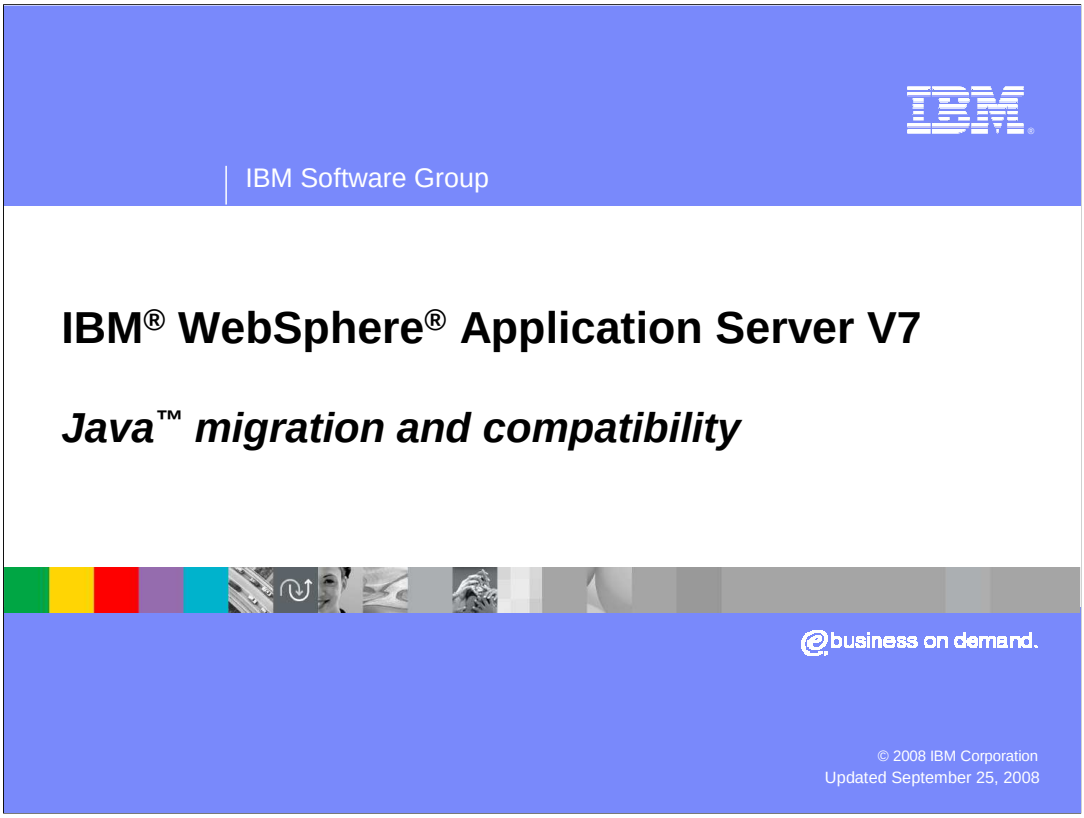


This presentation covers Java compatibility between the IBM SDK for Java Version 6, which is the underlying Java runtime for WebSphere Application Server V7, and previous releases of the IBM SDK for Java.

Agenda

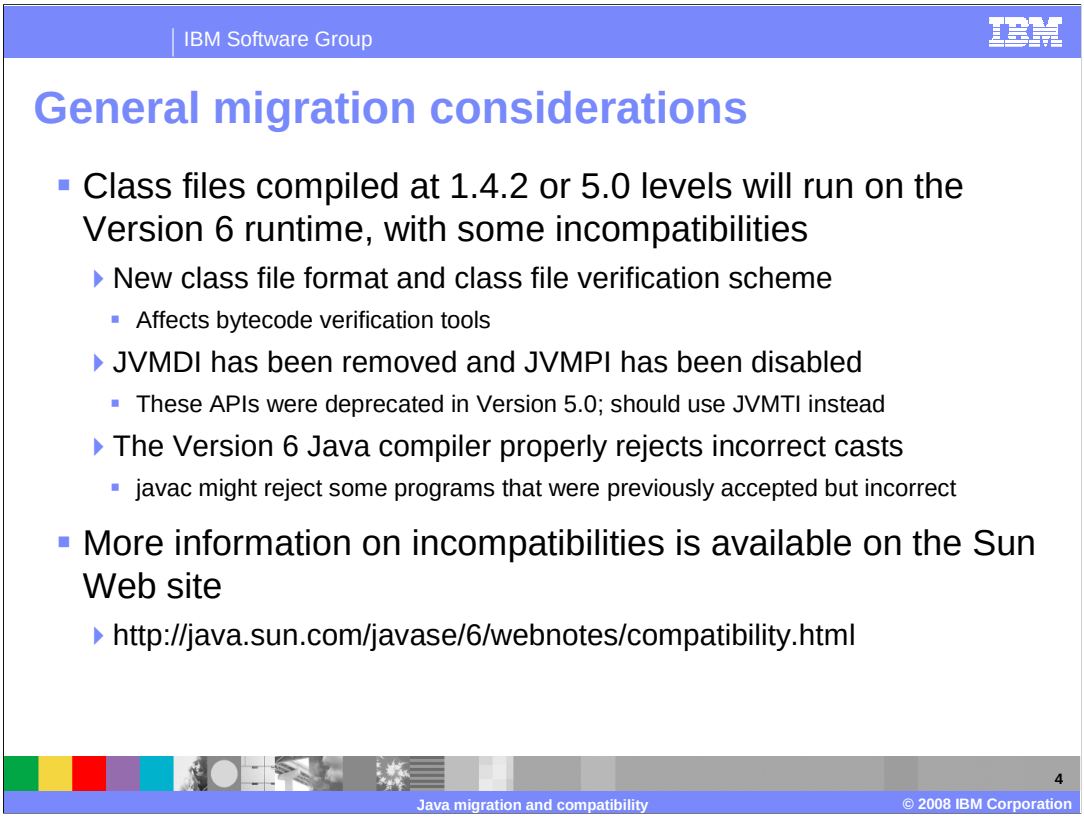
- Java compatibility
- Considerations for WebSphere Application Server

