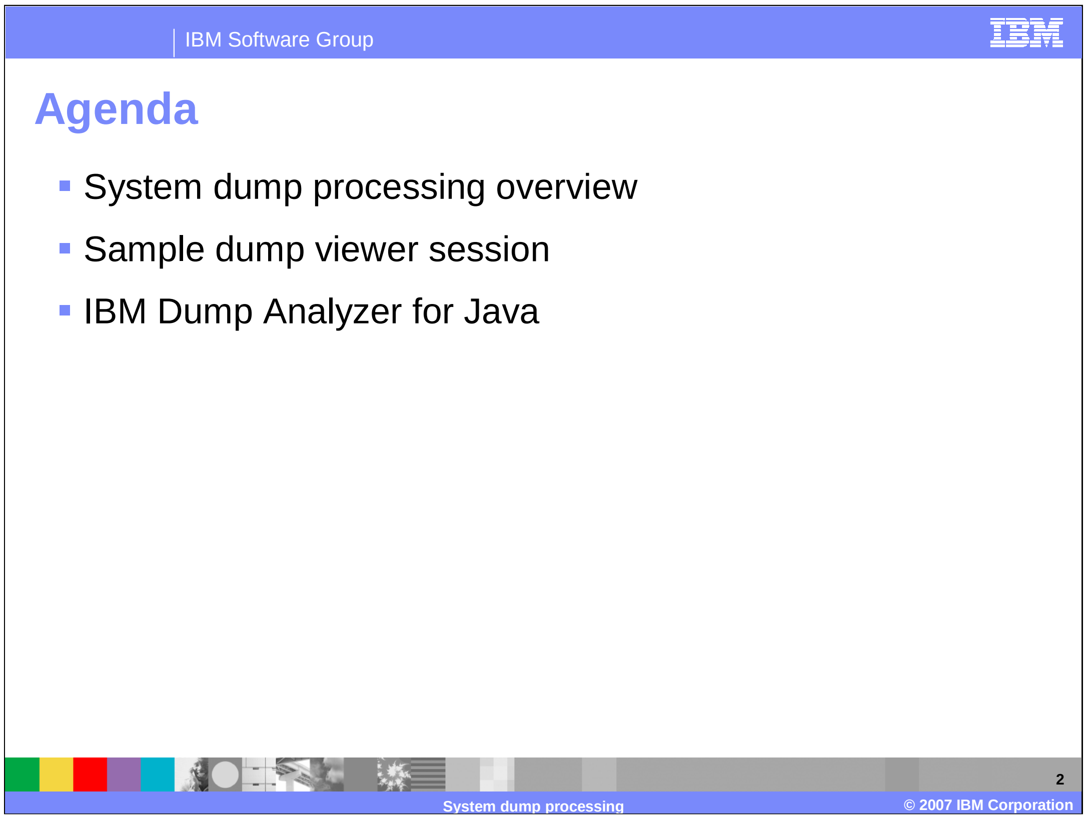
IBM Software Group

IBM® SDK, Java™ Technology Edition, V6

System dump processing

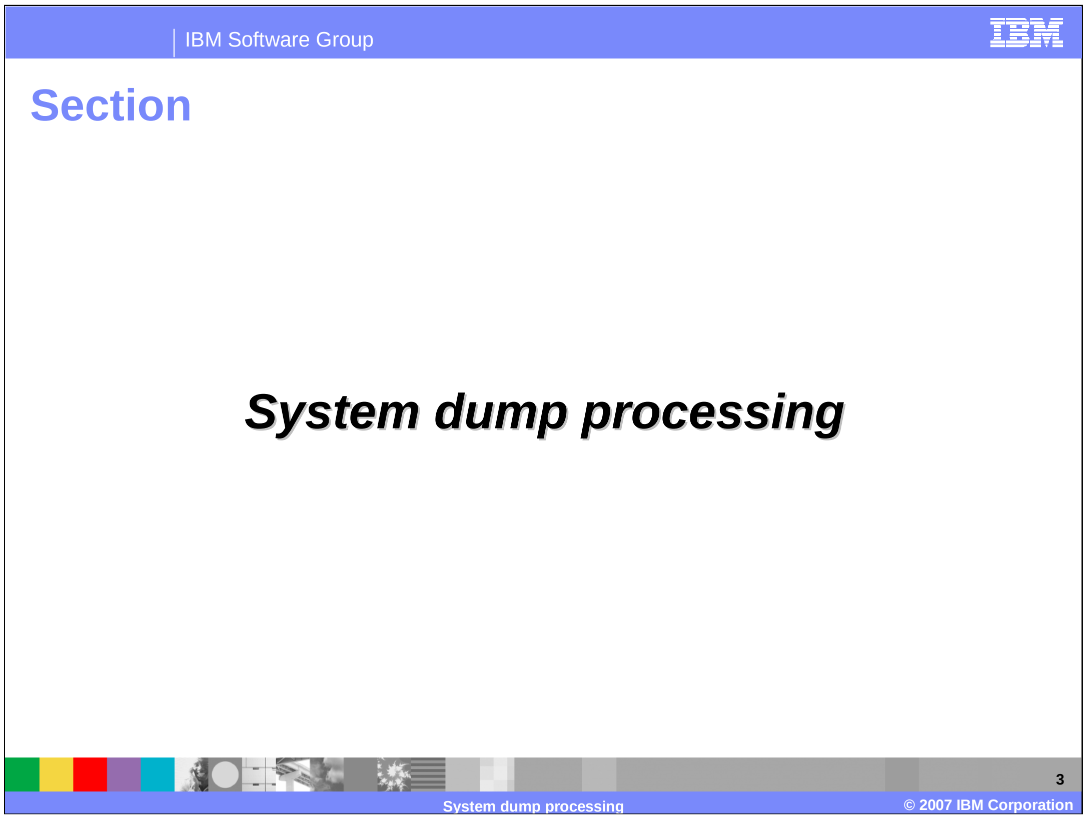
System dumps are debugging files produced by the virtual machine to help diagnose problems. This presentation will provide an overview of system dumps and tools available for processing system dumps.



