This fix corresponds to the security issues in BEA Security Advisory BEA07-177.00 and BEA07-178.00.
CR331724	When running AquaLogic Service Bus SFTP tests, BEA JRockit was creating regular core dumps. This issue occurred when two mutually exclusive code paths doing an arraycopy to the same to-array were both subject to an erroneous optimization.
CR328154	In some Solaris environments, BEA JRockit was unable to detect the number of sockets, cores and hardware threads. This caused the JVM to abort during start up and display error message: <code>[ERROR] Fatal error in JRockit during memory setup phase.</code> This situation would occur in a local zone associated with a subset of all available processors. This issue has been resolved.
CR326728	A customer experienced a system crash in <code>mmListAddLast</code> . This has been fixed.

2.14 Changes in the Oracle JRockit JVM R27.4

The following table lists changes in this version of the BEA JRockit JVM.

Issue ID	Description
CR345588	The JMAPI call <code>com.bea.jvm.GarbageCollector.hasCompaction()</code> was returning incorrect values. This has been fixed.
CR345574	If the JVM was started with <code>-Xcheckjni</code> and a thread reference passed into a JVMTI callback function was used in a JNI call, the JVM would exit with the error <code>[ERROR][native] Invalid reference</code> . This has been fixed.
CR336996	In earlier versions of Oracle JRockit, calling the JVMTI function <code>IterateOverHeap</code> with <code>JVMTI_HEAP_OBJECT_TAGGED</code> and then <code>JVMTI_HEAP_OBJECT_UNTAGGED</code> would sometimes report more objects than doing <code>IterateOverHeap</code> with <code>JVMTI_HEAP_OBJECT_EITHER</code> . This has been fixed.
CR336790	Modifying a next pointer through reflection was causing a memory leak when processing phantom reference objects. This has been fixed in this version of Oracle JRockit.
CR336285	Sometimes the OS would fail to suspend a thread, which lead to Oracle JRockit crashing and throwing an <code>EXCEPTION_ACCESS_VIOLATION</code> error. This is now fixed.
CR334151	In earlier versions of Oracle JRockit, the JVM could hang while doing a Latency Analysis Recording and recording a park event. This has been fixed.
CR332182	In earlier versions of Oracle JRockit, the <code>java.lang.management.ThreadInfo</code> API and the JVMTI functions <code>GetOwnedMonitorStackDepthInfo</code> and <code>GetOwnedMonitorInfo</code> did not return monitors that had been entered by calling the JNI function <code>MonitorEnter</code> . This has been fixed.
CR332016	In some cases when allocation of a large array failed, the JVMTI <code>ResourceExhausted</code> event was not sent. This has been fixed.
CR332012	In earlier versions of Oracle JRockit, the <code>jvmtiHeapReferenceCallback</code> callback function was sometimes called with the wrong <code>class_tag</code> parameter. This has been fixed.
CR332002	In earlier versions of Oracle JRockit, the JVMTI functions <code>FollowReferences</code> and <code>IterateThroughHeap</code> did not respect the <code>klass</code> parameter. This has been fixed.
CR331991	In earlier versions of Oracle JRockit, the JVMTI function <code>GetClassVersionNumbers</code> did not return <code>JVMTI_ERROR_ABSENT_INFORMATION</code> for primitive and array classes. This has been fixed.
CR331724	When running AquaLogic Service Bus SFTP tests, Oracle JRockit was creating regular core dumps. This issue occurred when two mutually exclusive code paths doing an arraycopy to the same to-array were both subject to an erroneous optimization.
CR328924	Oracle JRockit no longer fails the Java Language Specification requirement on unique references for boxed integers in the -128 to 127 interval.
CR328368	Allocation prefetch has been enabled by default on AMD's Opteron architectures.
CR328154	In some Solaris environments, Oracle JRockit was unable to detect the number of sockets, cores and hardware threads. This caused the JVM to abort during start up and display error message: <code>[ERROR] Fatal error in JRockit during memory setup phase</code> . This situation would occur in a local zone associated with a subset of all available processors. This issue has been resolved.

Issue ID	Description
CR327167	The JVMTI ClassPrepare event was previously dependant on class initialization order and thus subject to user class hierarchy design. In R27.4 this has changed so that ClassPrepare is always sent according to specification.
CR321557	The experimental and unsupported API GCControl, containing the methods <code>jrockit.ext.GCControl.forceGCToExit(boolean enabled)</code> and <code>jrockit.ext.GCControl.fullGC()</code> is now removed.
CR321348	Call profiling is an optimization feature known to provide significant benefit to many workloads, including the SPECjbb2005 benchmark. Up until Oracle JRockit R27.1, you could enable this feature by using the <code>-XXaggressive</code> command-line option, but it was removed from the flag in R27.1. As of Oracle JRockit R27.4, you can enable call profiling by using the <code>-XXcallProfiling</code> command-line option.
CR318629	Due to a bug in the attach framework (Sun bug #6559427), Mission Control was leaking several handles per locally-running JVM (JVM running on the same machine as Mission Control is) every time a Mission Control polls for locally running JVMs. This has been fixed in R27.4.
CR311802	Some customers experienced linked lists breaking, which would result in leaking objects caused by a modification of next pointers through reflection. This is now fixed.
CR304741	This version of Oracle JRockit has two new Long perf variables: <code>jrockit.gc.oc.compactionInternalCount</code> <code>jrockit.gc.oc.compactionExternalCount</code> These variables count the number of internal and external compactions, respectively. They both sum up to the previously existing <code>jrockit.gc.oc.compaction.no</code> value.
CR104868	A rewrite of an internal code generation framework for R27.4 has eliminated known bugs that were causing Oracle JRockit to crash.

2.15 Changes in the Oracle JRockit JVM R27.3

This section lists changes and known issues in the BEA JRockit R27.3.

2.15.1 Known Issues in the BEA JRockit JVM R27.3.1

The following table lists known issues in this version of the BEA JRockit JVM.

Issue ID	Description
CR336813	Oracle JRockit might sometimes incorrectly optimize loops, assigning the same negative value to all elements of an array.
CR331774	If you are running on Linux or Solaris and press Ctrl-C to properly shut down your application, it will actually terminate immediately and you risk losing any runtime data that hasn't been saved to disk or a database. This happens because Oracle JRockit fails to register the SIGINT signal handler used for the shut down hooks. Workaround: If you encounter this problem, please download the updated version of the product, R27.3.1 from the BEA Downloads page. This issue does not apply to applications running on Windows.

2.15.2 Changes in the BEA JRockit JVM R27.3

The following table lists changes made in this version of the Oracle JRockit JVM.

Issue ID	Description
CR329414	Oracle JRockit could, in rare cases, garbage collect an instance declared as final on which the application has made the instance of operation on. This would result in a NullPointerException in the application.
CR327343	The documentation for jrcmd at: http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/ctrlbreakhdlr.html now includes information on the known limitations of jrcmd.
CR324513	The default preferred TLA size (-XXtlaSize) has changed, see the BEA JRockit Command-Line Reference Manual for details.
CR323910	JRockit could crash due to stack overflow while optimizing very large methods. This has now been fixed.
CR323086	JRockit gave unexpected errors with the -Xcheck:jni command-line option. JRockit would detect a false positive when using JNI to do a downcast array assignment (assign an array of subclass to Object as element to an array of Object arrays, i.e set [byte -> [[Object]).
CR322633	JRockit sometimes could not load very large jsp classes, which resulted in the error: java.lang.ClassFormatError: <class> : illegal attribute length (SourceDebugExtension:91802) This error was caused by that JRockit used a limit of 65536 on the SourceDebugExtension attribute. This limit has now been removed.
CR322146	The garbage collector mode -XgcPrio:pausetime now uses a fixed nursery of the same size as -Xgc:gencon, which is 10MB times the number of hardware threads.
CR321899	When using the parallel garbage collector and if an evacuation was aborted because the time limit was reached, the evacuation area size was doubled. This bug could cause unnecessary long pause times in the parallel garbage collector. This has now been fixed.
CR321325	The JVMTI function GetAllStackTraces previously returned JVMTI_ERROR_ILLEGAL_ARGUMENT if the max_frame_count parameter was 0 (zero). This has now been fixed.
CR319804	The JVMTI function GetObjectMonitorUsage could return JVMTI_ERROR_THREAD_NOT_ALIVE if the thread holding the object terminated during the call. To comply with the specification, this has now been changed to return JVMTI_ERROR_NONE and set info_ptr->owner to NULL instead.
CR319764	JRockit now handles names that are as long they are allowed by the ZIP standard. The previous limitation of maximum 512 bytes long entry names in zip files no longer exists.
CR319239	JRockit did not always find all free memory in the fourth GB of the virtual memory space. This bug manifested on 64 bit Linux platforms and could lead to OutOfMemoryErrors when using a maximum java heap size between approximately 3 and 4 GB. This has now been fixed
CR319234	The JMAPI getProcessAffinity/suggestProcessAffinity now works correctly when running on a Linux system with a GLIBC version older than 2.3.4.
CR317171	Pause time measurements between R27.3 and earlier JRockit releases (except for JRockit R27.2) are now comparable.

Issue ID	Description
CR317113	JRockit now reports the correct amount of physical RAM in 32 bit machines with PAE extension and more than 4GB of RAM installed.
CR316942	JRockit no longer hangs if you specify a nursery size (-Xns) that is less than 18 times the thread-local area size (-XXtlaSize).
CR315905	The Mercury Profiler tool has been omitted as of this release.
CR315761	It is now possible to use the EPollSelectorProvider in java.nio on Linux ia64. The EPollSelectorProvider is only used if the system property java.nio.channels.spi.SelectorProvider has been set to sun.nio.ch.EPollSelectorProvider.
CR314598	In JRockit R27.1 and R27.2, when trying to run some MBean servers, some classes could not be found, even though they were on the classpath. This problem has now been fixed.
CR314527	Previously, on rare occasions, external compaction caused very long pause times (if the heap was fragmented) when trying to move a large object from the highest heap parts. This has now been fixed.
CR314031	JRockit no longer calculates the wrong serialVersionUID for some classes backported from JDK 5 to JDK 1.4 where Enums is used. Enums are now correctly ignored (they are not part of the 1.4 specifications) when calculating the serialVersionUID for 1.4.
CR312360	The -Xmanagement flag has been updated. You start a local in-memory connector if start without any arguments. You start a remote connector if: Any of the options authenticate, ssl, port, or autodiscovery is set. Any of the above options is set through system or management properties, (com.sun.management.jmxremote.port, jrockit.management.port, etc.). The password file, pointed to by the management property com.sun.management.jmxremote.password.file (default jmxremote.password), exists. The remote connector requires that you have username and password defined in the above password file unless the option authenticate is set to false. The remote connector uses SSL by default, unless the option ssl is set to false.
CR312194	To see the version of the time zone data (tzdata) of a JRE you can now run: <jdk>\bin\tzinfo The output shows Java version, JRE version, and Time Zone data (tzdata) version.
CR311518	The default TLA size now grows more aggressively and it has been increased to 2k at a 16Mb heap up to 256k at a 2GB heap.
CR311336	This release offers a new heuristic for updating the nursery size (for the static garbage collector) during runtime.
CR311327	The value for TotalGarbageCollectionTime is now showed in milliseconds on Windows XP.
CR293617	Singlestepping with JDI on a single-CPU computer is now faster and easier to use.

Issue ID	Description
CR262438	JRockit no longer fails to detect whether HyperThreading (HT) has been enabled or not, which means that it will no longer start a non-optimal number of garbage collection threads.
CR206755	The initial heap size will now be at least twice the size of the nursery if -Xns (the nursery size) is set and -Xms (the initial heap size) is not, unless this leads to that the initial heap size becomes larger than the maximum heap size.

