This presentation will cover functional differences that you will want to be aware of when you are migrating your Java environment to the IBM SDK, Java Technology Edition, Version 6, from previous IBM SDK releases.

Agenda

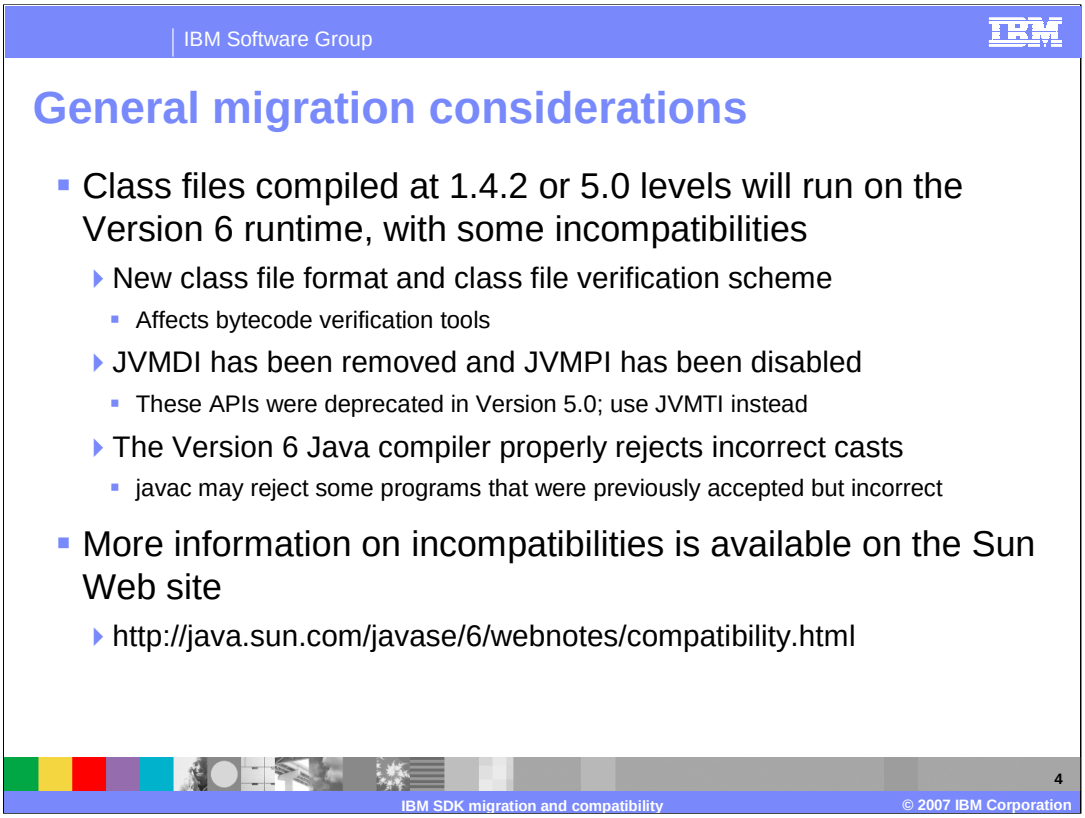
- General migration to the Java SE Platform 6
- IBM SDK migration

This presentation will cover general migration topics associated with moving up to the Java SE Platform 6 specification and migration considerations that are specific to the IBM SDK.

Section

General migration considerations

The first section of the presentation will briefly cover some of the major migration considerations and compatibility issues based on the industry standard Java 6 specification.



For the most part, Java class files that were compiled at 1.4.2 or 5.0 levels will continue to run unchanged on the Version 6 runtime. However, as with any Java specification update, there are some incompatibilities between Version 6 and the previous levels of the Java specification. In this release, the Java class file specification was updated under JSR 202. Again, this does **not** mean that class files that were compiled targeted to 1.4.2 or 5.0 levels are incompatible with Java 6. For the most part, this change will be transparent to users. However, if you develop byte code manipulation tools, you will need to update your tools to support the new class file format. In Java 5, the JVMDI and JVMPI APIs were deprecated and replaced by the JVM Tool Interface, JVMTI. These deprecated APIs were removed or disabled in Java 6, so you will need to use JVMTI instead. In Java 6, the implementation for casting was updated to more closely align with the Java Language Specification. As a result of this change, there may be rare cases in with the Java compiler, javac, will now reject programs that were previously accepted, but that were incorrect. There are many other changes in Java SE 6 that could impact you as you migrate to this release. Additional documentation on Java platform incompatibilities is available from Sun.

