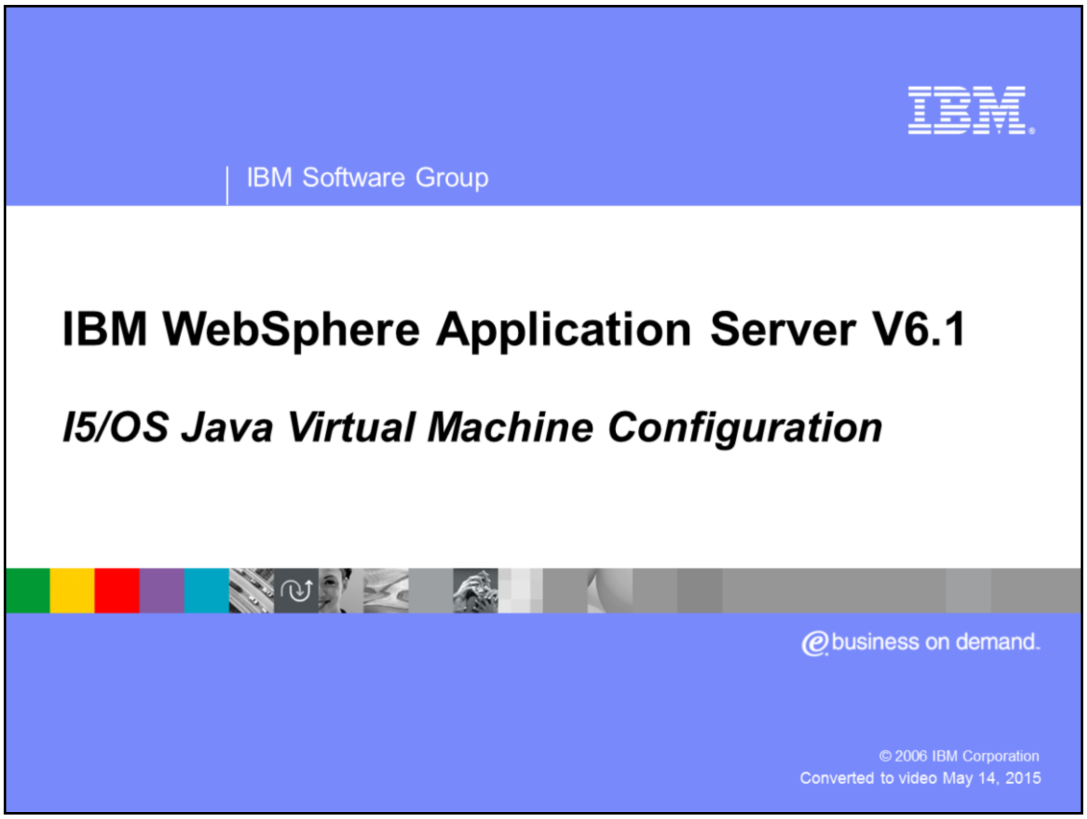


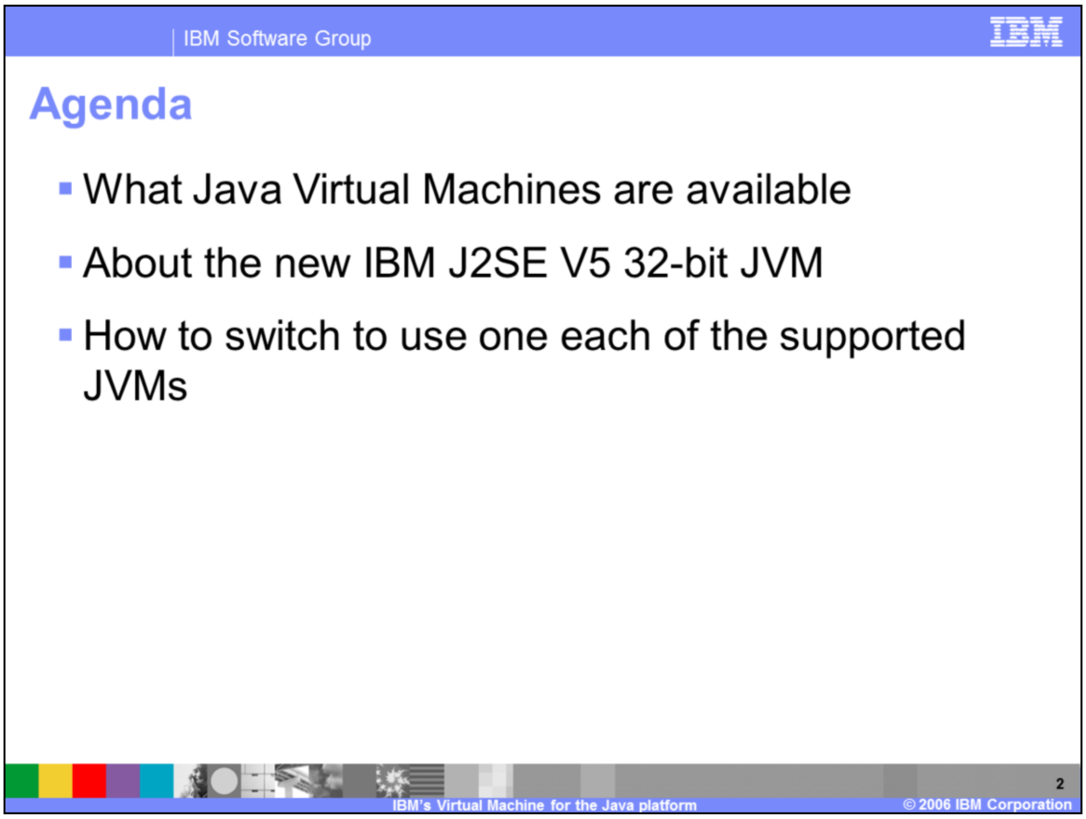
IBM Software Group

IBM WebSphere Application Server V6.1

I5/OS Java Virtual Machine Configuration

business on demand.

This presentation discusses how to enable your application server to use the new IBM J2SE™ Version 5 32-bit JVM when running WebSphere® Application Server V6.1 for i5/OS release V5R4M0.