2.16 Changes in the Oracle JRockit JVM R27.2

The following table lists changes made in this version of the Oracle JRockit JVM.

Change Request ID	Description
8139785	On Microsoft Windows Server 2003 "R2", Oracle JRockit R27.1 fails to escape the quotation characters around the word R2 when writing the operating system name to a JRA recording. This breaks the XML-structure of the recording and causes problems when reading the recording. To make the recording readable, manually remove the quotation marks around R2 in the operating system name field of the recording file (a text search for the four characters "R" in the recording's XML-file should find it). This problem has been fixed in JRockit version R27.2 and later.
CR315538	Problems were occurring because the JRA was unable to handle class unloading. This situation has been corrected.
CR311708	Oracle JRockit no longer detects false positive Java deadlocks.
CR311186	A regression in BEA JRockit R27.1 caused -Xverbose output to be buffered and therefore delayed. The output is no longer delayed.
CR310238	The javaw launcher in JDK 6 on Windows supports class-path wildcards. This is a backport of a fix for the problem reported by Sun in http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6510337 .
CR309555	The JNI method ToReflectedMethod crashed if the class parameter was NULL. This has now been fixed.
CR308967	Sometimes BEA JRockit crashed when the method <code>java.util.concurrent.atomic.AtomicReferenceArray.compareAndSet</code> was called. This has now been fixed.
CR308312	The byte code verification in BEA JRockit has been relaxed in cases where JRockit's strict byte code verification would otherwise cause <code>ClassFormatErrors</code> to be thrown.
CR307903	If a thread was interrupted for garbage collection while copying an array, the garbage collection could result in long pauses. This has now been fixed.
CR307114	When reflection was used to set volatile static variables in JRockit on Windows, JRockit crashed. This has now been fixed.
CR306848	The <code>jstat</code> tool (in the <code>/bin</code> directory) used counters that triggered error messages about Unresolved Symbols. This has now been fixed and <code>jstat</code> no longer uses the obsolete counters.
CR306729	The heuristics used by <code>-Xgcprio:throughput</code> to set nursery size and select garbage collector strategy have been improved.
CR306048	The <code>referrer_index</code> argument to the <code>jvmtiObjectReferenceCallback</code> function was not always set to -1 when it should have been. This has now been fixed.

Change Request ID	Description
CR305091	The jrcmd utility on ia64 had problems with user names longer than 9 characters. This has now been fixed.
CR304733	BEA JRockit's method profiler timing counters are no longer available on Fujitsu's SPARC implementation SPARC64, since JRockit sometimes gave the wrong timing data (negative numbers) on that platform.
CR304335	The method getNurserySize() in the GarbageCollector class now works as documented, that is, it throws a NotAvailableException when the JVM is running a single-spaced garbage collector (without a nursery). Previously, the method returned 0 (zero). The method setNurserySize() now throws a NotAvailableException when the JVM is running a single-spaced garbage collector.
CR303790	The new command line option -XX:MaximumNurseryPercentage limits the maximum size of the nursery to a percentage of the free heap space available after the latest old collection. The default value is 95%.
CR302924	A ctrl-break handler can now be sent to stop a JRA recording even if JRA has not actually started recording yet (but is in a start up state). If JRA has just been started, then there may be a short delay before the recording is actually stopped. Previously, if you sent a ctrl-break handler to stop JRA before it had actually started recording, you would have generated the following error message: Error: No JRA recording running.
CR301964	Issues with thread names not being available in Linux environments are fixed.
CR299662	The parameters genpar and singlepar have been added to the -Xgc option. Using -Xgc with these parameters are equivalent to using the -XXsetgc option with the parameters genparpar and singleparpar.
CR299651	The option -XlargePages has been added. This option is intended to replace -XXlargePages but the old command-line option is retained for backward compatibility purposes.
CR298847	At all times, the following relations are now automatically maintained between minimum and preferred TLA size, large object limit, and minimum block size: -XXlargeObjectLimit <= -XXtlaSize:min <= -XXminBlockSize -XXtlaSize:min <= -XXtlaSize:preferred If you set two or more of the options, then you must make sure that the values you use fulfil these criteria. If you only set one of the options, the others will adapt if necessary so that the values are valid. It is recommended that you primarily set the TLA size parameters for memory management tuning purposes, while letting BEA JRockit automatically adjust the large object limit and minimum block size. By default, the large object limit will be set to whichever is the lower value of (1) the minimum TLA size and (2) the preferred TLA size divided by 2. The default minimum block size is 2k.
CR297814	The limit of capturing only the first 20 frames of allocation stack traces in the Memory Leak Detector has been removed.
CR278996	-Xverbose:referents (alternatively, -Xverbose:verboserefs) gives you printouts of all reference objects for each old generation garbage collection as well as the referents to which they point. In previous BEA JRockit versions, this could be achieved with the option -Djrockit.verboserefs.

2.17 Changes in the Oracle JRockit JVM R27.1

The following table lists changes made in this version of the Oracle JRockit JVM.

Change Request ID	Description
CR266204	Optimized method calls on null objects now throws <code>NullPointerException</code> as is the expected behavior.
CR264251	The JRockit installer on Windows previously only had an “Install Public JRE” option on 32-bit Windows. This option is now also available with the 64-bit JRockit versions for Windows (x86-64 and Itanium).
CR274190	The command-line option <code>-Xverbose:gcpause</code> has been improved, see <code>-Xverbose</code> in the Oracle JRockit Reference Manual at http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/refman/index.html for more information.
CR276950	JRockit R27.1 and later no longer supports Linux versions using the old <code>LinuxThreads</code> library, which includes Red Hat AS 2.1, SuSE ES 8.0, and other distributions based on the standard 2.4 kernel. See the Supported Configurations documentation for information on supported releases at http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/jrdocs/suppPlat/supp_plat.html .
CR277922	It is now possible to install both the 32-bit and the 64-bit BEA JRockit JDK/JRE on Windows x86_64, side by side.
CR280059	JRockit no longer generates faulty machine code for Java code that stores the <code>>this<</code> pointer into its own object. This caused Java variables to be set to zero in rare situations.
CR281323	Internal JVM threads in the “system” <code>ThreadGroup</code> did not have the correct value for <code>isDaemon()</code> and <code>getPriority()</code> . This has now been fixed.
CR281936, CR283237	Previously, if the memory was very fragmented when JRockit started, JRockit crashed during startup without creating any dump file. This problem is now fixed.
CR283236	The behavior of <code>java.lang.StackTraceElement</code> has been changed to conform to that of the Sun reference implementation. In previous releases the method signature was included in the resulting <code>String</code> , while the new version behaves as described in http://java.sun.com/j2se/1.5.0/docs/api/java/lang/StackTraceElement.html#toString() This also affects the behavior of <code>Throwable.printStackTrace()</code> , in that the result will not list any method signatures.
CR283454, CR283915	Previously long thread sleeps issued by <code>Thread.sleep(...)</code> and <code>Object.wait(...)</code> could end too early if the sleeps were longer than <code>0x3FFFFFFF</code> milliseconds (approximately 12.4 days). This problem has now been fixed.
CR284604	The JVMTI API version of the Java SE 5.0 version of JRockit R27.1 is 1.1. This can be seen with the <code>GetVersionNumber</code> API function. Some capabilities of JVMTI 1.1 are not available in the Java SE 5.0 version of JRockit.
CR286267	Previously an uncommon internal deadlock could occur. In the stack traces of a thread dump, the deadlock situation may have had calls that had to do with class loading.
CR286625	The BEA standalone installer filenames for Windows platforms have changed names. Now “windows” is used instead of “win” to name the target operating system in the platform part.

Change Request ID	Description
CR286926	Command line option verification has become stricter and you will receive more comprehensive error messages and warnings when using command line options that are incorrect. See the Oracle JRockit Reference Manual for details.
CR291898	JRockit could in rare circumstances crash when compiling methods calling <code>String.indexOf()</code> . This has been fixed.
CR291969	JRockit no longer crashes due to a previous bug in the JRockit code optimizer that was triggered by applications throwing a large amount of exceptions.
CR294608	By default, the output from <code>-Xverbose</code> now includes the logging level as well as the module. Use <code>-XverboseDecorations</code> to change the default settings.
CR296429	The JMAPI function <code>JVM.getProcessAffinity()</code> now works properly on Linux.
CR296668	A bug that caused JRockit to crash in the internal function <code>handlersMatchExceptForUnlockHandler</code> has been fixed.
CR297036	The Security Vulnerability in RSA Signature Verification issue, Sun Alert ID 102686, fixed in Sun JDK 1.4.2_13 has been back ported to JRockit 1.4.2_12 R27.1.0.
CR297037	The Security Vulnerability in RSA Signature Verification issue, Sun Alert ID 102686, fixed in Sun JDK 5.0 update 9, has been back ported to JRockit 5.0 update 8 R27.1.0.
CR297675	Now JRockit shows the correct value for <code>TotalGCTime</code> . The incorrect value could previously be observed through the JRockit Management API, and through the WLS Console.
CR298049	JRockit no longer crashes when compiling bytecode where normal control flow and exception control flow do not follow javac conventions. As far as BEA is aware, such bytecode is only produced by a small set of bytecode obfuscators.
CR298365	The command-line option <code>-Xmanagement</code> has changed behavior in this release.
CR301758	As part of the rewrite of JRCMD in R27, the <code><jre>/bin/jrcmd</code> launcher has been removed (the <code><jdk>/bin/jrcmd</code> still remains), i.e. the JRCMD tool is now only available if installing the full JRockit JDK, not only the JRockit JRE.
CR302559	The "Security Vulnerability in Serialization" issue, Sun Alert ID 102731, fixed in Sun JDK 1.4.2_13, has been back ported to JRockit 1.4.2_12 R27.1.

2.18 Known Issues

The following table lists issues that, in some circumstances, might affect the performance of the JRockit JDK.

Issue ID	Description
Untracked	<p>For Linux users only.</p> <p>The JRockit JVM is crashing due to a signal handling conflict.</p> <p>If you are using the JRockit JVM in conjunction with a native library that relies on OS signals you may experience crashes due to a signal handling conflict between Oracle JRockit and the native library.</p> <p>Workaround:</p> <p>Set the environment variable LD_PRELOAD as follows:</p> <pre>export LD_PRELOAD=\$JROCKIT_HOME/jre/lib/i386/libjsig.so</pre> <p>Oracle Engineering found this conflict using IBM's MQSeries native drivers, and it may be present in other libraries that rely on native code.</p> <p>For more information, see: http://java.sun.com/j2se/1.5.0/docs/guide/vm/signal-chaining.html</p>
9830596	<p>Taking a LAT recording while running with a byte code modifying agent (like Wily Introscope) could lead to ClassCircularityError error or NoClassDefFoundError.</p> <p>Workaround:</p> <p>Run Oracle JRockit with: <code>-Djrockit.agent_and_lat.enable=true</code></p> <p>This will disable agent preprocessing of latency classes as well as four classes that the latency event object depends on:</p> <pre>java/io/ByteArrayOutputStream java/io/DataOutputStream java/io/DataOutput java/io/IOException</pre>
9258993	<p>Using the command <code>set_filename</code> in a <code>ctrlbreak.act</code> file could cause the Oracle JRockit JVM to crash on some systems</p>
8889334	<p>Sometimes when trying to launch Oracle JRockit <code>jrcmd</code> on Windows, the command will return:</p> <pre>. <pid>: Not enough storage is available to process this command .</pre> <p>This happens when you try to launch <code>jrcmd</code> in a different Windows station from the one on which the target JVM is running. This can occur, for example, when trying to use <code>jrcmd</code> in a Terminal Services environment or if the JVM is running as a Windows service. <code>jrcmd</code> does not support communication between Windows stations. To avoid this situation, ensure that <code>jrcmd</code> is launched on the same Window station as the target JVM process.</p>