The first section of this presentation covers general Java compatibility topics that are relevant to any Java application. The second section includes topics that are specifically related to WebSphere Application Server.

Section

Java compatibility

This portion of the presentation covers general Java compatibility topics. Most of the ideas discussed here are related to potential application migration issues that people have seen when migrating to Java SE 6 from previous releases. The majority of these issues are based on changes in the underlying Java 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 is 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 might 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 can impact you as you migrate to this release. Additional documentation on Java platform incompatibilities is available from Sun.

Migrating from Java 1.4.2 or earlier

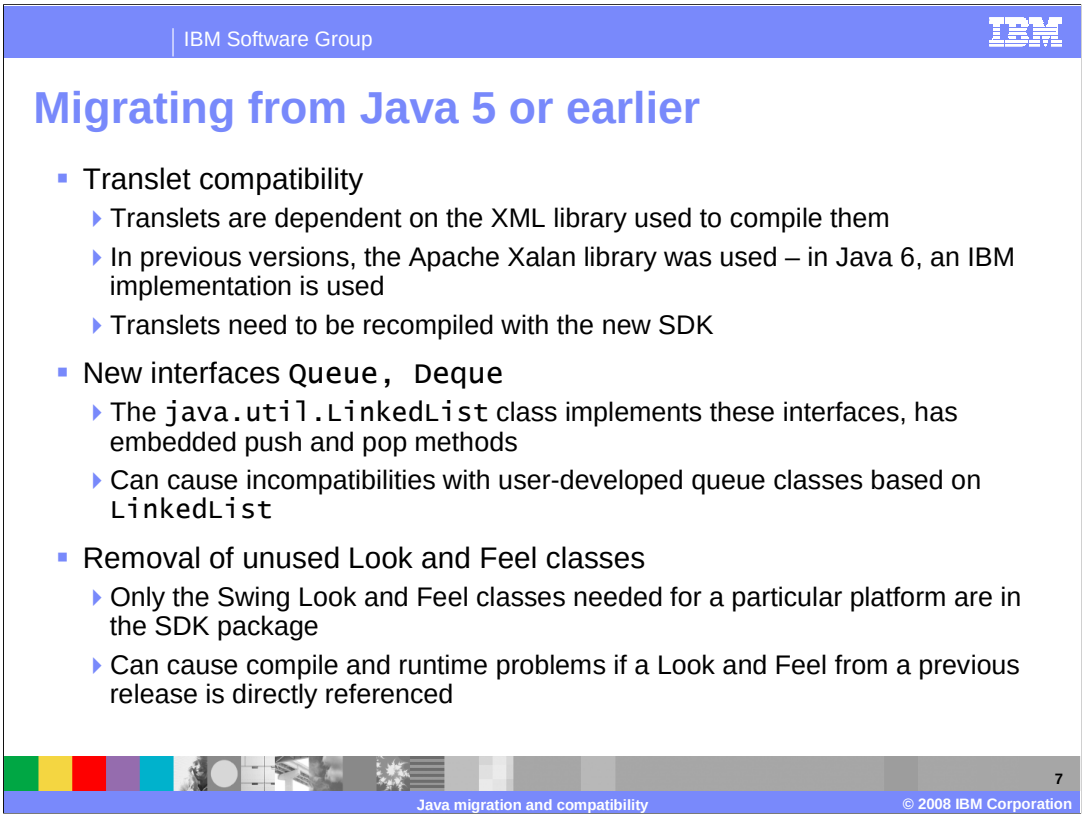
- JNI implementation conforms to the JNI specification, but differs from the Version 1.4.2 implementation
  - Returns copies of objects rather than pinning the objects
  - This difference can expose errors in JNI application code
- Format and content of garbage collector verbose logs obtained using **–verbose:gc** have changed
  - Data is now formatted as XML
  - Content reflects the changes to the implementation of garbage collection in the JVM
  - Most of the statistics that are output have changed
- JVM Monitoring Interface (JVMMI) is no longer available
  - You must rewrite JVMMI applications to use the JVM Tool Interface (JVMTI) instead