Agenda

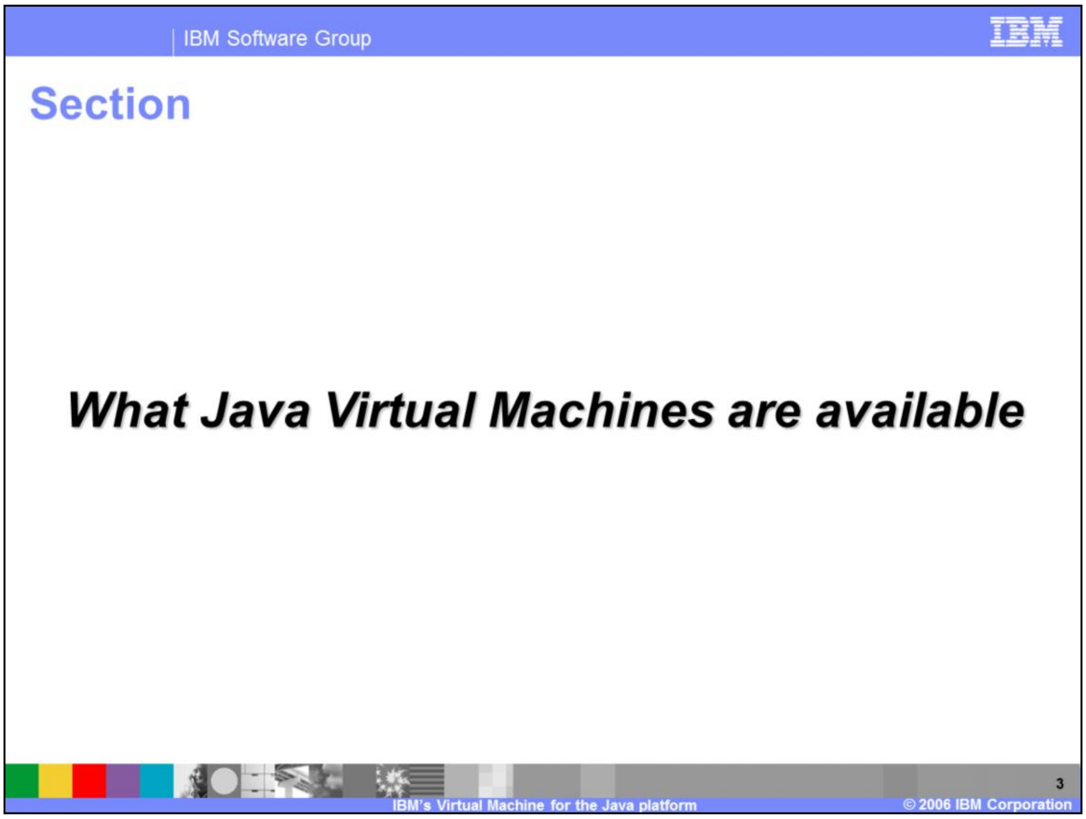
- What Java Virtual Machines are available
- About the new IBM J2SE V5 32-bit JVM
- How to switch to use one each of the supported JVMs

The goal is to:

•give you an understanding of the two JVMs available on i5/OS starting in V5R4M0

•provide information on the new IBM 32-bit JVM that is available in V5R4M0

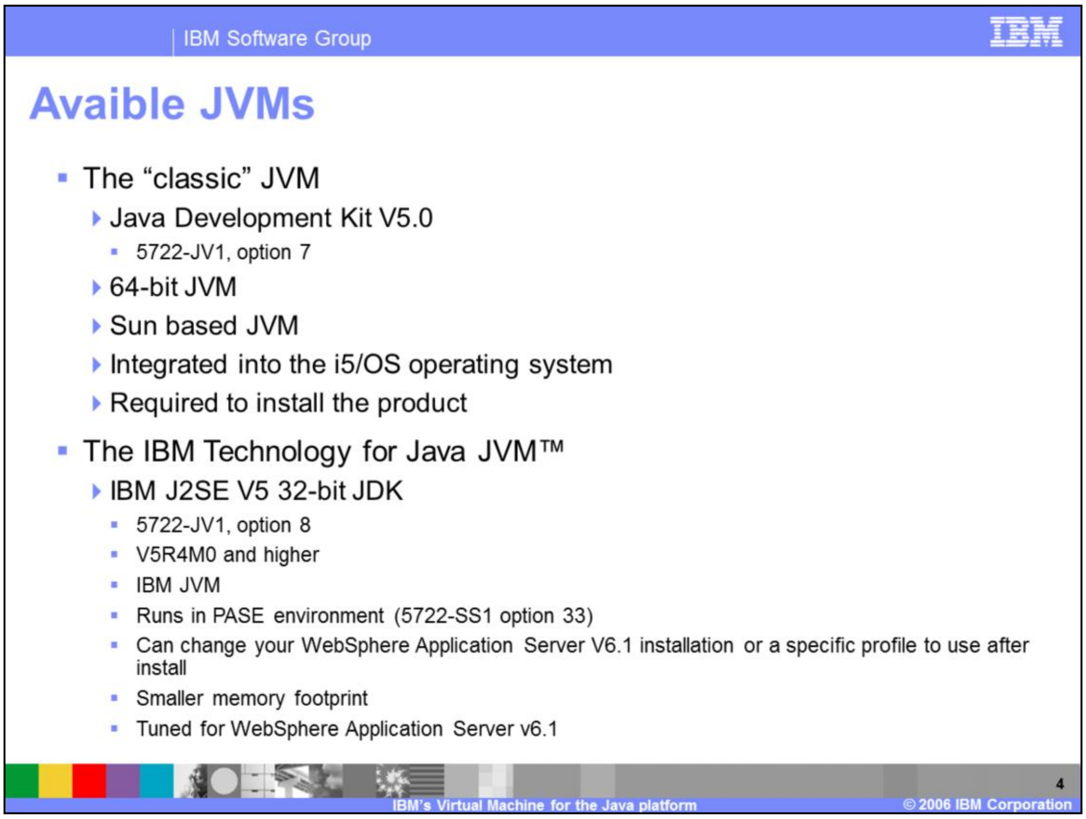
•and show you how to enable your WebSphere Application Server installation or just a specific profile, to use the new IBM JVM.



Section

What Java Virtual Machines are available

This section indicates which Java virtual machines are available to use with WebSphere Application Server V6.1 on i5/OS.



IBM Software Group

Avaible JVMs

- The "classic" JVM
  - Java Development Kit V5.0
    - 5722-JV1, option 7
  - 64-bit JVM
  - Sun based JVM
  - Integrated into the i5/OS operating system
  - Required to install the product
- The IBM Technology for Java JVM™
  - IBM J2SE V5 32-bit JDK
    - 5722-JV1, option 8
    - V5R4M0 and higher
    - IBM JVM
    - Runs in PASE environment (5722-SS1 option 33)
    - Can change your WebSphere Application Server V6.1 installation or a specific profile to use after install
    - Smaller memory footprint
    - Tuned for WebSphere Application Server v6.1

4

IBM's Virtual Machine for the Java platform © 2006 IBM Corporation

Beginning with i5/OS release V5R4M0, there are two JDK 5.0 JVMs available for you to use with WebSphere Application Server V6.1 for i5/OS.

The "Classic" JVM is the i5/OS specific JVM which is integrated into i5/OS. The Classic JVM is a 64-bit, Sun®-based JVM. This is the JDK used to install the WebSphere Application Server version 6.1 product and the one the application server is configured to use out of the box.