Issue ID	Description
8418795	<p>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:</p> <p>OVM 2.1.2</p> <p>OEL 4.7 ia32</p> <p>Patch required. The para-virtualized kernel for OEL needs to be 2.6.9-78.0.13.0.1.1.ELxenU or later.</p> <p>OEL 4.7 x64</p> <p>GA bits works fine</p> <p>OEL 5.3 ia32 and x64</p> <p>GA bits works fine</p> <p>JRockit supports both hardware and para-virtualized versions and both OEL 4 and OEL 5.</p> <p>This issue was also tracked as bug 8324323.</p>
8271382	<p><code>JVMFactory.getJVM().getMachine().getPhysicalMemory().getUsedMemory()</code> will return 0 if run on a 64-bit machine with a 32-bit JRockit instance.</p>
CR378758	<p>In rare cases, a faulty code optimization will make multiplying the results of two function calls with the double return type into a new double value generate an incorrect result.</p>
CR372377	<p>Sometimes, when debugging the throwing of an uncaught <code>java.lang.ClassCastException</code> with Eclipse, the debugged JVM might crash.</p>
CR371381	<p>The JRockit JVM is incompatible with Sun HotSpot when serializing <code>java.math.BigDecimal</code> objects over IIOP.</p>
CR364607	<p>Supplying a file name (including path) the length of which exceeds 256 characters to the <code>java.util.zip.ZipFile</code> constructor on Windows will fail and throw a <code>java.io.FileNotFoundException</code> even if the file exists.</p>
CR363637	<p>The JRockit JVM might occasionally crash in the <code>pkcs11_softtoken.so</code> native library when executing AES cryptography code. This is due to a bug in Solaris versions 10.4 and earlier. The bug has been fixed in Solaris 10.5.</p>
CR363487	<p>Due to an issue with 64-bit Windows Vista/2008, if you try to install the 64-bit JRockit JVM in the standard folder, you'd be told the installer cannot write there; however if you chose to install to another folder (one that the installer can write to), the files are copied and then the installer tries to write the uninstaller information and fails. The result is a broken installation without a working uninstaller.</p> <p>Workaround:</p> <p>Right-click on the file and select Run as administrator when installing the JRockit JDK.</p> <p>This issue has been fixed in Oracle JRockit R27.6.</p>

Issue ID	Description
CR361457	<p>Connecting the Memory Leak Detector to a running JRockit JVM (version R26.3 through R27.5) built for Linux IA-32 might cause the JVM to crash if the JVM process uses more than about 1020 file descriptors at the time. This might only happen if the file descriptor limit has been set higher than 1024 (typically by using the <code>ulimit</code> command).</p> <p>Workaround:</p> <p>Currently you can start the Memory Leak Detector Server at JVM startup, when few file descriptors are in use. To do this, add <code>-Djrockit.memleak.port=12345</code> early in the JVM command line.</p> <p>Now, using JRockit Mission Control, create a custom connection in the JRockit Browser with a Custom JMX service URL of <code>service:jmx:mlp://localhost:12345</code>. (Replace localhost and the port 12345 as needed). Using this connection, you can connect the Memory Leak Detector in JRockit Mission Control to this JVM once (without restarting the JVM).</p> <p>Note that using many file descriptors might be an indication of a resource leak in the Java application. Make sure to always close opened files and sockets. You should not rely on the Garbage Collection and the object finalization to free a non-Java resource such as a file descriptor.</p> <p>For troubleshooting information, see https://support.oracle.com.</p>
CR361032	<p>When debugging Java floating point applications on SSE2-enabled platforms (supported on Pentium 4 or newer and AMD Opteron, Athlon 64 from 2003 or later processors) using a 32-bit version of the JRockit JVM on Windows Vista x64, local floating point variable values might be bogus. This is due to a bug in Windows Vista x64. It has been fixed in Windows Vista x64 SP1.</p>
CR359954	<p>In JRockit Mission Control released with Oracle JRockit JVM R27.4, the value shown in Heap Usage Before for a garbage collection in a JRA recording is incorrect. The value shown is actually the value of Heap Usage After for the proceeding collection. This will be fixed in Oracle JRockit JVM R27.5.</p>
CR359328	<p>The ACL on the per-user <code>hspferdata_<user></code> directory might cause administration problems on some systems. One example of the problem is that the files under the directory can be removed by a user with Administrator privileges but the directory itself cannot.</p> <p>Workaround:</p> <p>Logged in as Administrator, modify the ACL with the <code>cacls</code> command before attempting to remove it.</p>
CR357402	<p>If a Java program uses too many (currently 2097151) active monitors (that is, by doing wait/notify or contended synchronization on too many objects) the JVM's internal monitor index can overflow. This is more likely to happen when using large heaps and few garbage collections occur. If this happens, the JVM will crash with an error saying "The number of active Object monitors has overflowed."</p>

Issue ID	Description
CR349882	<p>JRockit JDK installation and the use of applications on compositing window managers under Linux may fail with the following error message: xcb_xlib.c:50: xcb_xlib_unlock: Assertion 'c->xlib.lock' failed. This is due to a known problem in the native Sun JDK libraries; see Sun Bug 6532373 at http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6532373.</p> <p>Workaround:</p> <p>To work around this issue, please refer to Workarounds for CR349882</p> <p>This issue has been fixed in Oracle JRockit 5.0 R27.6.3 and Oracle JRockit 6 R27.6.3.</p>
CR348820	<p>Debugging a Java program when running the JRockit JVM with hardware performance counters on IA64 (-XXhpm) can result in exceptions. This is because the implementation of the hardware performance counters uses signals to communication with the JVM and the debugger agent implementation (JDWP) does not correctly handle this.</p> <p>Workaround:</p> <p>Disable hardware performance counters when debugging.</p>
CR341568	<p>The JRockit JDK exposes Mbeans under the bea.jrockit.management domain, accessible through the Management Console. These MBeans are proprietary and subject to change at any time. While you can access these MBeans directly using other JMX based tools, such direct access is not supported by Oracle.</p>
CR341293	<p>With Oracle JRockit R27.1 and later, JVMPI and JVMTI agents cannot be used at the same time. If you attempt to do this, only the transformations applied by the JVMPI agent are processed. The workaround is to use only one of these two interfaces, with a preference for JVMTI since JVMPI has been deprecated as of J2SE 5.0.</p>
CR340838	<p>When starting the JRockit JVM with a maximum heap size larger than the maximum address space (4 GB) on Linux ia32, the JVM might crash, instead of terminating nicely and generating an error message, as it should.</p>
CR340660	<p>Calling the methods isLoaded(), load() or force() in java.nio.MappedByteBuffer on an empty buffer throws an IOException instead of silently returning. This is know to affect the IntelliJ IDEA IDE.</p>
CR339469	<p>Copying event information from the Thread Latency Log table to the clipboard does not work properly. Only the header information will be copied. This issue will be fixed in the Mission Control included in JRockit JVM R27.5.0.</p>
CR338731	<p>Some events in the JRA latency recordings have their thread ID's set to 0. In particular, this applies to JVM Event Wait->Signalling thread, Java Synchronization->Last holder thread and Java Synchronization->Holder thread.</p>

Issue ID	Description
CR338678	<p>If you are running JRockit JDK 1.4.2, you might receive an incorrect error message when using the command-line option <code>-Xmanagement</code> with the parameter <code>class</code>, followed by additional parameters; for example:</p> <pre>java -Xmanagement:class=foo,ssl=false Hello</pre> <p>results in this error message:</p> <pre>Unknown parameter class Could not create the Java virtual machine</pre> <p>You cannot specify any parameters after the <code>class</code> parameter, so the correct error message should be:</p> <pre>Unknown parameter ssl Could not create the Java virtual machine</pre> <p>For more information, please refer to the <code>-Xmanagement</code> documentation.</p>
CR337697	<p>Compiling a program that uses the JRockit Management API (JMAPI) with <code>javac</code> from a Java SE 6 version of the JRockit JDK will give an error saying that “package <code>com.bea.jvm</code> does not exist”.</p> <p>Workaround:</p> <p>Do one of the following:</p> <p>Delete (or rename) the file <code><jrockit_home>\lib\ct.sym</code> and then recompile.</p> <p>Use <code>javac</code> from a 5.0 version of the JRockit JDK instead.</p>
CR337475	<p>In a JRA recording, the number of allocated TLA (Thread Local Areas) is recorded, as well as the preferred size of a TLA (in bytes). The JRA GUI will multiply these values to get the number of bytes allocated in TLAs during the entire recording; however, the size of the TLAs actually used can sometimes be a bit smaller than the reported size (the preferred size is only a preferred size; fragmentation can cause the TLAs to become smaller) and the value printed in the GUI can be overestimated.</p>
CR328975	<p>Latency data in a JRA recording will be erased from the disk if comments on the Notes tab in Mission Control are saved. Non-latency data will still be available, but the message “Warning! Error(s) when reading JRA-recording” will appear.</p> <p>Workaround:</p> <p>Don’t use the Notes tab in Mission Control when working with recordings that contain latency data.</p>
CR328964	<p>The <code>-XXcompactSetLimit</code> flag does not always limit the compaction set. In some circumstances compactations can exceed the given limit, typically in the initial compactations before the whole heap has been processed for external compaction.</p>
CR328729	<p>When starting a JRA-recording by using Mission Control, the recording might not start and the error message, “Could not delete file” will appear. This happens when the recording has the exact same filename as a previously-started recording.</p> <p>Workaround:</p> <p>In the JRA-recording wizard, give each recording a unique name or close Mission Control and restart it.</p>

Issue ID	Description
CR326746	<p>The <code>set_filename</code> handler will not update the output for the running command batch.</p> <p>Workaround:</p> <p>Issue a <code>set_filename</code> command.</p> <p>Issue the commands that you wish to send to the output.</p>
CR322908	<p>A known issue in Red Hat Enterprise Linux 5.0 on <code>x86_64</code> with the <code>dladdr()</code> call in <code>glibc</code> might cause irregular behavior or a crash when running graphical applications; see also Red Hat Errata RHBA-2007:0619-3 at:</p> <p>http://rhn.redhat.com/errata/RHBA-2007-0619.html</p> <p>The issue is fixed in Red Hat Enterprise Linux 5.1.</p>
CR317171	<p>A regression has been introduced in R27.2 in how pause times are measured. Pause times are visible in the JRockit Runtime Analyzer Tool and the verbose logs in the <code>Mark:Final:StopThreads</code> pause part, where they appear to be much longer than in previous JRockit JVM versions. This means that pause time measurements are not comparable between R27.2 and earlier JRockit JVM versions. The actual pause times have not changed.</p> <p>This issue has been fixed in BEA JRockit R27.3.</p>
CR316942	<p>If you specify a nursery size (<code>-Xns</code>) that is less than 18 times the thread-local area size (<code>-XXtlaSize</code>), the JRockit JVM will hang without printing an error.</p> <p>Workaround:</p> <p>Increase the specified nursery size (using <code>-Xns</code>) or lower the minimum TLA size.</p> <p>As of JRockit R27.1 the format for how to specify TLA size changed to specify both minimum and preferred TLA size. The old way (<code>-XXtlaSize :<size></code>) sets both minimum and preferred. Use <code>-XXtlaSize:preferred</code> to set the preferred TLA size, for example: <code>-XXtlaSize:preferred=64k</code>.</p> <p>This issue has been fixed in BEA JRockit R27.3.</p>
CR315939	<p>If you are using a 32-bit JVM and set the maximum heap size to a value above 4 GB, the JRockit JVM will allocate as large a heap as possible, but not exceeding 4 GB. This can result in the JVM throwing an internal out of memory error because the heap has taken all the address space.</p> <p>Workaround:</p> <p>When you encounter this situation, reduce the heap to a value less than 4 GB.</p>
CR315761	<p>It is not possible to use the <code>EPollSelectorProvider</code> in <code>java.nio</code> on Linux <code>ia64</code> with JRockit 5.0 R27.2. Note that the <code>EPollSelectorProvider</code> is only used if the system property <code>java.nio.channels.spi.SelectorProvider</code> has been set to <code>sun.nio.ch.EPollSelectorProvider</code>.</p> <p>This issue has been fixed in BEA JRockit R27.3.</p>
CR312235	<p>The code garbage collection is disabled during JRA recordings, so you might in special (rare) circumstances see an increased use of native memory during recordings. This can happen if you load a lot of classes when you do either a very long recording (several hours or even days) or shorter recordings back-to-back.</p> <p>Workaround:</p> <p>The workaround is to do several recordings, but leave some time (a few minutes should suffice) between JRA runs, so the JVM can run the code garbage collection.</p>