Agenda

- System dump processing overview
- Sample dump viewer session
- IBM Dump Analyzer for Java

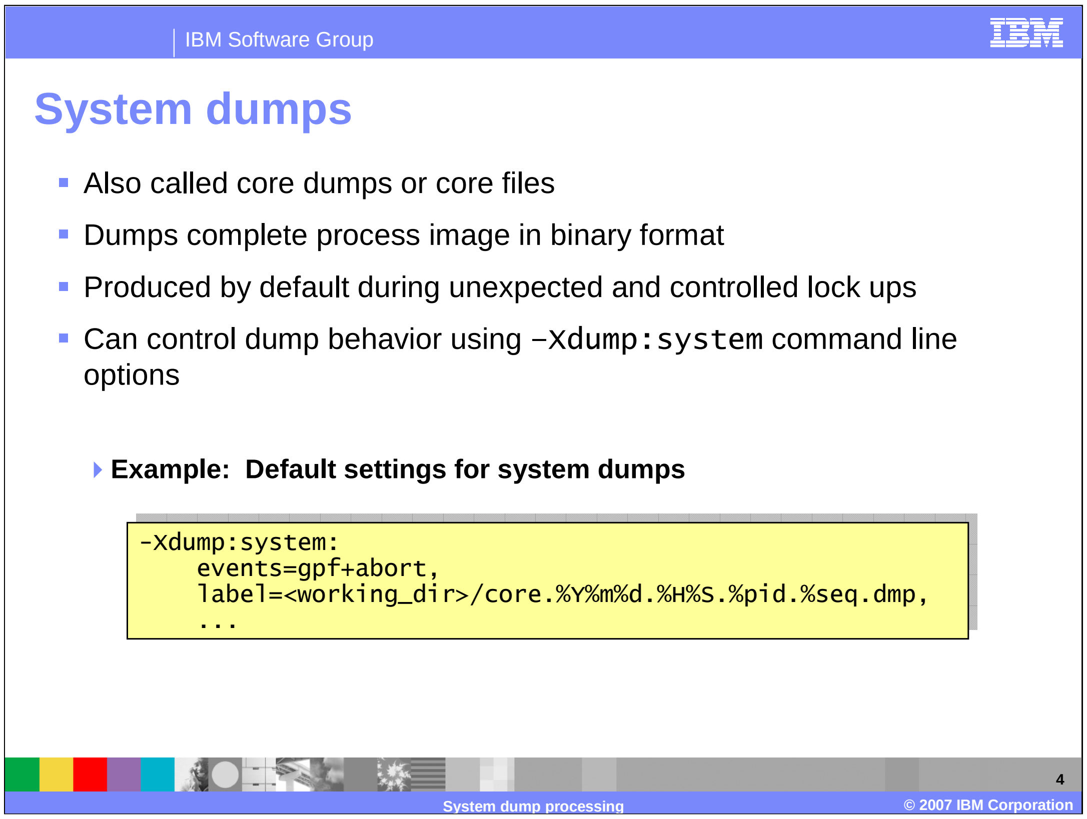
This presentation will cover the basics of what system dumps are, when they get produced, and some tools for processing them, including the command-line based dump viewer that is packaged with the IBM SDK for Java, and also a separate graphical system dump processing tool called the IBM Dump Analyzer for Java.