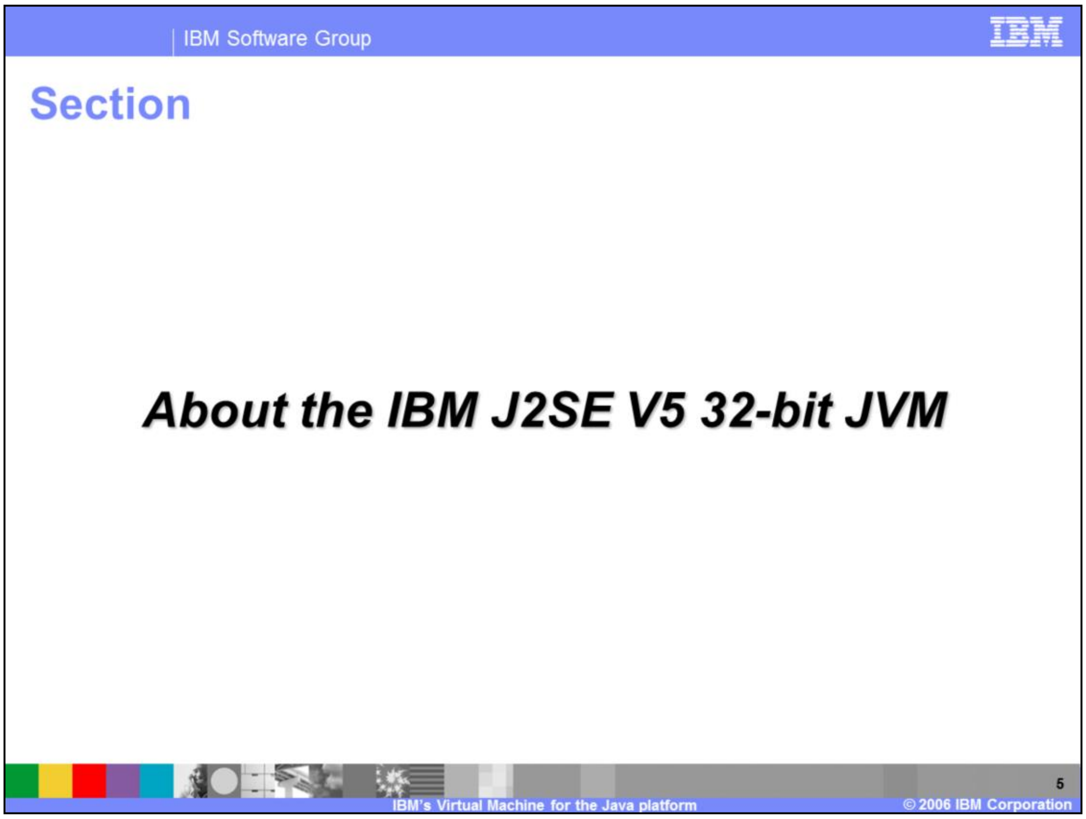
The IBM Technology for Java version 5 JVM is the new IBM J2SE Version 5 32-bit JVM available for V5R4M0 and higher. This is an IBM JVM that runs in the Portable Application Solutions Enablement (or PASE) environment.

IBM Technology for Java has two key potentially beneficial characteristics for WebSphere Application Server V6.1 users:

• *Significant performance improvements for many applications* - Most applications will see at least equivalent performance when comparing WebSphere Application Server on the Classic VM to IBM Technology for Java, with many applications seeing improvements of up to 20%.

•*The second key characteristic is that 32-bit addressing allows for a potentially considerable reduction in memory footprint* - Object references require only 4 bytes of memory as opposed to the 8 bytes required in the 64-bit Classic VM. For users running on small systems with relatively low memory demands this could offer a substantially smaller memory footprint. Performance tests have shown approximately 40% smaller Java Heap sizes when using IBM Technology for Java when compared to the Classic VM.

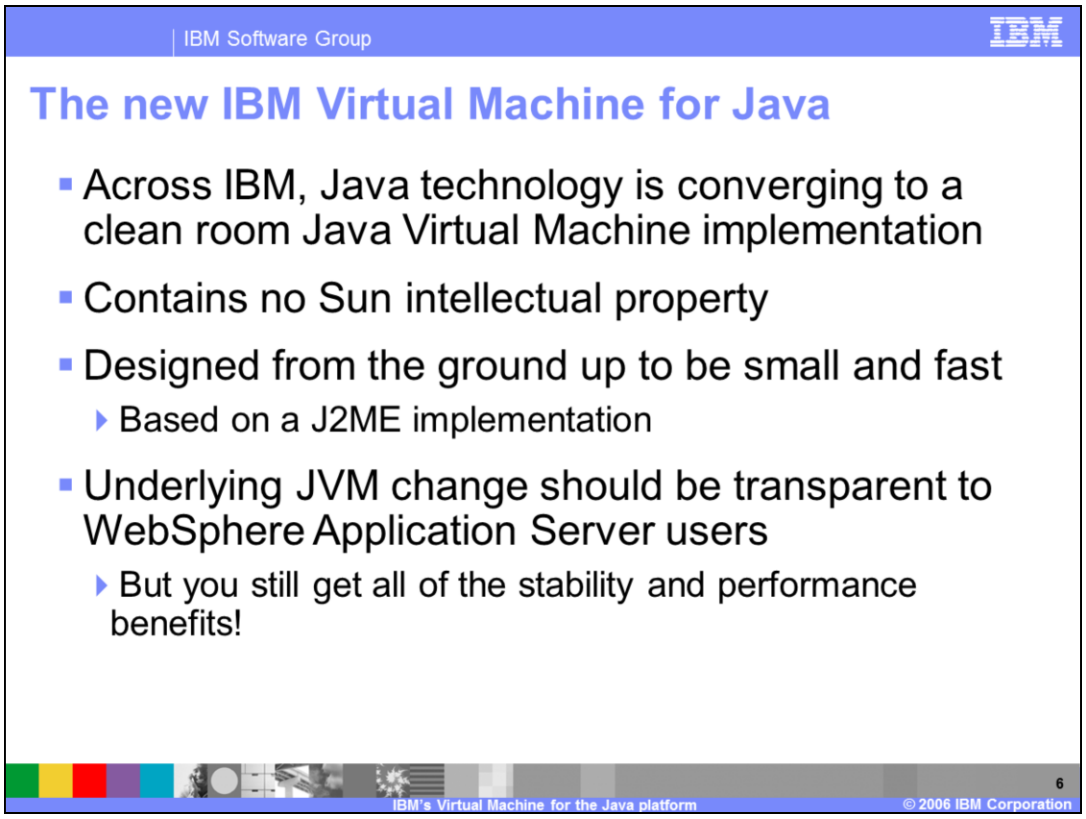
While 32-bit addressing can provide smaller memory footprints for some applications, it is imperative to understand the other end of the spectrum. Applications requiring large Java heaps may not be able to fit in the space available to a 32-bit implementation of Java. The 32-bit VM has a maximum heap size of around 2500 Megabytes due in part to WebSphere related memory demands like shared classes. The Classic VM should be used for applications that require a heap larger than 2500 MB.



Section

About the IBM J2SE V5 32-bit JVM

This section discusses the new IBM J2SE version 5 32-bit JVM.


IBM Software Group

The new IBM Virtual Machine for Java

- Across IBM, Java technology is converging to a clean room Java Virtual Machine implementation
- Contains no Sun intellectual property
- Designed from the ground up to be small and fast
  - Based on a J2ME implementation
- Underlying JVM change should be transparent to WebSphere Application Server users
  - But you still get all of the stability and performance benefits!