Issue ID	Description
CR311188	<p>On Solaris 10, a bug that makes getusage return bogus values in turn causes all printouts of page faults to present bogus values. You may get these printouts when you, for example, use <code>-Xverbose:memory</code>.</p> <p>The Solaris bug is identified as “6288308, Uninitialized struct causes getusage(3C) to return bogus data”. According to Sun, the first kernel patch with a fix for bug 6288308 is 118833-24. See http://sunsolve.sun.com/search/document.do?assetkey=urn:cds:docid:1-21-118833-24-1.</p>
CR310666	<p>A known issue in Sun’s J2SE 5.0 update 11 might cause BEA JRockit to dump when <code>PrinterJob.printDialog()</code> is called from a sub-thread. BEA has only identified the bug using Windows Vista on the IA32 architecture. This known issue will be fixed in 5.0 update 11 and included in JRockit R27.3.</p> <p>More information can be found in the Sun bug database: http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6358747.</p>
CR310230	<p>The new launcher <code>java-rmi.exe</code> that is included in JDK 6 on Windows does not work as expected. This is also reported by Sun in the original bug report at http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6512052.</p>
CR307903	<p>If a thread is interrupted for garbage collection while it is in the process of copying an array, then the garbage collection may result in very long pauses. If you get occasional long pause times, this may be the problem. Note that this issue has been fixed in BEA JRockit R27.2.</p>
CR307902 For Linux users only.	<p>If you explicitly request to use the Motif AWT instead of the default X11 AWT on Linux/IA64 and run a Linux version with a GLIBC version older than <code>glibc-2.3.4</code>, this operation might fail with an <code>UnsatisfiedLinkError</code> since the file <code><jre>/lib/ia64/motif21/libmawt.so</code> requires linkage to GLIBC <code>>= 2.3.4</code>.</p> <p>See the see the Supported Configurations document for supported Linux versions.</p>
CR305844	<p>If you are using the JRockit JDK on the Itanium version of Windows Server 2003 and the Java application unexpectedly hangs during heavy system load, then it might mean that the JVM has triggered an operating system bug. At the OS level, this is manifested as the JRockit JVM blocking on a call to the Windows <code>GetThreadContext</code> function. Microsoft has posted a knowledge base article that also includes instructions for obtaining a hotfix for the problem. It is available at http://support.microsoft.com/kb/947504.</p>
CR305091	<p>The <code>jr cmd</code> utility can have problems on ia64 if your username is longer than 9 characters. This is not a problem on other platforms. The issue has been fixed in BEA JRockit R27.2.</p>
CR304556	<p>If the JRockit JVM is started with a minimum TLA size (<code>-XXtlaSize:min=X</code>) that is larger than the maximum specified heap size (<code>-Xmx</code>), JRockit will deadlock at startup and never start running.</p> <p>Workaround:</p> <p>Set a minimum TLA size that is less than the maximum heap. Typically, the TLA size should be much smaller than the heap size.</p>

Issue ID	Description
CR304335	<p>In R27.1, the JMAPI method <code>getNurserySize()</code> in the <code>GarbageCollector</code> class doesn't work as documented. If the garbage collector that the JVM is running isn't using a nursery, the method should throw a <code>NotAvailableException</code>. Instead it returns 0. This has been fixed in R27.2.</p> <p>Workaround:</p> <p>If you depend on the exception being thrown, e.g. checking if you use a nursery or not, you can work around the problem by both catching the <code>NotAvailableException</code>, as well as checking the return value and see if it returns 0. If it throws an exception, or returns 0, a nursery is not being used.</p>
CR302141	<p>Files containing JRA recordings can be dragged and dropped into JRockit Mission Control. However, when dropping multiple files, some open file tabs may be labeled "JRA Editor" instead of the actual file name.</p> <p>Workaround:</p> <p>Select a tab for the file, then the file is actually read and the label is set to the correct file name.</p>
CR300393	<p>If the nursery is too small, JRockit may get stuck in triggering young collections, "back to back", without promoting anything. This can be seen in the <code>-Xverbose:memdbg</code> outputs 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).</p> <p>Workaround:</p> <p>Increase the nursery size. If nursery size has been set automatically by <code>-Xgcprio:throughput</code>, it can be overridden by manually setting <code>-Xns</code> to a higher value.</p>
CR300097	<p>A known issue in Red Flag 5.0 with the <code>wait()</code> call might cause irregular behavior or crashes for Red Flag customers using this OS version. This has been fixed in AsianUX 2.0 SP1 (of which Red Flag 5.0 is a part) and we strongly recommend that our users upgrade their OS to resolve this issue.</p>
CR295457	<p>If an application is configured with a heap size close to 4 GB and includes a lot of classes, an out of memory situation might occur if some JVM internal structures and the Java heap both try to share the low 4 GB memory space of the process. If this happens, try to increase or decrease the Java heap size by using either the <code>-Xmx</code> option or disable compressed references by using the <code>-XXcompressedRefs=0</code> option.</p>
CR284519	<p>To use the <code>-XXmme</code> option on Red Hat Enterprise Linux 4.0 you need to have Red Hat Enterprise Linux 4.0 QU4 or a later release installed; otherwise, you might encounter sporadic crashes.</p>
CR283776	<p>In 5.0 Update 7, Sun changed the <code>serialVersionUID</code> for the <code>javax.xml.namespace.QName()</code> class due to a historical defect. For the original bug report, see CR6267224 in Sun's bug database at: http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6267224</p> <p>To use the old compatibility value, set the following system property:</p> <pre>com.sun.xml.namespace.QName.useCompatibleSerialVersionUID=1.0</pre>

2.18.1 Workarounds for CR349882

This section instructions for working around the known issue described in CR349882.

Workaround 1

Apply the following patch, `patch_java.sh`, to the unpacked JDK or JRE.

```
$ cat patch_java.sh
#!/bin/sh
jh=$1
sed -i 's/XINERAMA/FAKEEXTN/g' $jh/lib/*/xawt/libmawt.so
sed -i 's/XINERAMA/FAKEEXTN/g' $jh/lib/*/motif21/libmawt.so
$ patch_java.sh <path-to-jrockit-jdk>/jre
$ patch_java.sh <path-to-jrockit-jre>
```

Workaround 2

In case the JRockit JDK installer `*.bin` fails immediately with the same error message as above, you can work around it by unpacking the installer program manually, applying the following patch to the internal JRE, then starting the installer yourself in GUI mode using the patched internal JRE.

```
$ cat patch_and_run_installer.sh
#!/bin/sh
jrinstaller=$1
rm -rf mytmp
unzip -d mytmp $jrinstaller
cd mytmp
GUI=`cat autorun.inf |grep GUI= | cut -d= -f2`
UNZIP=`cat autorun.inf |grep UNZIP= | cut -d= -f2`
UNZIPTO=`cat autorun.inf |grep UNZIPTO= | cut -d= -f2`
unzip -d $UNZIPTO $UNZIP
jh=$UNZIPTO
sed -i 's/XINERAMA/FAKEEXTN/g' $jh/lib/*/xawt/libmawt.so
sed -i 's/XINERAMA/FAKEEXTN/g' $jh/lib/*/motif21/libmawt.so
$GUI
$ patch_and_run_installer.sh <jrockit-installer>.bin
```

After successful installation using workaround 2 you might also have to apply workaround 1, using the path to the installed JRockit.

R26 Release Information

This document contains important details for BEA JRockit R26. It contains information on the following subjects:

- [Section 3.1, "New Features and Enhancements in JRockit R26.4"](#)
- [Section 3.2, "New Features and Enhancements in JRockit R26.3"](#)
- [Section 3.3, "New Features and Enhancements in JRockit R26.2"](#)
- [Section 3.4, "New Features and Enhancements in JRockit R26.1"](#)
- [Section 3.5, "New Features and Enhancements in JRockit R26.0"](#)
- [Section 3.6, "Most Recent Changes"](#)
- [Section 3.7, "Known Issues"](#)

3.1 New Features and Enhancements in JRockit R26.4

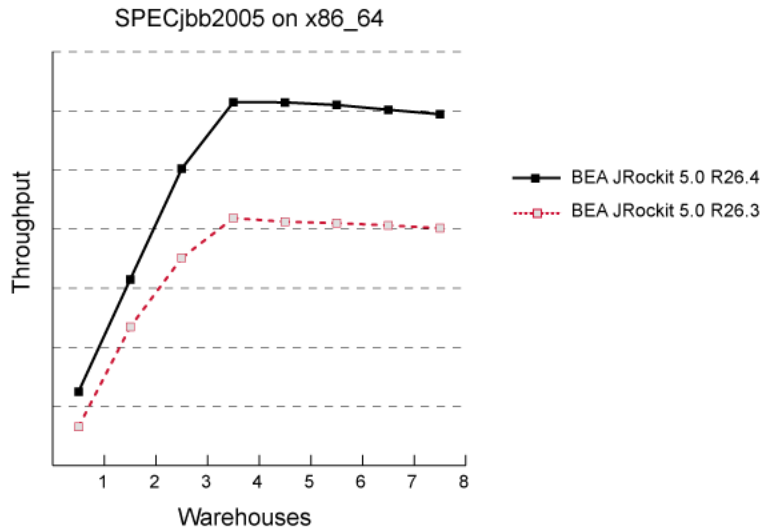
BEA JRockit R26.4 provides, apart from full support for J2SE 5.0 on all supported platforms, the following areas of improvement:

- [Performance Improvements on 64-bit Platforms](#)
- [Performance Improvements for Low Latency Applications](#)
- [Performance Improvements on SPARC](#)
- [Additional Tuning Possibilities](#)

3.1.1 Performance Improvements on 64-bit Platforms

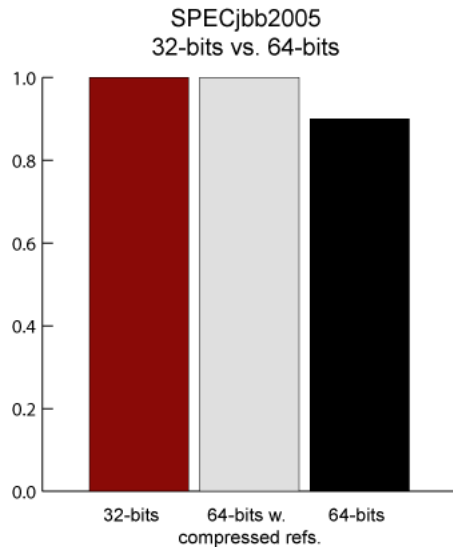
- General improvements on x86_64 platforms. See Figure 3-1 for a comparison between the BEA JRockit R26.3 and R26.4 releases.

Figure 3-1 Comparison between BEA JRockit R26.3 and R26.4



- Compressed references has greatly increased the performance on 64-bit platforms. See Figure 3-2 for a comparison of a 64-bit platform with and without compressed references compared to a 32-bit platform. For more information on compressed references, see `XXcompressedRefs` in the BEA JRockit Reference Manual.