Section

System dump processing

This section will provide an overview of system dumps and system dump processing.



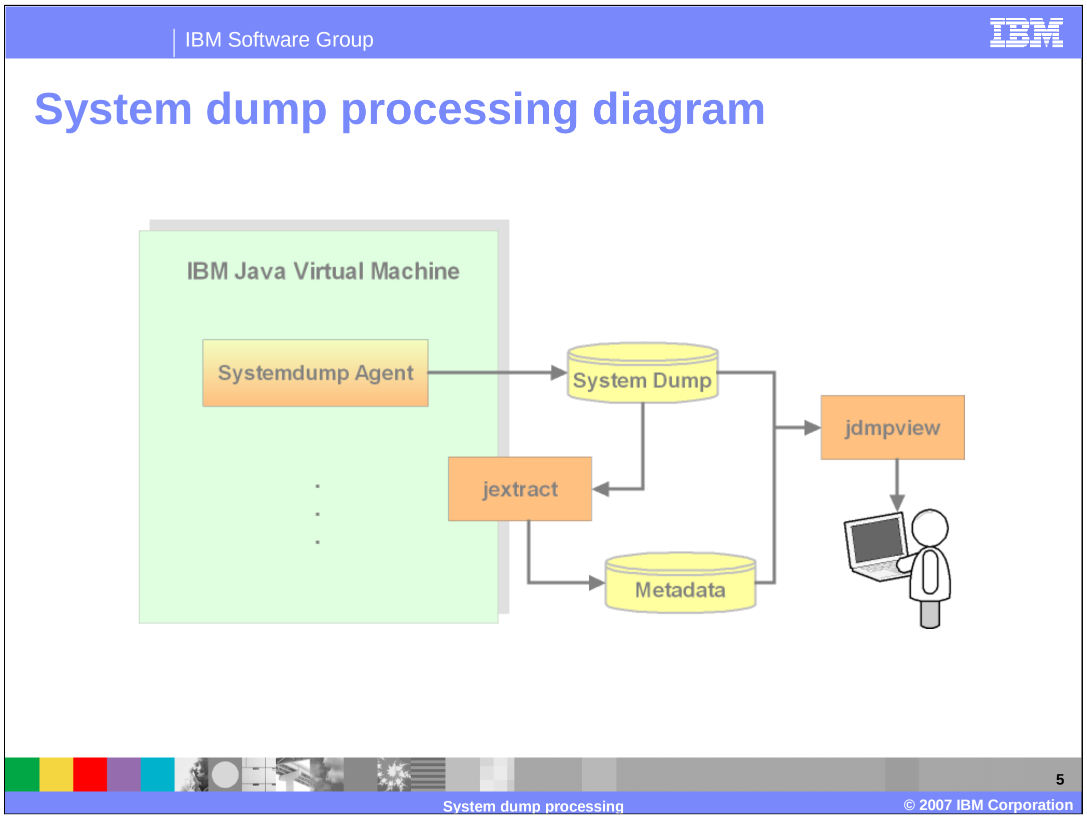
System dumps

- Also called core dumps or core files
- Dumps complete process image in binary format
- Produced by default during unexpected and controlled lock ups
- Can control dump behavior using `–Xdump:system` command line options

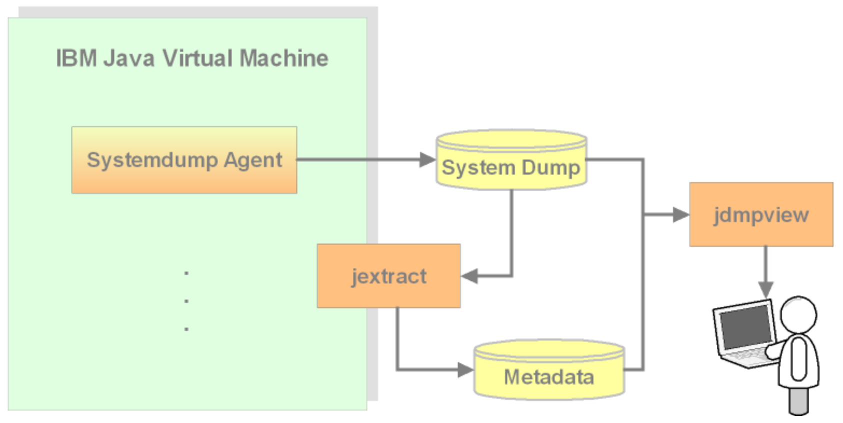
  - **Example: Default settings for system dumps**

```
-Xdump:system:
    events=gpf+abort,
    label=<working_dir>/core.%Y%m%d.%H%S.%pid.%seq.dmp,
    ...
```