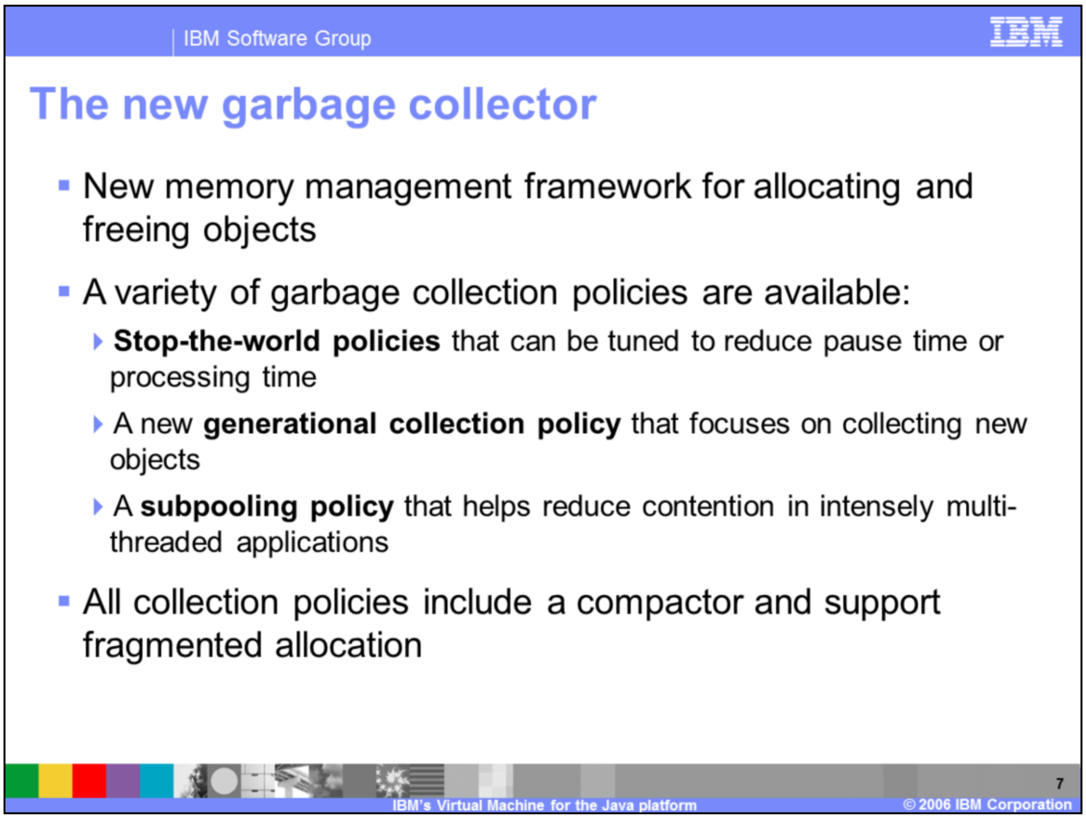
6

IBM's Virtual Machine for the Java platform

© 2006 IBM Corporation

Across IBM, Java technology is converging to a single, clean room Java Virtual Machine implementation containing no Sun intellectual property. This new Virtual Machine leverages the Java expertise of people all across the company and has been developed entirely in-house. This new Java technology replaces operating system specific Virtual Machine implementations and the virtual machine switch should be relatively transparent to you. Pure Java applications, targeted at the proper JDK level, will function unchanged. This new Virtual Machine is based on J2ME implementation, so it has been designed from the ground up to be fast and have a small memory footprint.

In WebSphere Application Server Version 6.1, this new virtual machine is supported on all platforms including i5/OS starting with release V5R4M0.

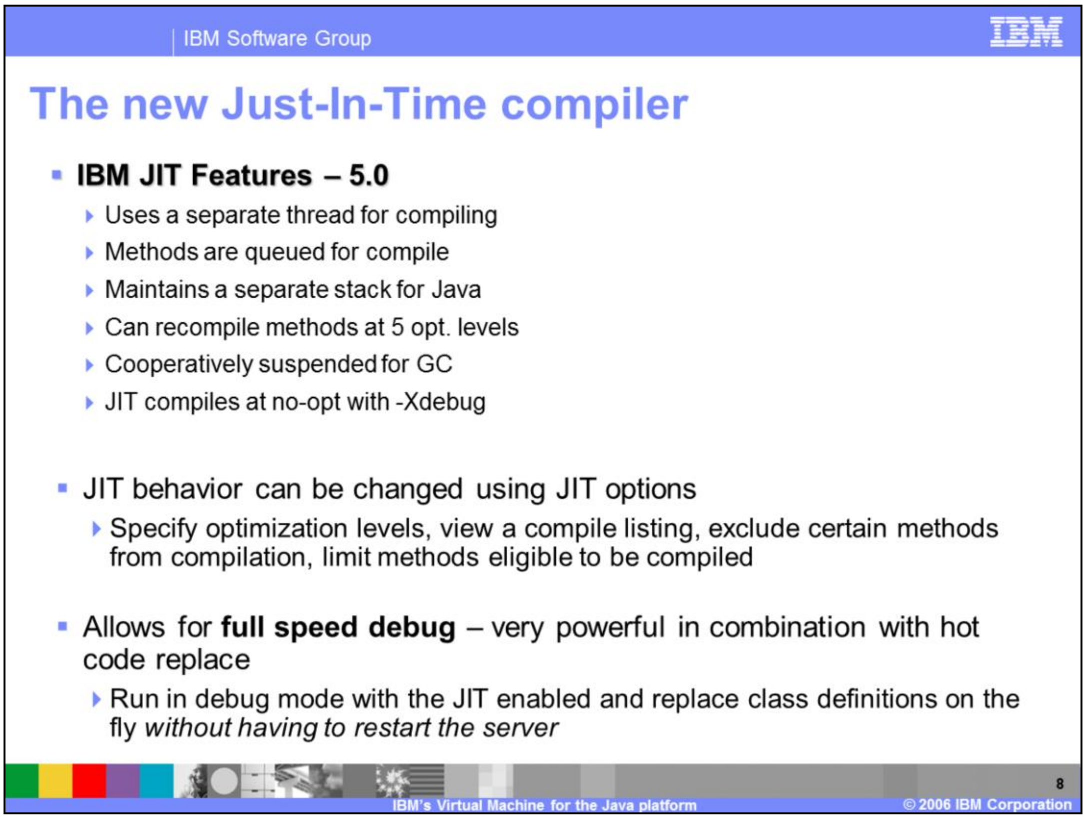


The new garbage collector

- New memory management framework for allocating and freeing objects
- A variety of garbage collection policies are available:
  - **Stop-the-world policies** that can be tuned to reduce pause time or processing time
  - A new **generational collection policy** that focuses on collecting new objects
  - A **subpooling policy** that helps reduce contention in intensely multi-threaded applications
- All collection policies include a compactor and support fragmented allocation

The new Virtual Machine for Java is built on a new memory management framework for allocating and freeing objects, which is the new garbage collector. This GC scheme supports several collection policies, including two stop-the-world policies. Optthruput (the default policy) is tuned to reduce overall processing time and should be used when some pauses, which can range from milliseconds to seconds, are acceptable. The second stop-the-world policy, optavgpause, is tuned to reduce the average pause time for a garbage collection cycle, but generally consumes slightly more processing time overall.

The generational garbage collection policy is new in IBM Technology for Java V5. This policy is based on the idea that objects that have been around for a long time are more likely to continue to be needed, so it focuses on trying to collect newly created objects first. This policy aims to both avoid long pause times and to minimize processing time. The subpooling policy can be useful in intensely multi-threaded applications since it splits garbage collection resources into different buckets to reduce resource contention. All of these collection policies include a compactor and also support fragmented allocation. These features help reduce the overall memory footprint of the Virtual Machine.


IBM Software Group

The new Just-In-Time compiler

- **IBM JIT Features – 5.0**
  - Uses a separate thread for compiling
  - Methods are queued for compile
  - Maintains a separate stack for Java
  - Can recompile methods at 5 opt. levels
  - Cooperatively suspended for GC
  - JIT compiles at no-opt with -Xdebug
- JIT behavior can be changed using JIT options
  - Specify optimization levels, view a compile listing, exclude certain methods from compilation, limit methods eligible to be compiled
- Allows for **full speed debug** – very powerful in combination with hot code replace
  - Run in debug mode with the JIT enabled and replace class definitions on the fly *without having to restart the server*