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 can 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 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 Java 5 or earlier

- Minor packaging changes
  - The JVM shared library libjvm.so is now stored in jre/lib/<arch>/j9vm and jre/lib/<arch>/classic
  - JVM classes are held in multiple JAR files in the jre/lib directory, replacing the single rt.jar and core.jar from earlier releases
- Generification issues
  - More core classes have been generified, which can introduce compile-time incompatibilities
  - **Example:**
    - Constructors from `javax.management.ObjectName` class
    - ***Java 5:*** `ObjectName(String domain, Hashtable table)`
    - ***Java 6:*** `ObjectName(String domain, Hashtable<String,String> table)`

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. The concept of generics was introduced in Java SE 5, and now in Java SE 6, more internal classes have been modified to use generics, in other words, they have been generified. This can introduce some compile-time incompatibilities with older classes. One example is with the javax.management.ObjectName class, whose ObjectName(String domain, Hashtable table) constructor was changed to ObjectName(String domain, Hashtable<String,String> table). It is possible to pass a Properties object into the original constructor because Properties extends java.util.Hashtable<Object, Object>. However, now that the constructor requires a more-specific Hashtable<String, String>, a Properties object is no longer valid because Hashtable<Object, Object> cannot be cast to Hashtable<String, String>. In cases like these, you need to rewrite your Java source code to correspond to the new specification.



Translets, which are precompiled XSLTs, are dependent on the XML library used to compile them. In previous versions of Java, the Apache Xalan library was used, but this release uses an IBM implementation. Translets will need to be recompiled on the new SDK before they will work. This will not affect XSLTs compiled at runtime using the JAXP interface. Also, the Xalan command-line compiler will still be available, but internally it compiles with the new implementation.

It is fairly common to use the java.util.LinkedList class as a base class for a queue by adding a *push* and a *pop* method. In Java SE 6, the LinkedList class now implements the new Queue and Deque interfaces and has its own *push* and *pop* methods. A compile-time error can occur if the user-implementation of those methods is incompatible with the base implementation.

Past versions of the class library in the IBM SDK contained Swing Look and Feel classes for all platforms, even if they were not used on a particular platform. In the Java SE 6 class library, only the Look and Feel classes needed for a particular platform are included. This can cause compile and runtime problems if a specific implementations of Look and Feel classes are directly referenced. There is rarely a need to directly interact with an implementation, but if it is required, it is better to do so reflectively, or to handle any exceptions that might occur if the Look and Feel class is not available.

Section

Considerations for WebSphere Application Server

This section describes special considerations for running Java applications in a mixed node environment in WebSphere Application Server.

Running in mixed-release cells

- Since older releases of the application server run on older versions of Java, special considerations need to be taken for mixed nodes
- Can use the Java compiler to target compilation for a particular release and maintain bytecode compatibility
- If bytecode compatibility is not used
  - Any application or resource classes compiled against Java 6 will only run on WebSphere Application Server V7 nodes
  - Any application or resource classes compiled against Java 5 will only run on WebSphere Application Server V6.1 or higher nodes

In WebSphere Application Server, it is possible to build mixed-release cells that contain nodes that are running on different levels of the product. For example, you might have a Version 7 deployment manager that manages a Version 7 node and a Version 6.1 node. Since older releases of the application server run on older versions of Java, mixed nodes require special consideration. For example, an application compiled with the default settings in WebSphere Application Server V7 is compiled according to the Java SE 6 specification, and you are unable to run it in your Version 6.1 node because that node runs on Java SE 5. One way to work around this is to use the target release settings on the Java compiler to compile applications targeted at a previous release. This will force the compiler to maintain bytecode compatibility with an older version of the Java SE specification. If bytecode compatibility is not used, then any application or resources classes compiled against Java 6 will only run on WebSphere Application Server V7 nodes, and any application or resource classes compiled against Java 5 will only run on WebSphere Application Server V6.1 or higher nodes.

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
- Changes in the Java SE 6 specification might introduce incompatibilities with older classes
- Use bytecode compatibility to run mixed-release cells

While the IBM SDK for Java Version 6 supports most classes compiled at the 1.4.2 and 5.0 levels, some changes in the Java SE 6 specification might introduce incompatibilities with older classes. For example, the Java Virtual Machine Tool Interface is now the supported set of APIs for developing debugging and profiling tools, and more internal classes incorporate generics, which can introduce some compile-time failures. In WebSphere Application Server, you can compile Java programs with bytecode compatibility enabled which will allow you to run applications compiled in WebSphere Application Server V7 to run in a mixed-release cell.

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

Here are some links to related resources.

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_WASv7_JavaCompatibility.ppt

This module is also available in PDF format at: ../WASv7_JavaCompatibility.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 WebSphere

A current list of other IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

Java, JNI, 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.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.