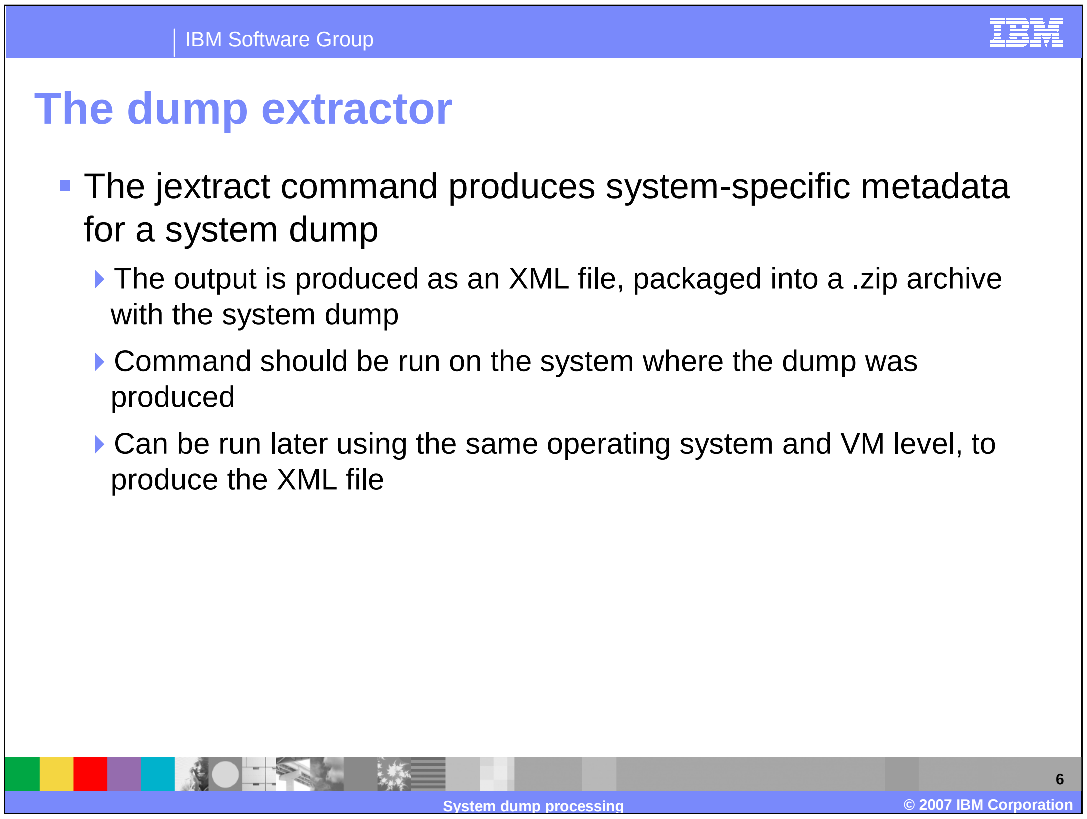
A system dump, sometimes called a core dump or a core file, consists of all the memory that is being used by the JVM. This includes the application heap, along with all JVM and user libraries. The dump contains a complete image of the JVM process in binary format. The bigger the footprint of an application, the bigger its dump. A dump of a major server-based application might take up many gigabytes of file space and take several minutes to complete. System dumps are produced by default whenever a JVM stops and are useful for helping to determine the cause of a lock up. You can also configure your Java environment to produce system dumps in response to other events, like when a user signal occurs or when the virtual machine throws a particular type of exception. Dump agents are configured using –Xdump command line options, like in the example on this slide. This example shows some of the default settings for when and how system dumps will be created. Notice that the default system dump agent monitors for gpf and abort events, and that the system dump file that is produced is customized to contain current date, time, and process information. If you ever want to check the default dump options and dump file locations for your Java environment, you can launch a JVM with the option –Xdump:what to display the current default dump behavior.