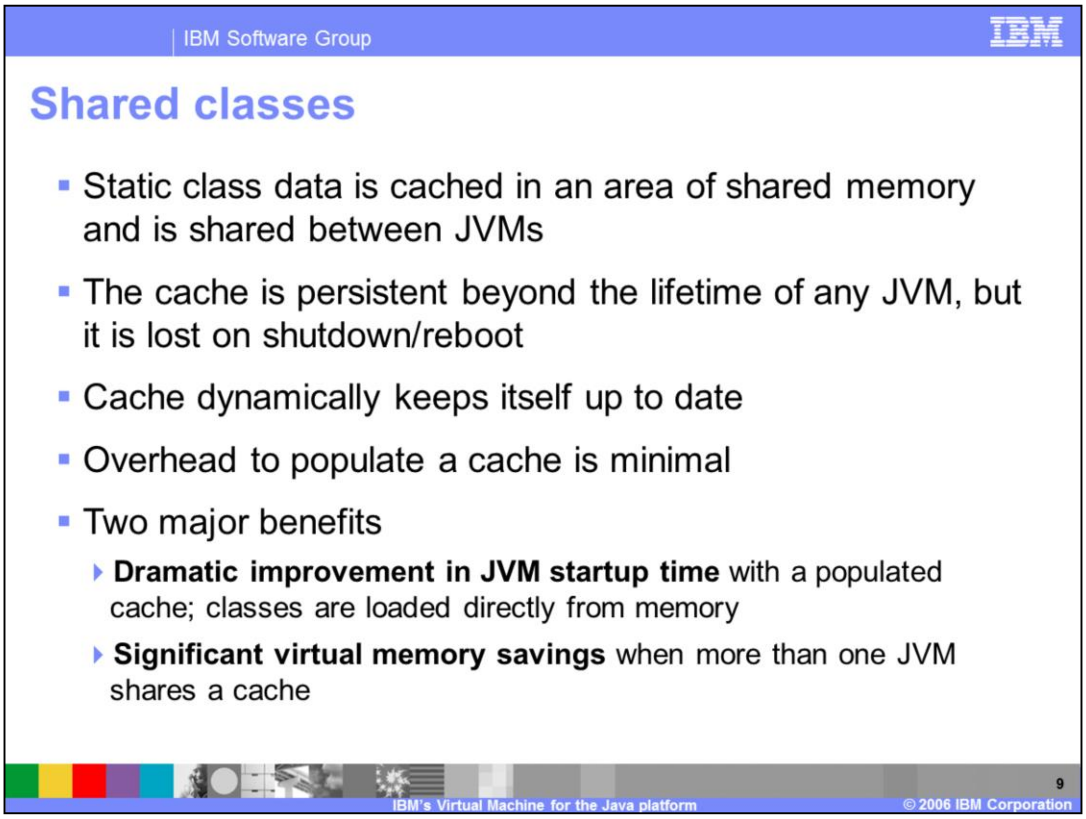
8

IBM's Virtual Machine for the Java platform

© 2006 IBM Corporation

The IBM Just-in-time compiler, or JIT, that is included with the IBM Technology for Java Virtual Machine has several enhancements and different behaviors than previous versions of the IBM JIT. It uses a separate thread for compiling JIT methods, so it is not going to consume any of the resources of your other Java threads. The new JIT also provides five different optimization levels, as opposed to the previous version which had a single optimization strategy. JIT behavior can, of course, be configured using JIT options.

One of the most powerful new features of the JIT is something called full speed debug. This allows you to run with the JIT enabled in debug mode. You can use this feature, in combination with hot code replace, to replace class definitions on the fly. That way, you can, for example, be stopped in debug mode, compile a new version of a Java class that contains a fix that you would like to test out, swap in the new class file, and continue running with that new class file definition in place. This can save you a great deal of time because you can go through the whole process without having to restart your server, and you have the added performance benefit of running with the JIT enabled.



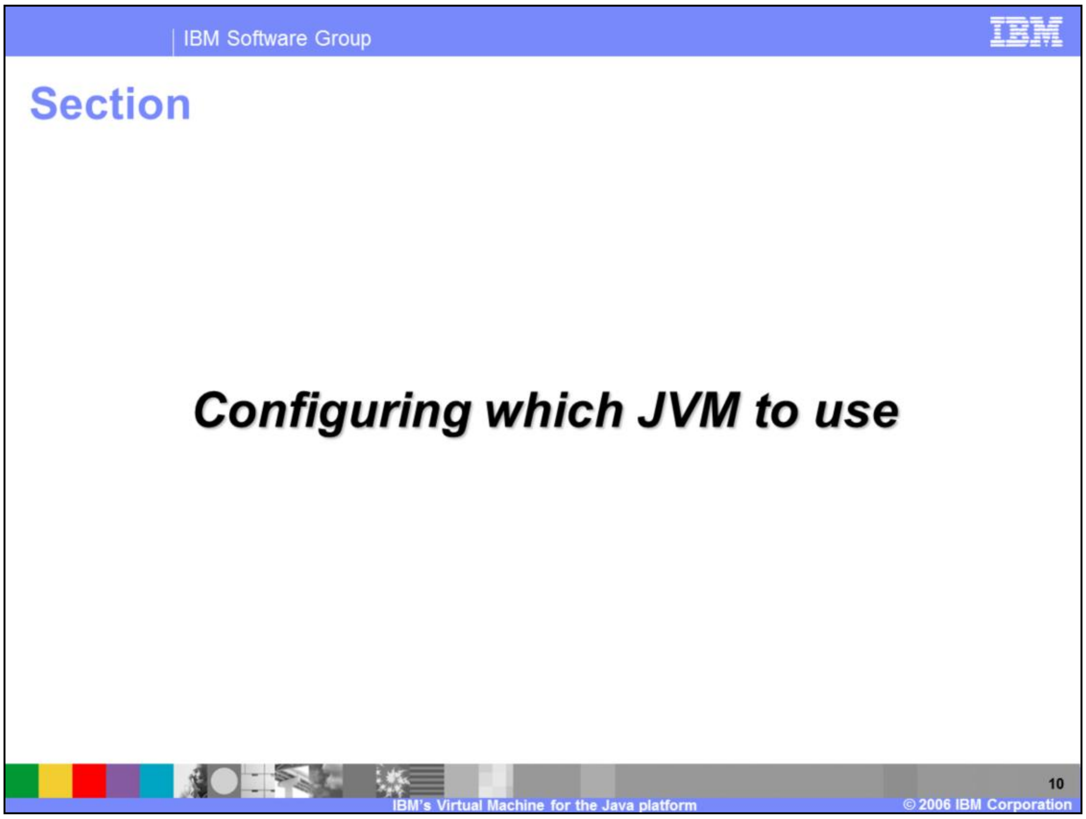
Shared classes

- Static class data is cached in an area of shared memory and is shared between JVMs
- The cache is persistent beyond the lifetime of any JVM, but it is lost on shutdown/reboot
- Cache dynamically keeps itself up to date
- Overhead to populate a cache is minimal
- Two major benefits
  - **Dramatic improvement in JVM startup time** with a populated cache; classes are loaded directly from memory
  - **Significant virtual memory savings** when more than one JVM shares a cache

The shared class cache lives in an area of shared memory accessible to all Java Virtual Machines running on a given system and contains a variety of static class data, for both bootstrap classes and application classes. Any JVM can create a cache, connect to the cache, or modify the cache. There is no defined direction of control or hierarchical relationship. The cache dynamically keeps itself up-to-date, and the overhead to do so is minimal, somewhere between 0-5%, depending on the application and the platform. The behavior of the cache can be configured using JVM options.

The shared class cache is persistent beyond the lifetime of any JVM on the machine. Even if you do not have any JVMs that are currently active and running, the shared class cache still exists. This persistence leads to the first major benefit associated with the shared class cache, which is improved startup time. With a populated cache, many of the classes required for JVM startup are already resident in memory and do not need to be loaded in from your hard drive or some other resource. This can result in dramatic startup speed improvements. The shared class cache is persistent beyond the lifetime of any JVM, but it is lost when your system shuts down or is IPL'd. So, the first time that you start a JVM after your system has been IPL'd, you will experience some delay while the class cache is being populated for the first time.