Figure 3-2 Comparison between 64-bit platform with and without compressed references and a 32-bit platform



3.1.2 Performance Improvements for Low Latency Applications

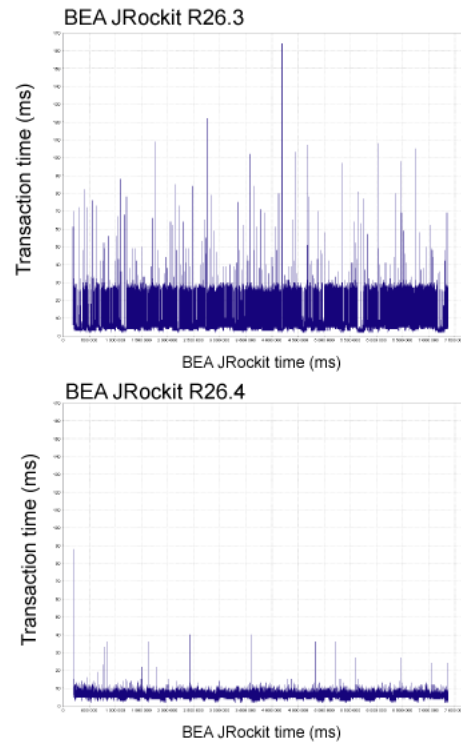
The performance of BEA JRockit has gone through major improvements for low latency applications by the following additions:

- Lock tuning on all supported platforms—Intel Itanium, Intel Xeon, AMD Opteron, Sun SPARC (including T1), and Fujitsu SPARC 64 V.

- Incremental handling of weak native handles have been moved to the concurrent phase, which lowers the pause times.

See Figure 3-3 for an illustration of how the performance have been improved with the above improvements.

Figure 3-3 Performance improvements compared between BEA JRockit R26.3 and R26.4



3.1.3 Performance Improvements on SPARC

SPARC performance running BEA WebLogic Server has been improved by approximately 15% compared to BEA JRockit R26.3.

3.1.4 Additional Tuning Possibilities

The following tuning options have been added in this release:

- XXgcThreads
- XXoptThreads
- XXcompressedRefs
- XXlazyUnlocking
- XXthroughputCompaction

For complete information on how to use all tuning options in JRockit, please see the JRockit Reference Manual.

3.2 New Features and Enhancements in JRockit R26.3

There have been performance enhancements to the Deterministic garbage collector:

- More efficient compaction.
- Improved handling of reference objects.

3.3 New Features and Enhancements in JRockit R26.2

This is the first BEA JRockit JDK 1.4.2 release containing full support for the Memory Leak tool and the JRA.

3.4 New Features and Enhancements in JRockit R26.1

BEA JRockit is now available for Solaris on Sparc. Please see BEA JRockit Supported Configurations for exact configurations.

Note: The Sparc version of BEA JRockit is a 64-bit JVM only.

3.5 New Features and Enhancements in JRockit R26.0

- Significant improvements in code generation speed, especially for large methods, e.g. jsp:s.
- Deterministic garbage collector (requires a separate license). For more information, see the Memory Management Guide.

3.6 Most Recent Changes

The following sections describe changes in JRockit:

- [Changes in the JRockit R26.4 Release](#)
- [Changes in the JRockit R26.3 Release](#)
- [Changes in the JRockit R26.2 Release](#)
- [Changes in the JRockit R26.0 Release](#)

3.6.1 Changes in the JRockit R26.4 Release

The following CRs have been corrected for the JRockit R26.4 release.

Change Request ID	Description
CR279188	<p>When JRockit 1.4.2_10 R26.3 calculates the SUID for a class to be serialized it includes the synthetic bits in the calculation, which generates a SUID that differs from the Sun 1.4.2_10 and BEA JRockit 1.4.2_08 R24. This problem causes serialization to fail.</p> <p>Synthetic attributes are included in a class when the class references a class pointer <code>ATestClass.class</code> or uses assertions.</p> <p>This means that RMI communication between 1.4.2_10 R26.3 and Sun 1.4.2_10 can fail. RMI communication between BEA JRockit 1.4.2_10 R26.3 and BEA JRockit 1.4.2_08 R24.5 can also fail in the same way. Serialized classes, stored in a database, might also fail to be loaded.</p> <p>This has been fixed in R26.4.</p>

Change Request ID	Description
CR258122 , CR258159, CR264680	The following new options have been added: <ul style="list-style-type: none"> ■ -XXgcTreads ■ -XXlazyUnlocking ■ -XXthroughputCompaction ■ -XXinternalCompactionMultiplier ■ -XXexternalCompactionMultiplier
CR258934	New arguments have been added to enable control of large pages (-XXlargePages) on Solaris.
CR267987	Using java.nio.channels.SelectionKey.OP_CONNECT will make BEA JRockit block forever. This has now been fixed.
CR268133	Previously java.lang.reflect.Method.getParameterAnnotations() returned the wrong result for methods that did not have annotations. This has now been fixed.
CR268439	Previously when calling JVMTI functions from a non-attached thread caused BEA JRockit to crash. This has now been fixed.
CR269375	For each old collection, the reason for a garbage collection is printed in the verbose:memdbg.
CR271551	Previously buffers that had been allocated through ByteBuffer.allocateDirect() would not be released, even when discarded by the application, which caused C heap memory leaks. This has now been fixed.
CR272364	The JVMTI function GetThreadInfo returned an error if passed NULL as the thread. It now treats NULL as the current thread in accordance with the specification.
CR274636	Previously BEA JRockit 1.4.2 R26.2 and R26.3 threw a java.lang.NullPointerException when passing a null argument to java.lang.String.getBytes(String charsetName). This was not an issue with BEA JRockit 1.4.2 R24. This has now been fixed.
CR275108	Previously, when using the lookfor parameter to the Ctrl-Break handler print_object_summary could crash BEA JRockit if certain kinds of references were found. This has now been fixed. In addition, the output includes the kind of each reference.
CR275725	The implementation of BEA JRockit's System.nanoTime() on Linux provided the time since BEA JRockit started. Now BEA JRockit uses gettimeofday instead, which makes it easier to use the time to compare different JVM instances.
CR276308	The verbose framework in BEA JRockit is now stricter and possible ambiguous shortcuts have been removed, for example, the option -Xverbose:compact no longer works, instead you need to use -Xverbose:compaction.
CR276311	BEA JRockit R26.3.0 could, under rare circumstances, crash when using -Xgc:gencon. All platforms were affected. This has now been fixed.
CR276460	BEA JRockit no longer fails on RHEL4 when using FileChannel.transferTo.
CR276987	In BEA JRockit R26 releases prior to R26.4, a breakpoint could be garbage collected if the garbage collection went off before the method was generated. This has now been fixed.

Change Request ID	Description
CR277702	The JraRecordingStarter.jar is no longer shipped with BEA JRockit. Use jrcmd instead to start a JRA recording.
CR278411	Previously, BEA JRockit incorrectly threw a BufferUnderflowException in java.nio.DirectByteBuffer.put() instead of BufferOverflowException when the buffer overflowed. This has now been fixed.
CR279712	Previously, the Ctrl-Break handler print_threads could result in a memory leak. This was due to handles that were not released during the printing of the thread status. This problem has been noted on Linux and Solaris platforms only and it has now been fixed.
CR280443	Previously when you expanded types in the Type Graph in the BEA JRockit Memory Leak Tool, BEA JRockit hung and became unresponsive while consuming CPU resources. This issue has now been fixed.
CR258248	You can now use the environment variable JROCKIT_DUMP_PATH to tell BEA JRockit to save crash dump information in a different location than the current working directory. The path specified must exist and be writable.

3.6.2 Changes in the JRockit R26.3 Release

The following CRs have been corrected for the BEA JRockit R26.3 release.

Change Request ID	Description
CR265225	The BEA JRockit JRE installer in Silent mode will now be correctly installed if the USER_INSTALL_DIR in the XML-file has been set.
CR264064	The default stack size for Solaris on Sparc has doubled to 512 kB.
CR263376	RHEL 4.0 QU1 on Itanium contains a critical kernel bug that was corrected in QU2. Therefore, BEA require that you run BEA JRockit with RHEL 4.0 QU2 (or later) on Itanium systems. For IA32 and x64, there are no known issues that require an update to QU2, and you can stay on QU1.
CR262962	Hyperthreading detection has changed slightly, which makes BEA JRockit better at detecting whether Hyperthreading is available and turned on or off. This has also caused the BEA JRockit property jrockit.cpu.ia32.ht to default to the value "os" (OS detection) instead of "hw" (Hardware detection).
CR262571	Previously the sizeof parameter to the Ctrl-Break handler print_object_summary did not work as expected. This has been fixed.
CR262540	Previously when running HP OpenView Java Diagnostics Profiling Agent could cause crashes with BEA JRockit R26.0.0. This issue has now been fixed.
CR262527	JRockit no longer fails to handle bytecode where the control flow enters an exception handler without a thrown exception.
CR262448	Multiple issues with -Xdebug in R24.5.0 have been fixed in this release.
CR260490	The new version of rtmon (librtm.so) is now compatible with Montecito systems, even if the kernel is not patched with the Montecito perfmon-2 patch. This means that there are no more failing on calls to RTMonRegisterThread() or RTMonStart(), which earlier caused BEA JRockit create a core dump.

Change Request ID	Description
CR260241	BEA JRockit sometimes crashed while stepping into an <code>ArrayIndexOutOfBoundsException</code> in the debugger on x86. This has now been fixed.
CR259231	Previously the JVMTI function <code>GetObjectsWithTags</code> was broken. This has now been fixed.
CR258894	When running BEA JRockit with <code>gcprio:pausetime</code> or <code>gcprio:deterministic</code> , it is now possible to set the pausetime target using JMAPI or JLMEXT. See the Javadocs for further information.
CR258395	BEA JRockit never called the JVMTI function <code>Agent_OnUnload</code> when the JVM shut down. This has now been fixed.
CR258200	In BEA JRockit R26 releases prior to R26.3, BEA JRockit running could livelock due to a thread priority issue on Windows platforms. This has now been fixed.
CR257905	(Linux only) When BEA JRockit was started from a process that blocks signal (such as Sun's HotSpot JVM) it was not possible to send <code>SIGQUIT</code> to or use the <code>jr cmd</code> tool against BEA JRockit. This has now been fixed by unblocking signals that BEA JRockit listens to.
CR257799	JRockit now allows more than one JVMTI agent to get the <code>can_tag_objects</code> capability.
CR257687	In BEA JRockit R26 releases prior to R26.3, BEA JRockit could crash due to a bug in the implementation of <code>Class.getMethod()</code> . The "this" reference was not treated properly by the garbage collector if threads were stopped in bad locations. This has now been fixed.
CR257039	The new option <code>-XXdisableGCHeuristics</code> disables the dynamic garbage collector selection heuristics when <code>-Xgcprio</code> is used. The JMAPI for changing garbage collectors in runtime can still be used. If you disable the dynamic garbage collector selection heuristics, it will affect the behavior of the dynamic garbage collector and may lead to lower throughput or longer garbage collection pauses.
CR256867	The initialization of the initial (main) thread on Linux now respects the <code>-Xss:<size></code> option, and commits an area corresponding to the largest of this and the current system stack <code>rlimit</code> (see <code>man rlimit</code>).
CR255959	A fix for Sun bug 5103041 has been added.
CR255271	The compaction for the deterministic garbage collector has been improved. The following options have been added: <ul style="list-style-type: none"> <code>-XXinitialPointerVectorSize:<nn></code> <code>-XXmaxPooledPointerVectorSize:<nn></code> <code>-XXpointerMatrixLinearSeekDistance:<nn></code> The option <code>-XXcompactSetLimit</code> now also works with <code>-Xgcprio:deterministic</code> .
CR254297	A race condition existed in BEA JRockit when the Memory Leak Server was stopped immediately after a client had connected, which caused sockets to be left in <code>CLOSE_WAIT</code> state. On Linux, this could eventually cause the BEA JRockit process to run out of file descriptors. This problem has now been fixed.
CR252610	The problem regarding the handling of TrueType fonts for certain font files has now been fixed.