System dump processing diagram



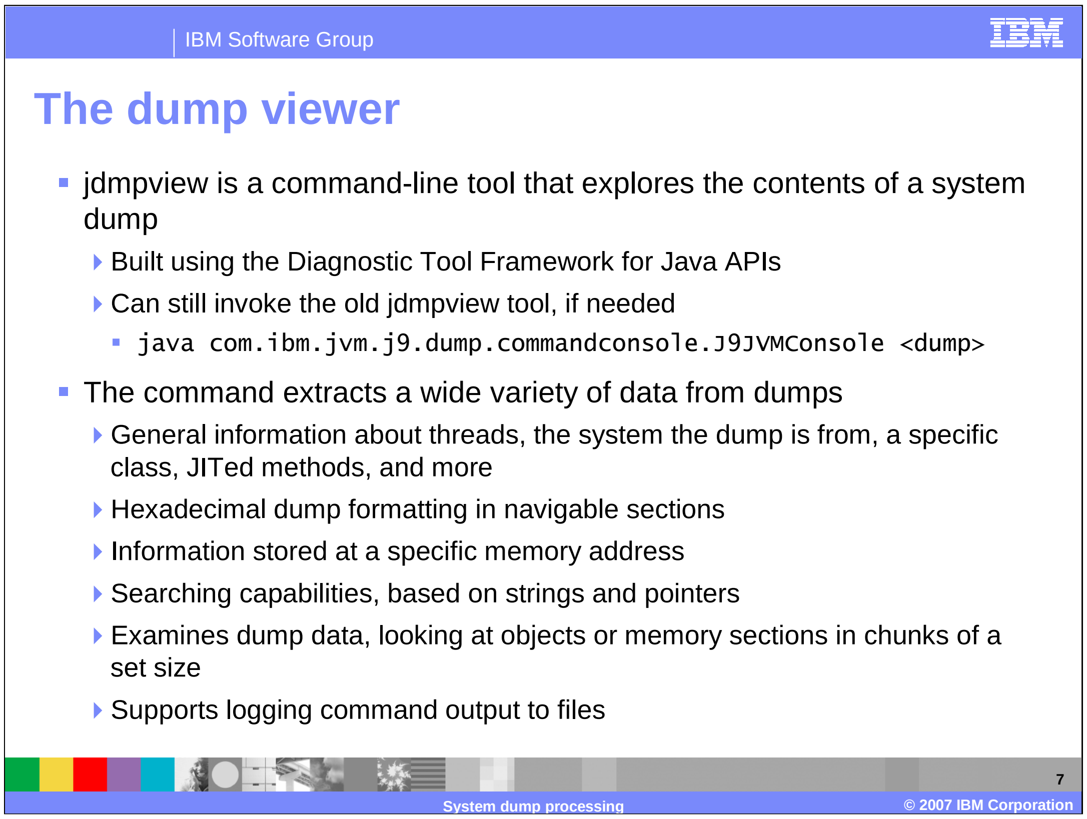
This diagram illustrates the system dump processing flow, using built-in components and commands in the IBM SDK. A system dump agent that is configured to monitor for certain events – like when a JVM has a general protection fault – will trigger a dump when those events occur. A system dump file should be fed into the dump extractor, for pre-processing before you try to analyze your dump with any other tools. The dump extractor which will pull out useful metadata about your Java operating environment and create an archive that contains the metadata and your system dump file, which you can then use in a system dump processing tool.



The dump extractor

- The jextract command produces system-specific metadata for a system dump
  - The output is produced as an XML file, packaged into a .zip archive with the system dump
  - Command should be run on the system where the dump was produced
  - Can be run later using the same operating system and VM level, to produce the XML file