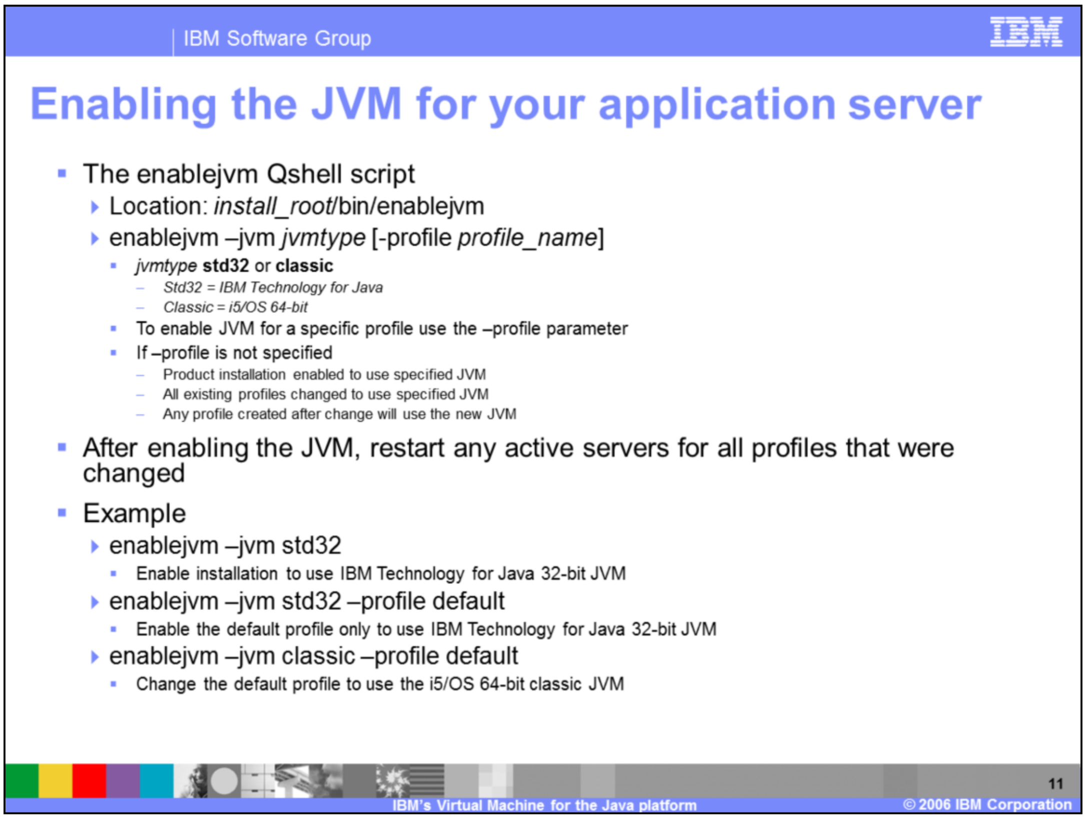
The class cache also provides significant memory savings at runtime because you no longer need multiple copies of class data for several common classes across many JVMs. They are all stored in a single location in the class cache. The more JVMs you have connected to the cache, the more memory you save.



Section

Configuring which JVM to use

This section provides information on how to enable WebSphere Application Server Version 6.1 for i5/OS to use the IBM 32-bit JVM.


IBM Software Group

Enabling the JVM for your application server

- The enablejvm Qshell script
  - Location: *install_root*/bin/enablejvm
  - enablejvm –jvm *jvmtype* [-profile *profile_name*]
    - *jvmtype* **std32** or **classic**
      - *Std32 = IBM Technology for Java*
      - *Classic = i5/OS 64-bit*
    - To enable JVM for a specific profile use the –profile parameter
    - If –profile is not specified
      - Product installation enabled to use specified JVM
      - All existing profiles changed to use specified JVM
      - Any profile created after change will use the new JVM
- After enabling the JVM, restart any active servers for all profiles that were changed
- Example
  - enablejvm –jvm std32
    - Enable installation to use IBM Technology for Java 32-bit JVM
  - enablejvm –jvm std32 –profile default
    - Enable the default profile only to use IBM Technology for Java 32-bit JVM
  - enablejvm –jvm classic –profile default
    - Change the default profile to use the i5/OS 64-bit classic JVM

11

IBM's Virtual Machine for the Java platform © 2006 IBM Corporation

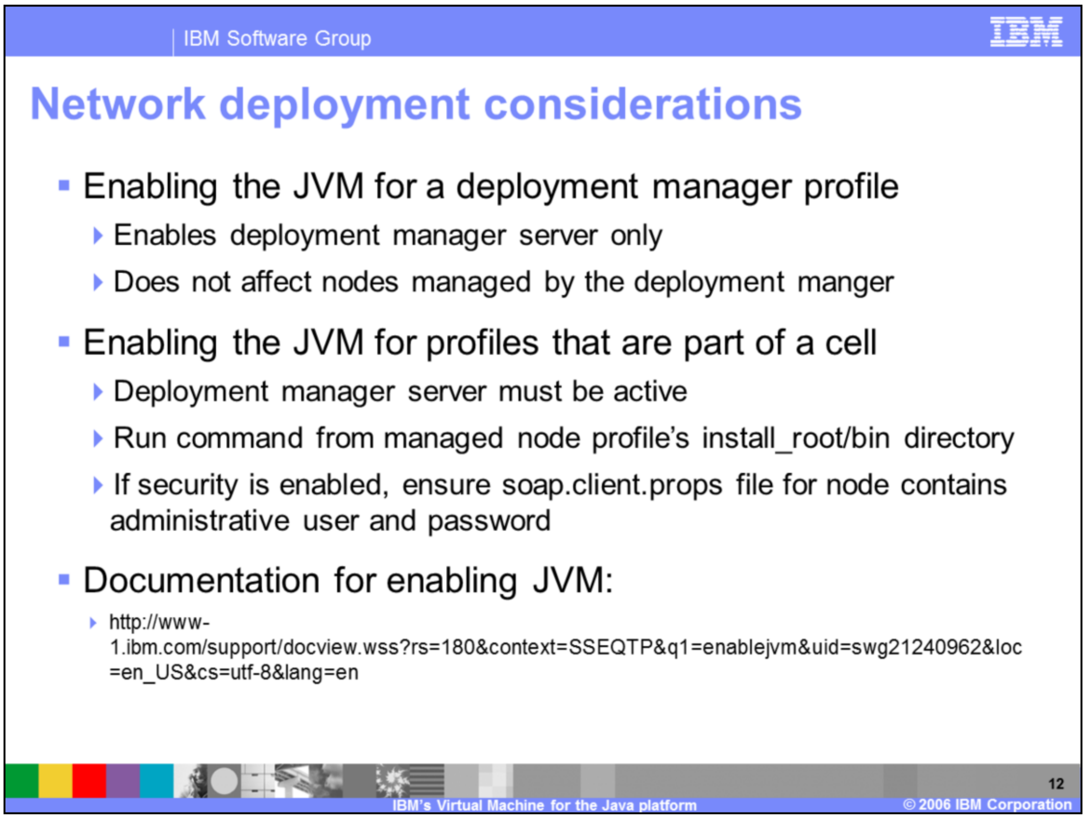
You can use the *enablejvm* Qshell script to change the JVM that your WebSphere Application Server version 6.1 installation or profile is using.

The enablejvm tool has a a single required parameter, -jvm, which specifies the JVM to use. For IBM Technology for Java 32-bit, specify –jvm std32. To switch back to the classic i5/OS 64-bit JVM, specify –jvm classic.