Change Request ID	Description
CR251838	Previously a problem regarding the handling of TrueType fonts for certain font files caused a call to <code>java.awt.Font.getXXX()</code> methods that resulted in an <code>IllegalArgumentException</code> being thrown. This issue has now been fixed. (See CR252610 for more information).
CR248132	Explicit font support for Asianux, part of Red Flag Linux, has been added for BEA JRockit 1.4.2, keeping the support for Red Flag 4.1 font patch CR200703 for <code>LANG=zh_CN.GB2312</code> locale. The font support is based on the Red Hat Linux font support.
CR247393	Linux and Solaris man pages have been removed.
CR247026	<p>In earlier BEA JRockit R26 versions, on Windows operating systems, BEA JRockit could sometimes expose a problem in the OS related to multimedia timers that caused the system time to be adjusted backwards.</p> <p>This could cause the system time to jump back by about 1 minute. If this happened, you could turn off the use of multimedia timers with <code>-Djrockit.periodictask.usemmtimers=false</code>.</p> <p>This problem has now been fixed.</p>
CR189181	Explicit font support for Asianux, part of Red Flag Linux, has been added for BEA JRockit 5.0. The font support is based on the Red Hat Linux font support.
CR179421	BEA JRockit no longer misses to report some contended monitors to JVMPI.

3.6.3 Changes in the JRockit R26.2 Release

The following CRs have been corrected for the BEA JRockit R26.2 release.

Change Request ID	Description
CR256719	The JVM experiences a slowdown with large number of threads doing reflective invocation of the same method concurrently.
CR239984	In earlier releases of the BEA JRockit JDK 1.4.2, it was necessary to specify <code>-Xmanagement:class=com.JRockit.management.rmp.RmpSocketListener</code> on the command line to start up the Management Server. This can now be done by simply specifying <code>-Xmanagement</code> or by setting the management server port with <code>jrockit.managementserver.port</code> .
CR249272	The property <code>jrockit.managementserver.usejmx</code> has been added for BEA JRockit JDK 1.5. Setting this property to <code>false</code> will make BEA JRockit use the RMP-protocol instead of the default management protocol (JMX) on a BEA JRockit JDK 1.5.
CR250218	When trying to print heap references, non-fatal JVMPI error messages were displayed. This has now been fixed.
CR250712	A race could cause BEA JRockit to crash during a JRA recording if a thread completed at the wrong moment. This race has been fixed.
CR252315	The compaction heuristics now ignore exceptional compactions when adjusting the compact ratio and pointerset limit.
CR252348	The option <code>-Xverbose:cpuinfo</code> is now available on IA64.
CR252567	The default stack size on X64 platforms has doubled from previous releases.

Change Request ID	Description
CR253588	The amount of free space is now calculated correctly when BEA JRockit calculates the maximum allowed nursery size during automatic nursery resizing.
CR253952	Several JVMPI problems have been fixed.
CR254354	For some garbage collectors, the minimum block size value, set by <code>-XXminBlockSize</code> , was also (incorrectly) used as thread-local area size. With this fix, increasing the <code>-XXminBlockSize</code> value will no longer affect the thread-local area size. If you have been using <code>-XXminBlockSize</code> to adjust the thread-local area size, you now must also set <code>-XXlargeObjectLimit</code> and <code>-XXtlaSize</code> to the same value as you set <code>-XXminBlockSize</code> , as described in the BEA JRockit Reference Manual.
CR257184, CR257379	The option <code>-Xpausetarget</code> didn't always work when running <code>-Xgcprio:deterministic</code> . This has now been fixed.
CR257540	The JNI function <code>GetDirectBufferAddress</code> has now been changed to work with all direct <code>java.nio.Buffer</code> s. Previously it only worked for direct <code>java.nio.ByteBuffer</code> s.
CR257840	To make it easier to diagnose JRockit crashes, the option <code>-XXdumpfullstate</code> has been made default. This means that if BEA JRockit crashes a lot more information is saved to disk than was the case previously. To get the old behavior use <code>-XXdumpsize:normal</code> .
CR258002	Loading extra data from zip-file entries now work.
N/A	Java Web Start and the Java Plug-in were included in the previous 1.4.2 BEA JRockit version, i.e. BEA JRockit 1.4.2_08 R24.5.0. These features are dropped for BEA JRockit R26 on 1.4.2.

3.6.4 Changes in the JRockit R26.0 Release

The following CRs have been corrected for the BEA JRockit R26.0 release.

Change Request ID	Description
CR211951	In previous versions of BEA JRockit, the JVM Process Load was capped to 100/number of CPUs on multi CPU Windows machines. This has now been fixed.
CR213687, CR213685	The non-supported option <code>-XXprintStackOverflow</code> has been added. This option produces a full stackdump when the <code>StackOverflowError</code> is thrown.
CR218035, CR230226	In the previous release, BEA JRockit was known to hang or crash on 2.6 kernels on Itanium, due to a bug in the Linux 2.6.11 (and previous) kernel. The bug has now been fixed in kernel 2.6.12 by the Linux vendors. To run this release of BEA JRockit, you need to use SLES 9 SP2 or RHEL4 U1 (or later).
CR219610	The default freelist cache size is 10% of the current heap size (with a minimum of 3 MB).
CR225145	Changes to <code>java.vendor*</code> system properties. The correct values are: <code>java.vendor = "BEA Systems, Inc."</code> <code>java.vendor.url = "http://www.bea.com/"</code> <code>java.vendor.url.bug = "http://support.bea.com"</code>

Change Request ID	Description
CR226460	The experimental code cache feature has been removed due to stability issues.
CR228592	Using the Memory Leak Tool could in some instances make JRockit freeze or crash. This has now been fixed.
CR228822	<p>When running BEA JRockit on a single CPU machine, the code optimizer was in some cases too intrusive (true for BEA JRockit 5.0 R25.2.0). The problem was sometimes noticed at the first start of the WebLogic console. This problem has now been fixed.</p> <p>In the previous release of BEA JRockit, this problem was worked around by setting the flag <code>-Djrockit.codegen.optpriority=1</code>; if you are using this flag, remove it when updating to this release.</p>
CR229981	Improved behavior of internal locks that can lead to better performance during heavy loads.
CR230236	The <code>-Xmanagement</code> option resulted in an overhead even though the BEA JRockit Management Console was not connected. This has now been fixed.
CR232847	Improved floating point performance.
CR235100	<p>The <code>java.vm.version</code> for the previous release was: dra-45238-20050523-2021-win-ia32</p> <p>The <code>java.vm.version</code> for this release is: R26.0.0-188-52875-1.5.0_04-20051110-0917-win-ia32</p> <p>The <code>java.vm.info</code> for the previous release was: R25.2.0-28</p> <p>The <code>java.vm.info</code> for this release is: <empty></p>
CR235101	Previously, BEA JRockit calculated <code>MemoryMXBean.getNonHeapMemoryUsage().used</code> as the process virtual bytes minus the heap size. Now <code>MemoryMXBean.getNonHeapMemoryUsage().used</code> is calculated as the process rw memory minus the heap size.
CR235105	The heap occupancy trigger heuristics have been corrected.
CR235107, CR236922	<p>When running JRockit with a concurrent garbage collector, the garbage collector starts before the heap is completely full to be able to finish the garbage collection before running out of heap memory.</p> <p>It tries to determine when a garbage collection needs to be triggered through heuristics, but in certain situations it might be beneficial to set this trigger by hand and to a fixed value. Use the following argument <code>-XXgcTrigger=<int></code>, where <code>int</code> is an integer that takes values between 0 and 100. The value specifies the amount of free heap, measured in percentage, that should be available for the argument to trigger.</p>
CR235682	Previously selecting a generational, concurrent mark, concurrent sweep strategy resulted in a non generational, parallel mark, parallel sweep strategy being chosen. This has now been fixed.
CR236723	A warning appears at start-up if you are running with a “suspicious” thread system.
CR237093	<p>The time for the reference update phase was measured incorrectly when running the parallel garbage collector. This made the statistics that the (compaction) heuristics are based on incorrect.</p> <p>The pause target for compaction increased too fast when running static garbage collections. In this release, it can be increased with at most 50% for each garbage collection. The initial value has been set to 100 ms (this may be further tuned).</p>

Change Request ID	Description
CR238220	CPU load and CPU description is now returned as CompositeData in JRockitConsoleMXBean.
CR239499	The Memory Leak Detector did not work properly when the management server was started by the Ctrl-Break handler (using ctrlhandler.act) as opposed to using the -Xmanagement startup option. This has now been fixed and the Memory Leak Detector works as expected, regardless of how the management server has been started.
CR239968	In previous versions, the maximum stack trace depth value, in JRA recordings, was always 16 frames. In this version it is possible to set this value by adding the "tracedepth" option to -XXjra and the "jrarecording" ctrlbreak handler. The default value is still 16 frames.
CR240355	In this release it is possible to change verbosity for the "memory," "memdbg," and "gcpause" subsystems using the ctrlbreak handler.
CR240359	The syntax for the "verbosity" ctrlbreak handler has changed. The previous argument "args" has been changed to "set." Run the ctrlbreak handler "help verbosity" for more details.
CR240510	The information from jrockit.verboserefs has been improved and now includes more details regarding garbage collection. Support for verbose information in ctrlbreakhandler/jrcmd has been added.
CR241377	The default stack size on Solaris/Sparc is 256k.
CR241546	This release of BEA JRockit does not ship with Java Web Start or Java Plug-in. Some earlier releases did and the installers and uninstallers of those versions do not behave properly when this release is installed. To avoid problems with the installation, do one of the following before installing: 1. Uninstall all earlier BEA JRockit JRE releases before installing this release. Java Web Start and Java Plug-in will not be available after this process. 2. Install all earlier BEA JRockit JRE releases that are needed before installing this release. Java Web Start and Java Plug-in will be available if included in any of the earlier releases. Note: Do not install or uninstall an earlier release of BEA JRockit JRE while this release is installed. Doing so may corrupt the state of Java Web Start and Java Plug-in.

Change Request ID	Description
CR241638	<p>The following compaction tuning flags have been added:</p> <p><code>-XXinternalCompactRatio</code></p> <p>Sets the number of heap parts to compact during internal compaction. Default is dynamic or 8 when running <code>-Xgcprio:throughput</code>.</p> <p><code>-XXexternalCompactRatio</code></p> <p>Sets the number of heap parts to compact during external compaction (aka "evacuation"). Default is dynamic or 8 when running <code>-Xgcprio:throughput</code>.</p> <p><code>-XXheapParts</code></p> <p>Sets the number of heap parts. Default is 128.</p> <p>Furthermore, the system property <code>jrocket.gc.usematrix</code> has been turned into an <code>-XX</code> option.</p> <p><code>-XXusePointerMatrix</code></p> <p>Indicates that the pointer matrix should be used instead of the <code>pointerSet</code>. The pointer matrix is default when running <code>-Xgcprio:deterministic</code> or <code>-Xgcprio:pausetime</code>.</p>
CR241665	In the management API, the functions <code>getMAC</code> and <code>getMTU</code> are supported on Windows. On Unix systems these functions return an empty string or zero.
CR242307	To get fixes for potential security vulnerabilities, this release has upgraded <code>zlib</code> from <code>zlib-1.2.1</code> to <code>zlib-1.2.3</code> .
CR242944	The command <code>jrocket.oomdiagnostics.filename</code> specifies where to write out of memory diagnostics (if this is enabled through <code>jrocket.oomdiagnostics</code>). If diagnostics are enabled and no file is specified, the output ends up where the <code>-Xverbose</code> information ends up (typically <code>stderr</code>).
CR244403 , CR238634	Traversing superclasses of interfaces in <code>find_method</code> have been removed. These classes returned methods that were not declared in the interface or its super interfaces, for example, <code>Object.*</code> . The supermost class of an interface is always <code>Object</code> .
CR245707	JVMDI is not supported in the BEA JRockit R26.0 release (nor in the 25.0 or 1.4.2 builds); however, JDWP and JDI are supported. This means that remote debugging tools will still work as in previous releases.
CR245732	Previously, retrieving JMAPI stack traces could deadlock in certain cases for traces that included overridden <code>hashCode</code> methods that had been taking locks. This has now been fixed.
CR255294	Previously, when calling <code>java.io.File.getCanonicalFile()</code> on a path where <code>java.io.FilePermission</code> was not granted, the call did not fail as expected. This has now been fixed and the appropriate exception is thrown.