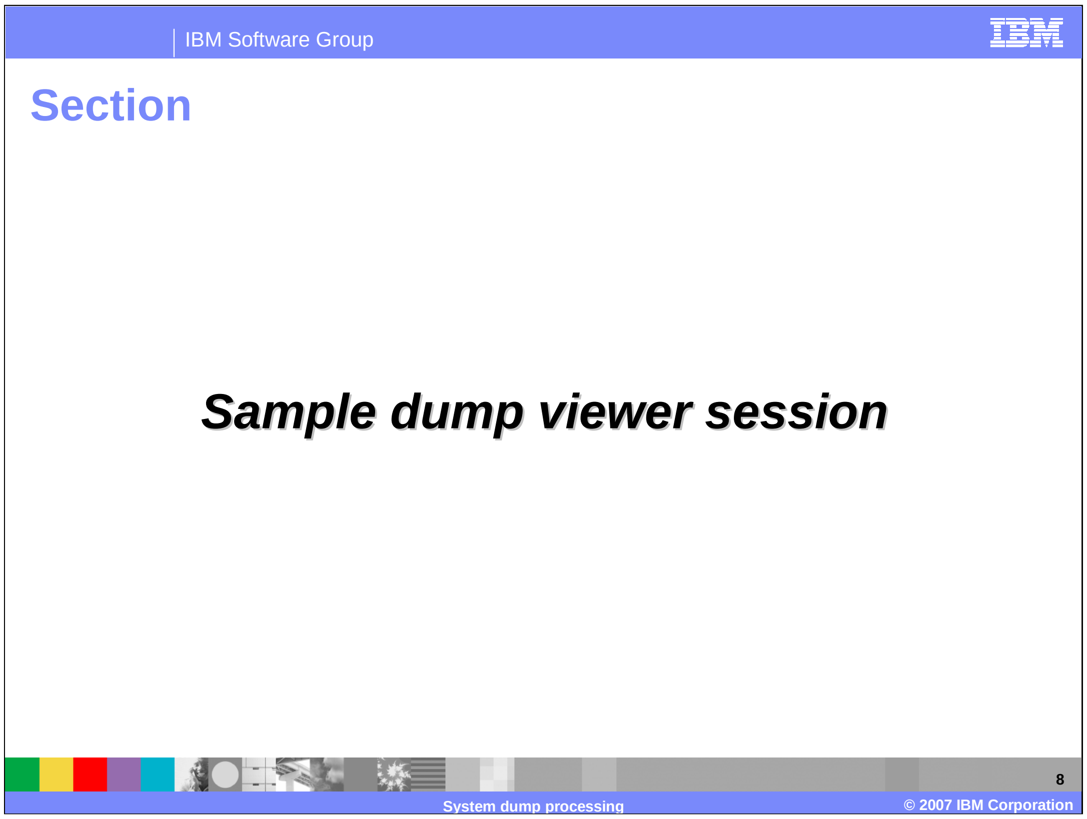
The jextract tool obtains platform specific information such as word size, endianness, data structure layouts, and symbolic information. It puts this information into an XML file. jextract also collects other useful files, depending on the platform, including trace files and copies of executable files and libraries and by default, compresses these into a single .zip archive for use in subsequent problem diagnosis. The jextract tool must be run on the same platform and the same JVM level (ideally the same machine) that was being used when the dump was produced. The combination of the dump file and the XML file produced by jextract allows the dump viewer and other tools to analyze and display Java information. The jextract tool is packaged in the sdk/jre/bin directory.



The dump viewer

- jdmpview is a command-line tool that explores the contents of a system dump
  - Built using the Diagnostic Tool Framework for Java APIs
  - Can still invoke the old jdmpview tool, if needed
    - `java com.ibm.jvm.j9.dump.commandconsole.J9JVMConsole <dump>`
- The command extracts a wide variety of data from dumps
  - General information about threads, the system the dump is from, a specific class, JITed methods, and more
  - Hexadecimal dump formatting in navigable sections
  - Information stored at a specific memory address
  - Searching capabilities, based on strings and pointers
  - Examines dump data, looking at objects or memory sections in chunks of a set size
  - Supports logging command output to files

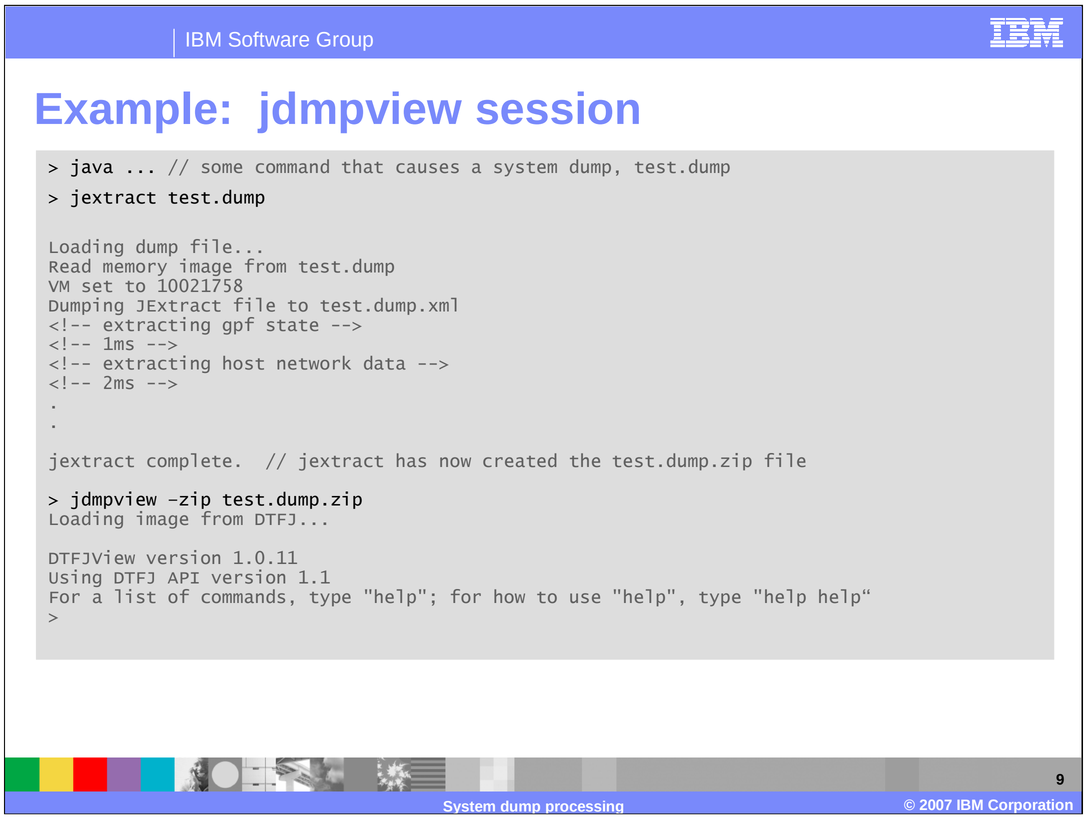
The dump viewer is a cross-platform tool that allows you to examine the contents of system dumps produced from the JVM. To be able to analyze platform-specific dumps, the dump viewer requires metadata created by the jextract tool. The dump viewer allows you to view both Java and native information from the time the dump was produced. You can explore information about the system on which the dump was produced, environment variables, threads and registers, heap contents, Just-in-time compiled methods, and other data. You can think of the dump viewer like a command-line debugger for analyzing system dumps. While previous releases of the IBM SDK also provided a dump viewer, the version of the dump viewer available with the IBM SDK for Java Version 6 has been updated to use the Diagnostic Tool Framework for Java. The current viewer is packaged in sdk/jre/bin. If you need to access the dump viewer from the previous release, you can invoke the Java program directly using the syntax shown on the slide.