Section

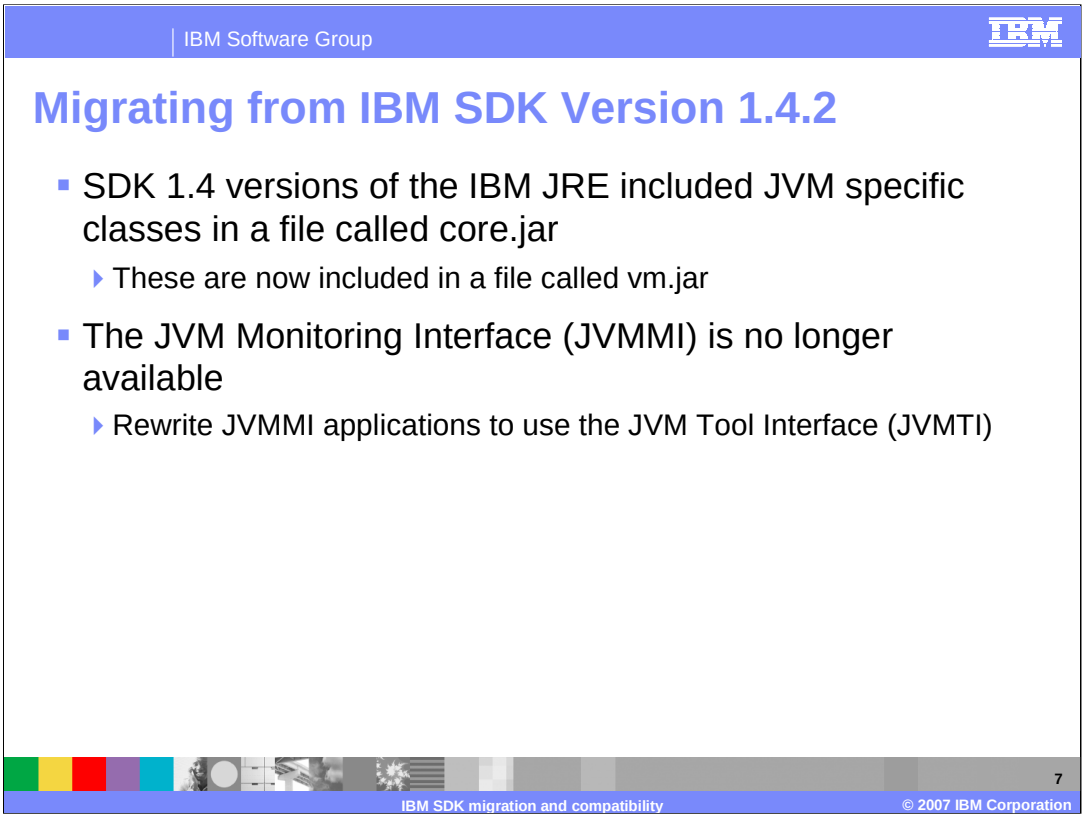
IBM SDK migration

This section of the presentation describes compatibility topics that are specific to the IBM SDK for Java.

Migrating from IBM SDK Version 1.4.2

- From Version 5.0, the IBM SDK contains new versions of the IBM Virtual Machine for Java and the Just-In-Time (JIT) compiler
- Implementation of JNI conforms to the JNI specification
  - Differs from the Version 1.4.2 implementation
  - Returns copies of objects rather than pinning the objects
  - This difference can expose errors in JNI application code
  - Use the `-Xcheck:jni` tool to help debug JNI issues
- Format and content of garbage collector verbose logs obtained using `-verbose:gc` have changed
  - Data is now formatted as XML
  - Content reflects the changes to the implementation of garbage collection in the JVM, and most of the statistics that are output have changed

From Version 1.4.2 to Version 5.0, the components of the IBM SDK changed substantially to include new versions of the IBM Virtual Machine for Java and the Just-In-Time compiler. While the IBM SDK has continued to comply with required industry specifications, this architectural shift introduces some differences between the IBM SDK for Java Version 1.4.2 and Version 5.0. The JNI implementation, while it still conforms to the JNI specification, differs from the Version 1.4.2 implementation in that objects are no longer pinned, but rather, copies of those objects get returned. This change could expose errors in your JNI applications, and you can use the –Xcheck:jni command-line tool to help debug JNI issues. The garbage collector component in the virtual machine also changed in Version 5.0. The verbose GC logs produced by the new garbage collector are in XML format and contain new data and statistics that reflect the structure of the updated garbage collector.