To change the JVM for a single profile, use the –profile parameter to indicate which profile you want to change. For example, to change the JVM used for profile myprofile32 to be the new 32-bit JVM, enter enablejvm –jvm std32 –profile myprofile32 on the Qshell command line.

If you do not specify the –profile parameter, all existing profiles for the WebSphere Application Server installation are changed to use the specified JVM. When a new profile is created, it is configured to use the new JVM.

After invoking enablejvm, be sure to restart any servers that were active for any profile that was modified.



IBM Software Group

Network deployment considerations

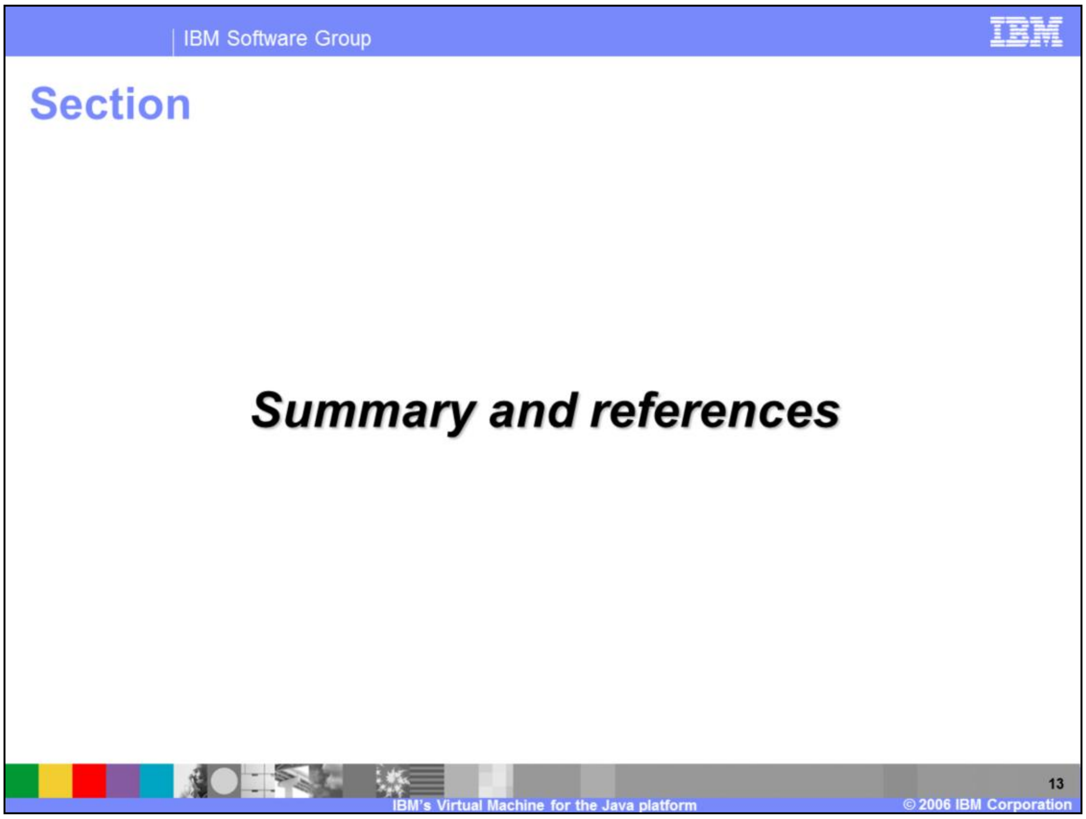
- Enabling the JVM for a deployment manager profile
  - Enables deployment manager server only
  - Does not affect nodes managed by the deployment manger
- Enabling the JVM for profiles that are part of a cell
  - Deployment manager server must be active
  - Run command from managed node profile's install_root/bin directory
  - If security is enabled, ensure soap.client.props file for node contains administrative user and password
- Documentation for enabling JVM:
  - http://www-1.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=enablejvm&uid=swg21240962&loc=en_US&cs=utf-8&lang=en

12

IBM's Virtual Machine for the Java platform © 2006 IBM Corporation

There are special considerations when enabling which JVM to use for a deployment manager or managed node which is part of a Network Deployment cell. When you enable the JVM for a deployment manager cell, only the deployment manager server is affected. None of the nodes managed by the deployment manager are changed.

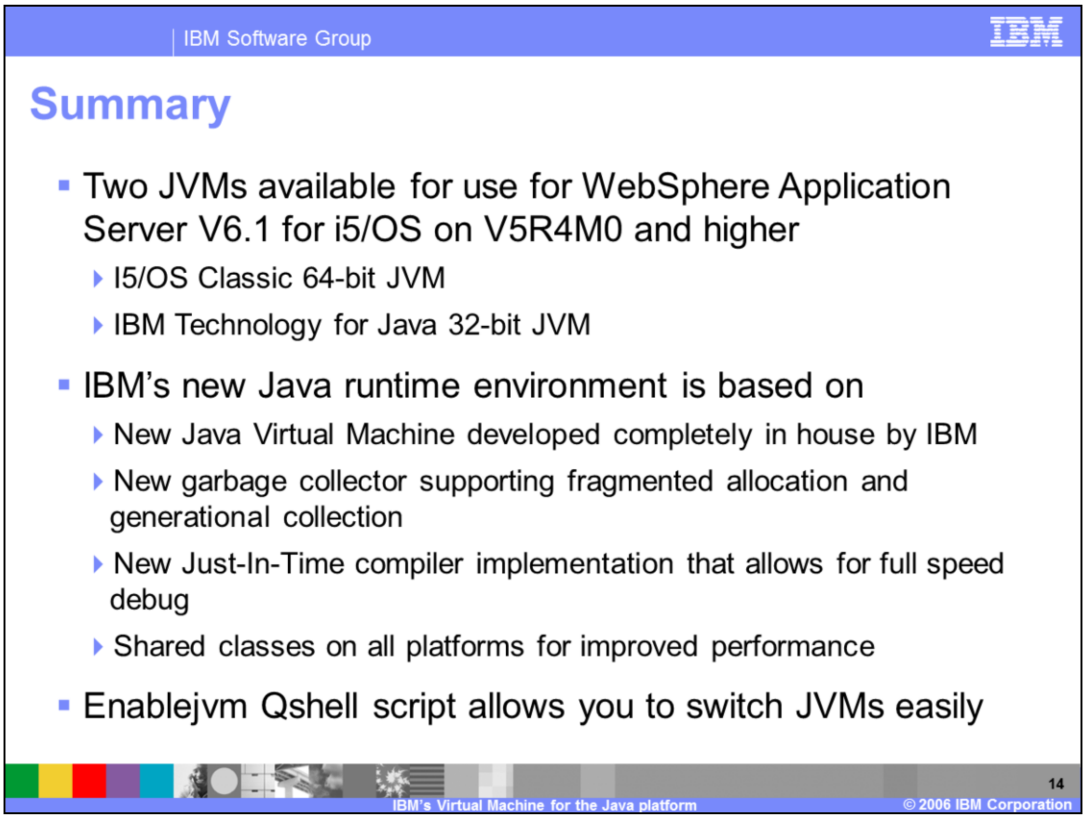
To enable the JVM for a node belonging to a Network Deployment cell, invoke the enablejvm Qshell script from the bin directory of the installation containing the managed node. The deployment manager for the node must be running when you invoke the command. The tool connects to the deployment manager to ensure that the master copy of the nodes configuration files are updated correctly. If administrative security is enabled for the deployment manager, you must specify the administrative user name and password in the soap.client.props file, located in the properties directory of the profile, for the profile you are enabling. Ensure that the com.ibm.SOAP.loginUserid and com.ibm.SOAP.loginPassword properties are set correctly prior to enabling the JVM for a profile that is part of a secure Network Deployment cell.