3.7 Known Issues

The following issues are known in BEA JRockit R26:

Issue ID	Description
Untracked	<p>For Linux users only.</p> <p>BEA JRockit is crashing due to a signal handling conflict.</p> <p>If you are using BEA JRockit in conjunction with a native library that relies on OS signals you may experience crashes due to a signal handling conflict between BEA JRockit and the native library.</p> <p>Workaround:</p> <p>Set the environment variable LD_PRELOAD as follows:</p> <pre>export LD_PRELOAD=\$JROCKIT_HOME/jre/lib/i386/libjsig.so</pre> <p>BEA Engineering found this conflict using IBM's MQSeries native drivers, and it may be present in other libraries that rely on native code.</p> <p>For more information, see:</p> <p>http://java.sun.com/j2se/1.5.0/docs/guide/vm/signal-chaining.html</p>
8165386	<p>Documentation Erratum</p> <p>Documentation at:</p> <p>http://fmwdocs.us.oracle.com/doclibs/fmw/E15289_01/doc.40/e15061/ctrlbreakhdlr.htm#i1001839</p> <p>suggests that less privileged user cannot list the running JRockit process, but if they know the process ID (PID), they can send commands to the process using jrcmd <pid> <command>. This could lead customers to think that jrcmd will work as long as the R26.4 jrcmd user is logged in as the less privileged user. This is incorrect.</p>
8165338	<p>Using the Post-Bind option might cause stack overflow in JRockit.</p> <p>Workaround:</p> <p>Use the JRockit option -Xss: in the WebLogic Server startup script to increase the default stack size if the Post-Bind option is being used in WLS. We cannot provide recommendations for what to set this to (default stack size on Linux_ia32 is 128kb and on Linux_x86_64 is 256kb); you should find a stack size that suits your application (that is, WLS starts up without crashing).</p>

Issue ID	Description
CR361457	<p>Connecting the Memory Leak Detector to a running JRockit JVM (version R26.3 through R27.5) built for Linux IA-32 might cause the JVM to crash if the JVM process uses more than about 1020 file descriptors at the time. This might only happen if the file descriptor limit has been set higher than 1024 (typically by using the <code>ulimit</code> command).</p> <p>Workaround:</p> <p>Currently you can start the Memory Leak Detector Server at JVM startup, when few file descriptors are in use. To do this, add <code>-Djrockit.memleak.port=12345</code> early in the JVM command line.</p> <p>Now, using JRockit Mission Control, create a custom connection in the JRockit Browser with a Custom JMX service URL of <code>service:jmx:mlp://localhost:12345</code>. (Replace localhost and the port 12345 as needed). Using this connection, you can connect the Memory Leak Detector in JRockit Mission Control to this JVM once (without restarting the JVM).</p> <p>Note that using many file descriptors might be an indication of a resource leak in the Java application. Make sure to always close opened files and sockets. You should not rely on the Garbage Collection and the object finalization to free a non-Java resource such as a file descriptor.</p> <p>For troubleshooting information, see https://support.oracle.com.</p>
CR311515	<p>The new explicit font support for Asianux, part of Red Flag Linux, available since BEA JRockit R26.3.0 depends on the existence and contents of the file <code>/etc/asianux-release</code> to correctly identify a Asianux compatible Linux distribution. If this file is not present the JRockit JDK or JRE will revert to look for <code>/etc/redhat-release</code> and instead identify a Red Hat Linux compatible distribution with Red Hat compatible font support.</p> <p>The font support for Asianux is based on the font support for Red Hat Linux, with an additional patch for Chinese <code>LANG=zh_CN.GB2312</code> locale for Red Flag Advanced Server 4.1 (and higher) in JRockit 1.4.2. Thus, if you are using Chinese <code>zh_CN</code> locale with JRockit 1.4.2 R26.3.0 and later on Red Flag Advanced Server 4.1 (and higher), that does not contain the file <code>/etc/asianux-release</code>, then BEA JRockit will load the Red Hat-specific <code>font.properties.zh_CN.Redhat</code> file instead of the expected Asianux-specific <code>font.properties.zh_CN.Asianux</code> file. This might cause the JVM to fail and result in unexpected behavior when it tries to load specific fonts with incorrect paths.</p> <p>On Red Flag Advanced Server 4.1 and Red Flag Advanced Server 4.1 (SP1) the file <code>/etc/asianux-release</code> file might not be installed by default and if that is the case you can apply either one of the following workarounds.</p> <ul style="list-style-type: none"> - Install the optional RPM package <code>asianux-release</code>, if available. This is the preferred solution. On Red Flag Advanced Server 4.1 (SP1) the optional package <code>asianux-release</code> may conflict with the possibly already installed optional package <code>redflag-release</code>, you may choose to either uninstall the <code>redflag-release</code> package before installing the <code>asianux-release</code> package, or to force install the <code>asianux-release</code> package in parallel with the <code>redflag-release</code> package. - Manually replace the contents of <code><jdk>/jre/lib/font.properties.zh_CN.Redhat</code> with the contents of <code><jdk>/jre/lib/font.properties.zh_CN.Asianux</code>.
CR307903	<p>If a thread is interrupted for garbage collection while it is in the process of copying an array, then the garbage collection may result in very long pauses. If you get occasional long pause times, this may be the problem. Note that this issue has been fixed in BEA JRockit R27.2.</p>

Issue ID	Description
CR307902 For Linux users only.	<p>If you explicitly request to use the Motif AWT instead of the default X11 AWT on Linux/IA64 and run a Linux version with a GLIBC version older than glibc-2.3.4, this operation might fail with an <code>UnsatisfiedLinkError</code> since the file <code><jre>/lib/ia64/motif21/libmawt.so</code> requires linkage to GLIBC $\geq 2.3.4$.</p> <p>See the see the Supported Configurations document for supported Linux versions.</p>
CR300097	<p>There is a known bug in RedFlag 5.0 with the <code>wait()</code> call that might cause undetermined behavior for RedFlag customers using this OS version</p>
CR286338	<p>In BEA JRockit 26.4, JRA Recordings always report that the heap usage is zero after a garbage collection with a single-spaced garbage collector. The <code>-Xverbose:memory</code> and <code>-Xverbose:memdbg</code> printouts report the correct value for the heap usage after garbage collection.</p>
CR284602	<p>The <code>jrcmd</code> tool might fail to find all BEA JRockit processes on a machine if there are any Java processes by other JVM vendors running on the same machine. Contact BEA JRockit Support for a patch that solves this problem.</p>
CR283915	<p>Long thread sleeps issued by <code>Thread.sleep(...)</code> and <code>Object.wait(...)</code> can end too early if the sleeps are longer than <code>0x3FFFFFFF</code> milliseconds (approximately 12.4 days).</p> <p>All platforms are affected.</p>
CR283787	<p>Upgrading from BEA WebLogic Platform 8.1 SP5 to 8.1 SP6 and running it on BEA JRockit R26 SP3 can result in a performance regression of up to 10%.</p> <p>Workaround:</p> <p>To avoid this regression, you can improve the performance of memory intensive applications by setting the command-line options <code>-XXtlaSize:<default 2kB></code> and <code>-XXlargeObjectLimit:<default 2kB></code>.</p>
CR280443	<p>When you expand types in the Type Graph in the BEA JRockit Memory Leak Tool, BEA JRockit can hang and become unresponsive while consuming CPU resources. This issue was introduced as a regression in BEA JRockit R26.2.</p> <p>This has been fixed in R26.4.</p>
CR279998	<p>Objects that are allocated with reflection, for example, with <code>java.lang.Class.newInstance()</code>, do not show up in the allocation stacktraces in the Memory Leak Detector Tool.</p>
CR279584	<p>The synchronization code in <code>java.util.Random.next()</code> has been optimized for the case where there is no contention, i.e. the <code>java.util.Random</code> object is used by only one thread. A drawback of this optimization is worse performance if the object is used heavily in several concurrent threads. This typically happens if the convenience method <code>java.lang.Math.random()</code> is used.</p> <p>To avoid this, create a new <code>java.util.Random</code> object instead of calling <code>java.lang.Math.random()</code>.</p>

Issue ID	Description
CR278796	<p data-bbox="610 226 1375 281">Null-pointer exception bypasses first catch block and is caught in the next.</p> <p data-bbox="610 296 1375 373">A catch block immediately following “return x.y;” where the variable x points to null, will not catch the null-pointer exception. The exception will be caught in the next catch block.</p> <pre data-bbox="610 388 1375 940"> public class Test { int y = 0; public static int foo() { try { Test x = null; return x.y; } catch (Exception e) { System.out.println("It works!"); } return 0; } public static void main(String[] args) { try { foo(); } catch (Exception ex) { System.out.println("Failure!"); } } } </pre>
CR276311	<p data-bbox="610 955 1375 1010">BEA JRockit R26.3.0 can, under rare circumstances, crash when using -Xgc:gencon. All platforms are affected.</p> <p data-bbox="610 1024 1375 1094">Workaround: Use another garbage collector or upgrade to BEA JRockit R26.4.0.</p>
CR275524	<p data-bbox="610 1108 1375 1186">On some Linux versions, the library functions exit() and _exit() do not always terminate the calling process. This causes BEA JRockit to hang during shut down on SLES 8.</p> <p data-bbox="610 1201 1375 1268">Workaround: Use a later SLES version.</p>
CR274636	<p data-bbox="610 1283 1375 1442">BEA JRockit 1.4.2 R26.2 and R26.3 throw a java.lang.NullPointerException when passing a null argument to java.lang.String.getBytes(String charsetName), while BEA JRockit 1.4.2 R24 did not. The behavior is unspecified but the change might cause problems when running TIBCO with XML messaging on BEA JRockit 1.4.2 R26.2 or R26.3.</p> <p data-bbox="610 1457 1375 1484">This has been fixed in R26.4.</p>
CR272699	<p data-bbox="610 1499 1375 1554">Occasionally, when using a JVMTI Java debugger, breakpoints are not hit. This issue can arise with any R26 version of the product.</p> <p data-bbox="610 1568 1375 1701">Workaround: Start BEA JRockit with the option -XXnoCodeGC. -XXnoCodeGC is only intended for troubleshooting and is not recommended nor supported for production use.</p>
CR271551	<p data-bbox="610 1715 1375 1793">Buffers that have been allocated through ByteBuffer.allocateDirect() would not be released, even when discarded by the application, which causes C heap memory leaks.</p> <p data-bbox="610 1808 1375 1837">This has been fixed in R26.4.</p>