Section

Sample dump viewer session

This section contains an example session of the dump viewer that illustrates a selection of the commands available and their use. To simplify the display, some of the contents of the terminal have been replaced by ellipses.



Example: jdmpview session

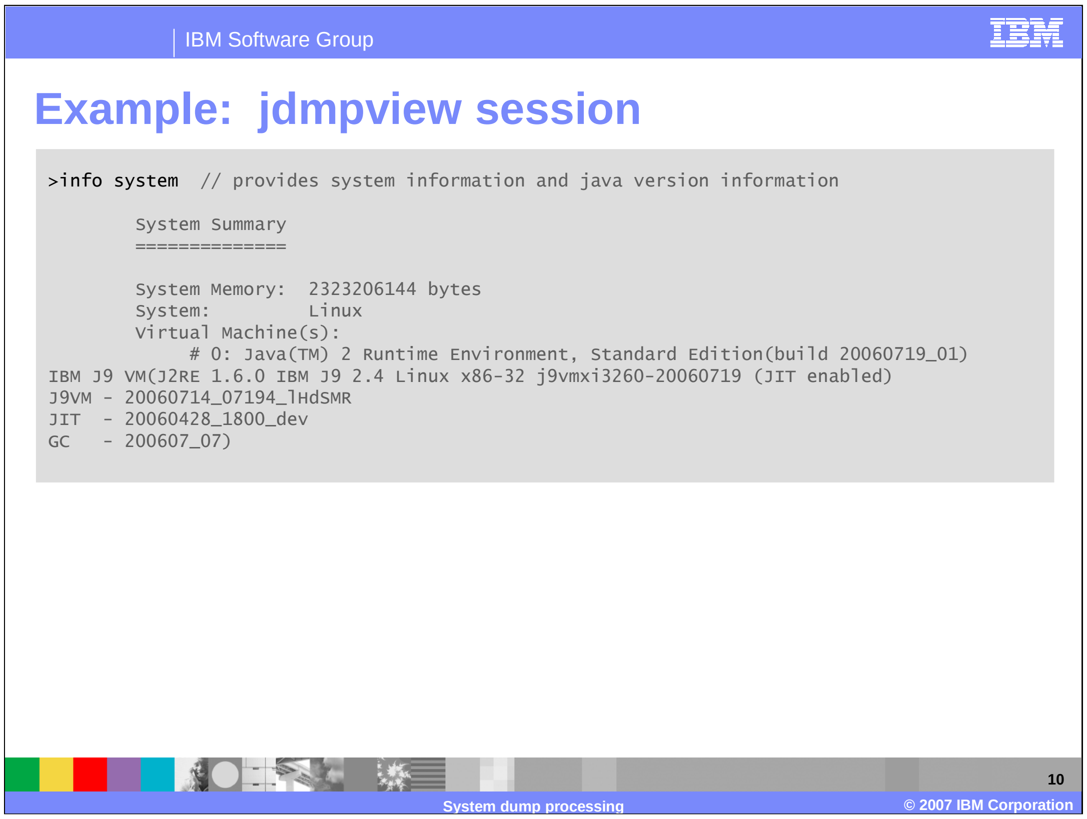
```
> java ... // some command that causes a system dump, test.dump
> jextract test.dump

Loading dump file...
Read memory image from test.dump
VM set to 10021758
Dumping JExtract file to test.dump.xml
<!-- extracting gpf state -->
<!-- 1ms -->
<!-- extracting host network data -->
<!-- 2ms -->
.
.
jextract complete.  // jextract has now created the test.dump.zip file

> jdmpview -zip test.dump.zip
Loading image from DTFJ...

DTFJView version 1.0.11
Using DTFJ API version 1.1
For a list of commands, type "help"; for how to use "help", type "help help“
>
```