Section

Summary and references

The last portion of the presentation contains a summary and references.

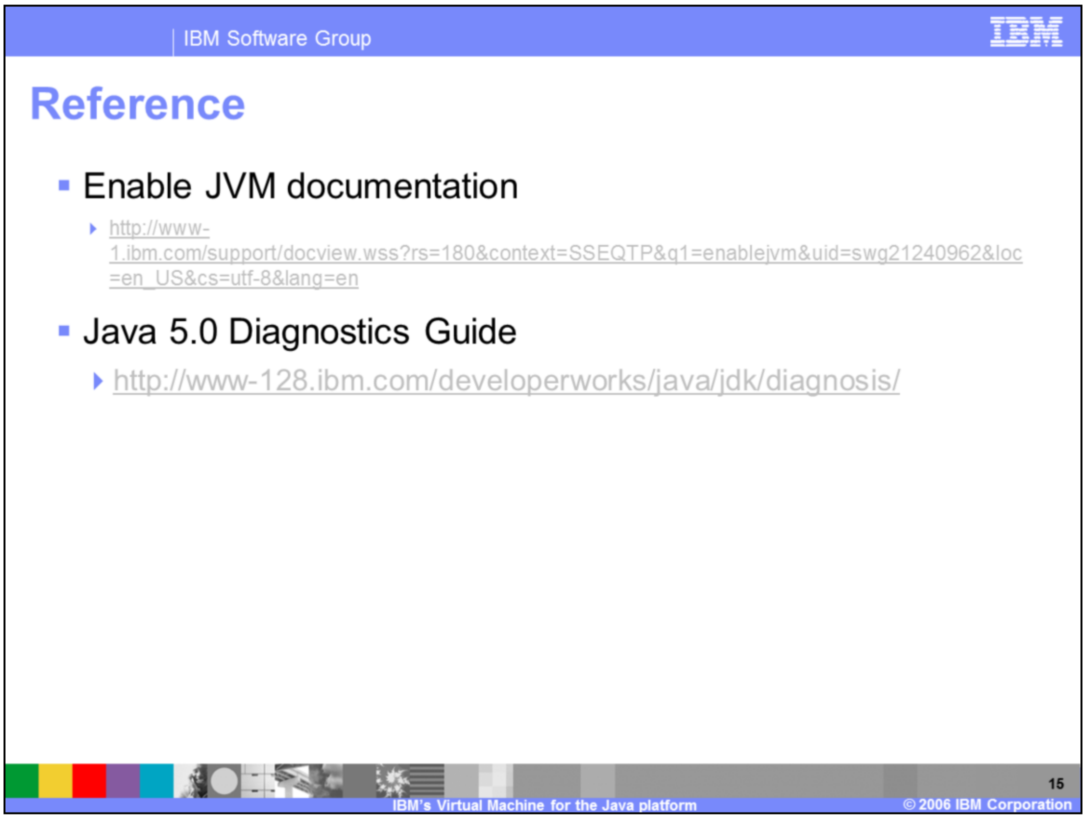


Summary

- Two JVMs available for use for WebSphere Application Server V6.1 for i5/OS on V5R4M0 and higher
  - I5/OS Classic 64-bit JVM
  - IBM Technology for Java 32-bit JVM
- IBM's new Java runtime environment is based on
  - New Java Virtual Machine developed completely in house by IBM
  - New garbage collector supporting fragmented allocation and generational collection
  - New Just-In-Time compiler implementation that allows for full speed debug
  - Shared classes on all platforms for improved performance
- Enablejvm Qshell script allows you to switch JVMs easily

WebSphere Application Server Version 6.1 can run on top of either the i5/OS Classic 64-bit JVM or the new IBM Technology for Java 32-bit JVM when running on i5/OS release V5R4M0 or higher. This new 32-bit JVM uses 40% less memory than the 64-bit JVM. The new Just-In-Time compiler offers new optimization levels and support for full speed debug. The addition of a shared class cache enables faster start-up time and substantial memory savings.

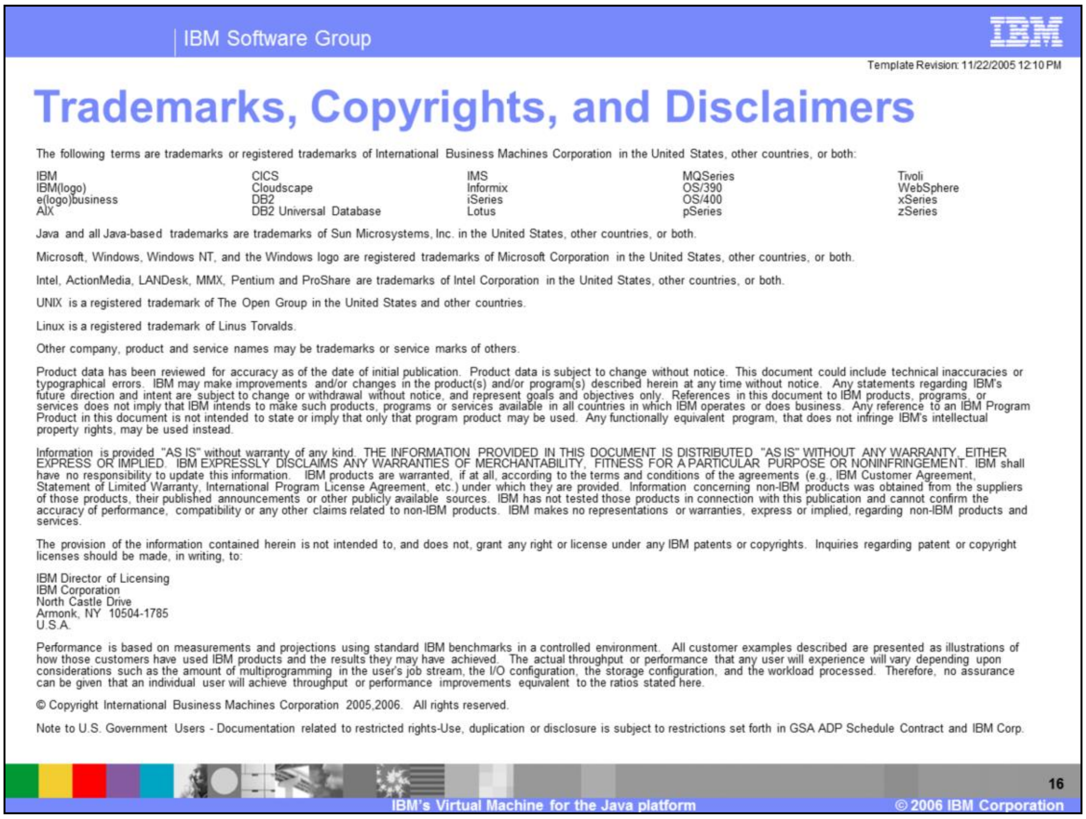
The enablejvm Qshell script allows you to switch JVMs quickly and easily.



Reference

- Enable JVM documentation
  - http://www-1.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=enablejvm&uid=swg21240962&loc=en_US&cs=utf-8&lang=en
- Java 5.0 Diagnostics Guide
  - http://www-128.ibm.com/developerworks/java/jdk/diagnosis/

The enable JVM documentation contains details on how to use the enablejvm tool. The Java 5.0 Diagnostics Guide contains additional information on the architecture of IBM Technology for Java 32-bit JVM.



Template Revision: 11/22/2005 12:10 PM

Trademarks, Copyrights, and Disclaimers