Issue ID	Description
CR269115	<p>BEA JRockit crashes when optimizing method. This issue can be identified by the following stack trace:</p> <pre> at renameVar+36()@... at irCompactVars+240()@... at ssaConvertTo+2356()@... ... </pre> <p>This crash has only been noted on the Solaris on Sparc platform but might be a problem on other platforms as well.</p> <p>Workaround:</p> <p>Use optfile and remove the particular method that causes the crash.</p>
CR268746	<p>On older Linux distributions that run LinuxThreads instead of NPTL, BEA JRockit can sometimes hang when it is shutting down.</p>
CR268439	<p>Calling JVMTI functions from a non-attached thread will cause BEA JRockit to crash.</p> <p>This has been fixed in R26.4.</p>
CR268423	<p>BEA JRockit releases previous to R26.3.0 do not contain the fix for the Australian Daylight Savings Time change for 2006. Please contact BEA Support for a patch</p>
CR267987	<p>Using <code>java.nio.channels.SelectionKey.OP_CONNECT</code> will make BEA JRockit block forever.</p> <p>This has been fixed in BEA JRockit R26.4.</p>
CR266871	<p>BEA JRockit R26.0.0 on Linux IA32 can experience problems setting up memory for object allocation. It will manifest itself through this printout (and then exit BEA JRockit):</p> <pre> [[JRockit] ERROR: Fatal error in JRockit during memory setup phase. Try to reduce the heap size using -Xmx:<size>m, i.e. "-Xmx:16m". Could not create the Java virtual machine. </pre> <p>Workaround:</p> <p>Try different -Xmx values to find a heap size that is setup correct.</p> <p>This known issue is valid for R26.0.0. The problem is fixed in releases R26.1.0 and later.</p>
CR266870	<p>IA64 RedFlag 4.1 creates broken core files when programs crash. This makes it impossible for BEA JRockit engineers to resolve customer issues on RF41/IA64.</p>
CR266667	<p>Slow startup because of a hang in <code>java.net.PlainSocketImpl.initProto()</code>, which typically is called when creating the first <code>Socket</code> or <code>ServerSocket</code>.</p> <p>In BEA JRockit 5.0 R26 the network stack is configured so that IPv6 is used in preference to IPv4 when it is present.</p> <p>During initialization of the network stack, the network code connects a socket to its own loopback interface to set up some data structures. Blocking this connection (e.g. with a firewall) will cause the initialization code to wait for a socket timeout, after which the system falls back on using IPv4.</p> <p>Workaround:</p> <p>Either set <code>-Djava.net.preferIPv4Stack=true</code>, which forces Java to use IPv4 instead, or you disable IPv6 entirely in the system. The proper fix is to allow IPv6 traffic from localhost to localhost.</p> <p>For more information, see the Sun documentation:</p> <p>http://java.sun.com/j2se/1.4.2/docs/guide/net/ipv6_guide/#ipv6-networking</p>

Issue ID	Description
CR265793	<p>With JRockit 1.4.2 R26, <code>java.lang.reflect.Array.set(Object array, int index, Object value)</code> always throws <code>NullPointerException</code> when value is null without checking if array is of primitive type.</p> <p>A patch addressing this issue is available through BEA support.</p>
CR265227	<p>The BEA JRockit in Silent mode will not be correctly installed if the <code>USER_INSTALL_DIR</code> in the XML-file has been set to anything other than the default installation path.</p> <p>During the installation, the registry settings will not be set correctly, causing the <code>.jar</code> file association to fail. The files <code>Java.exe</code> and <code>Javaw.exe</code> will not be copied to <code>%SystemRoot%\System32</code> either.</p> <p>This problem has been fixed in R26.3.</p>
CR264913 , CR244553	<p>A bug in the Linux operating system on x64 will cause BEA JRockit to crash if it is invoked from the <code>pthread_once</code> system call.</p> <p>Workaround:</p> <p>Install RHEL 4.0 QU3 or SUSE 9.0 SP3.</p>
CR262540	<p>An issue running HP OpenView Java Diagnostics Profiling Agent may cause crashes with BEA JRockit R26.0.0.</p> <p>This issue has been fixed in R26.3.0.</p>
CR262157	<p>In rare cases BEA JRockit can print incorrect information about locks in the stack dump, given by the <code>Ctrl-Break</code> handler or by the Management Console/MAPI.</p> <p>When a lock is taken by a method that has been optimized in a certain way (inlining), this lock can be printed as being taken, not only on the correct frame, but also as being taken on one or several nearby frames. This does not affect how BEA JRockit treats the lock when executing, only the stack dump itself.</p>
CR256312	<p>On Windows x64 and Itanium, when using the BEA JRockit JRE console mode installer to remove a previously installed BEA JRockit, the uninstall will be interrupted and the following message is displayed:</p> <p>A fatal error has occurred. This application will terminate.</p> <p>The uninstall information is now removed, but most of the files and registry settings are still left on the machine.</p> <p>Workaround to completely uninstall the BEA JRockit JRE:</p> <p>Run the installer in graphical mode and click <code>Remove previous and reinstall</code> when prompted; the BEA JRockit JRE is once again installed. To completely uninstall the BEA JRockit JRE (including its files and registry settings), use the graphical uninstall procedure.</p>

Issue ID	Description
CR252610	<p>A problem regarding the handling of TrueType fonts for certain font files may cause a call to <code>java.awt.Font.getXXX()</code> methods that results in an <code>IllegalArgumentException</code> being thrown.</p> <p>This problem has been reported to Sun as a problem found in Sun JDK 5.0 Update 4 in bug #6349101 (http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6349101). This problem can theoretically occur on all platforms but has been observed on RedFlag Linux Advanced Server 5.0 when the RPM package <code>ttfonts-zh_TW-5.0-2AX.noarch.rpm</code> is installed and the user is requesting a font with Chinese locale.</p> <p>Workaround:</p> <p>Uninstall the package <code>ttfonts-zh_TW-5.0-2AX</code>. This solves the problem but will at the same time remove those fonts from the system, which may cause other problems when trying to display Chinese text. If you try this, you should try to replace the use of the removed font with some other available font on the system.</p> <p>The problem has been fixed in release R26.3.0, but since that release does not include an Itanium version, it is still a known issue for the R26.0.0 Itanium version.</p>
CR251457	<p>If <code>ulimit -v</code> is used on Linux to limit the virtual memory usage, BEA JRockit may crash if the limit is set too low. The x64 version of BEA JRockit requires a much larger setting because it reserves addresses for compiled code at startup. The BEA recommendation is to not use the <code>ulimit -v</code> setting at all.</p>
CR251452	<p>BEA JRockit needs enough virtual memory (address space) to be able to run properly. Note that a high value on allowed virtual memory does not imply high memory consumption. On Linux, the amount of available virtual memory can be changed, typically by the command <code>ulimit -v <amount></code>.</p> <p>The default is an unlimited address space, which is the recommended. Do not change this default, since it risks having BEA JRockit running out of address space, which in turn causes BEA JRockit to terminate immediately.</p> <p>BEA JRockit requires at least the following amount of virtual memory to even start:</p> <p>On IA64: 88 MB On IA32: 61 MB On x64: 1100 MB</p> <p>If you start BEA JRockit with less virtual memory than twice the above values, you'll be given a warning. This is not recommended, and is likely to cause problems when running an actual Java application.</p>
CR249667	<p>Some applications may experience problems with the automatic nursery sizing heuristics when running in <code>-Xgcprio:throughput</code> (default) and <code>-Xgcprio:pausetime</code> mode. This causes too frequently triggered garbage collections.</p> <p>Workaround:</p> <p>Set the nursery size manually using <code>-Xns:<size></code> or to select a static garbage collector.</p> <p>The automatic heap resizing heuristics are not optimal for all applications. If your application has problems due to frequent garbage collections and the heap hasn't been expanded to the maximum heap size, you can increase the initial heap size (<code>-Xms:<size></code>) to improve the performance.</p>

Issue ID	Description
CR248565	When running BEA JRockit as an embedded service on Windows (for example, Jakarta Tomcat service wrapper), the directory <path-to-jdk>/jre/bin must be added to the PATH variable.
CR248551	When installing the 32-bit JRE on a Windows x64, the java.exe and javaw.exe files will not be copied to the system32 folder; they will be placed in the syswow64 folder instead. This is expected behavior for 32-bit applications running on Windows x64.
CR247613	The "jrcmd" utility shipped in the Windows ia32 BEA JRockit package does not work on Windows x64. Instead, use the "jrcmd" utility shipped with the Windows x64 package of BEA JRockit.
CR246634	Thread priorities are supported on the Windows platforms only.
CR246224, CR260004	When doing a JRA recording on Windows XP x64, the JRA recording incorrectly displays that it has been done on Windows 2003 Server. The current implementation of BEA JRockit cannot separate between the two different operating system's version information. This known issue is valid for R26.0 and R26.1.
CR245914	Java applications (such as Eclipse) may hang on Linux distributions which use the "gamin" File Alteration Monitor implementation, for example RHEL4. This is due to a bug in gamin's handling of signals. Workaround: Use "signal chaining" by loading the libjsig.so library. Do this by executing > export LD_PRELOAD=\$JDK_HOME/jre/lib/i386/libjsig.so before starting BEA JRockit. For more information on signal chaining, see: http://java.sun.com/j2se/1.5.0/docs/guide/vm/signal-chaining.html
CR244773 CR250025	BEA JRockit's nursery pool can be invalidated when the garbage collection strategy changes. According to the Java 2 Platform SE API documentation, a pool that has been invalidated may return null. Programmers must therefore assume that MemoryPoolMXBean#getUsage() and MemoryPoolMXBean#getPeakUsage() may return null at any time. The MemoryMonitor demo and the VerboseGC demo can throw NullPointerExceptions, since these attributes are not checked for nulls.
CR243996	The VerboseGC demo, located at \demo\management\VerboseGC\VerboseGC.jar, can throw a NullPointerException when used with a BEA JRockit that has a nursery. According to the API specification for the Java 2 Platform Standard Edition 5.0 the method getUsage() of the MemoryPoolMxBean can return null if a memory pool is not valid, which can be the case when you run BEA JRockit with a nursery. This validity check is missing in the demo and is the cause of the NullPointerException.

Issue ID	Description
CR242655	<p>In Windows, faulting code can be caught by Structured Exception Handling (SEH). The Microsoft C compiler allows a special construct, see below:</p> <pre data-bbox="691 323 1008 453"> __try { // do something that can fail }__except (filterException()) { // handle the fault } </pre> <p>This sets up an SEH handler, which would get called if the code in the <code>__try</code> block fails (for example, a read/write to an illegal address).</p> <p>However, on 64-bit Windows (IA64 and x64), BEA JRockit uses a new exception handling feature known as vectored exception handlers. The vectored exception handlers will be called before any SEH handler gets called. If the BEA JRockit vectored exception handler detects a fault in native code, it will make BEA JRockit produce a crash dump.</p> <p>The effect of this is that you cannot use SEH on 64-bit Windows in native code that gets called by BEA JRockit. Either install a vectored exception handler yourself and add it first in the chain, or test the memory before trying to read/write to it with <code>IsBadReadPtr()</code>.</p>
CR232872	<p>On RHEL3u6 and earlier, as well as RHEL4u2 and earlier, <code>fork()</code>ing new processes can sometimes fail. This can make in turn make <code>Runtime.exec()</code> fail. This has been fixed in RHEL3u7 and RHEL4u3.</p> <p>The Red Hat Issue Tracker case number for this is 77560. This isn't available in Red Hat's public bugzilla.</p>
CR210743	<p>If you are running SLES 8.0, RFAS 4.1, RHEL 3.0 QU4 or older, you might run into serious IO problems.</p> <p>Workaround (for RHEL):</p> <p>Install version QU5 or later.</p>
CR128962	<p>IPv6 support for Windows is included as an unsupported feature in this release.</p>