The shift to the Version 5.0 SDK also saw some packaging and API support changes. Classes that were previously packaged in the core.jar file are now included in a file called vm.jar. The Java 5 specification introduced a new debugging and profiling interface called the JVM Tool Interface. The JVMMI is no longer available, and any existing JVMMI applications will need to be updated to use the new JVMTI specification.

Migrating from IBM SDK Version 5.0

- JVM classes are held in multiple JAR files in the jre/lib directory
  - Replaces the single rt.jar (1.4.2) and core.jar (5.0) from earlier releases
- The JVM shared library libjvm.so is now stored in:
  - `jre/lib/<architecture>/j9vm`, and
  - `jre/lib/<architecture>/classic`
- Tracing class dependencies using `-verbose:Xclassdep` is not supported
  - If you specify `-verbose:Xclassdep`, the JVM will issue an error message and will not start

The packaging structure of the SDK has changed slightly in Version 6. There are now more JAR files that contain core JVM classes. You will find multiple JAR files in the jre/lib directory; these replace the single rt.jar and core.jar files from earlier releases. The location of the shared library libjvm.so has also changed. The exact location of the file will vary depending on the architecture of your system. For example, in the 64-bit SDK for AIX, the libjvm.so library is packaged in the jre/lib/ppc64/j9vm directory. If you have any configuration files or scripts that depend on the location of libjvm.so, you will need to update those files to point to the new file location. Finally, the ability to trace class dependencies using the –verbose:Xclassdep option is no longer supported. If you try to use that option, the JVM will issue an error message and will not start.

z/OS® migration considerations

- The serial reusability feature of the IBM SDK for z/OS®, Version 1.4.2 (31-bit) and earlier is not supported
  - If you specify any of these options associated with serial reusability – `-Xresettable, -Xjvmset, -Xscmax` – the JVM will issue an error message and will not start
  - You can share data between JVMs in an address space (the old `-Xjvmset` and `-Xscmax` options) using class data sharing between JVMs
  - The `-Xinitacsh` and `-Xinitth` options, which allowed heap sizes to be specified for the resettable JVM, are ignored
- The system property os.arch for IBM SDK for z/OS, Version 1.4.2 (31-bit) and earlier had a value of 390
  - For Java 5.0 and beyond, the value of os.arch is s390

The compatibility issues described on this slide are for the IBM SDK for Java on z/OS and apply to users who are migrating from Version 1.4.2 of the SDK. From Version 5.0 and on, the serial reusability feature is no longer supported, and the command line options associated with serial reusability will no longer be honored. Some of these parameters will cause the JVM to issue an error message and not start, while others will be ignored. You can take advantage of the shared data cache that was introduced in Version 5.0 if you would like to share data between JVMs in an address space. Finally, for Java 5.0 and beyond, the system property os.arch associated with the 31-bit SDK on z/OS has changed from 390 to s390.

Section

Summary and references

This section contains a summary and references.

Summary

- The IBM SDK, Java Technology Edition, Version 6, supports most classes compiled at the 1.4.2 and 5.0 levels
- Since Version 1.4.2, the underlying IBM Virtual Machine for Java and JIT compiler have changed
- Since Version 5.0, SDK packaging has changed
- Special migration considerations apply to the IBM SDK for z/OS

The IBM SDK, Java Technology Edition, Version 6, will support most class files that were compiled at the 1.4.2 or 5.0 level. Some incompatibilities introduced in Version 6 have to do with the new class file format, updates to the javac Java compiler, and deprecated tools APIs. A link to online documentation of industry standard incompatibilities in the Java specification is available in the references section of this presentation. Since Version 1.4.2, the underlying IBM Virtual Machine for Java and JIT compiler have changed, but are still compliant to industry standards. Since Java 5, some of the IBM SDK packaging has been updated. In particular, you should pay attention to the new location of the libjvm.so library and update any scripts that you have to point to the new file location. Special migration considerations may apply when migrating to Version 6 of the IBM SDK for Java on z/OS.

References

- Sun's Java SE 6 Release Notes
  - http://java.sun.com/javase/6/webnotes/features.html
- Sun's Java SE 6 Compatibility Notes
  - http://java.sun.com/javase/6/webnotes/compatibility.html
- Diagnostics guide
  - http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_Java6_Migration.ppt

This module is also available in PDF format at: ../Java6_Migration.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

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

IBM z/OS

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

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.