Before starting the dump viewer, you need to run your system dump file through the dump extractor. You can use the resulting .zip archive as input to the dump viewer. Invoke the dump viewer using the command jdmpview. The dump viewer is ready to process commands when the prompt appears in your terminal. Use the dump viewer to help to see a list of all of the commands that are available. Comprehensive documentation of all of the commands is also available in the Diagnostics Guide.



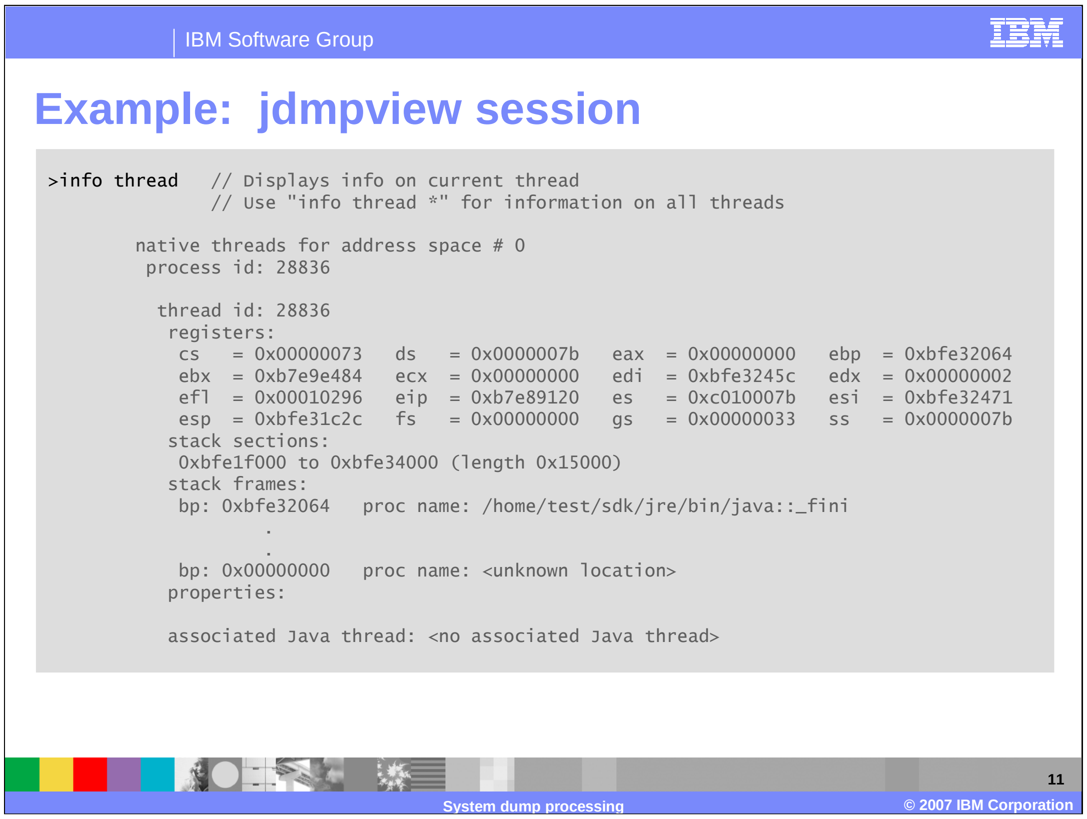
Example: jdmpview session

```
>info system  // provides system information and java version information

        System Summary
        ==============

        System Memory:  2323206144 bytes
        System:         Linux
        Virtual Machine(s):
             # 0: Java(TM) 2 Runtime Environment, Standard Edition(build 20060719_01)
IBM J9 VM(J2RE 1.6.0 IBM J9 2.4 Linux x86-32 j9vmxi3260-20060719 (JIT enabled)
J9VM - 20060714_07194_lHdSMR
JIT  - 20060428_1800_dev
GC   - 200607_07)
```

The “info system” command will display the system memory, operating system, and virtual machine version. It is a good idea to run the “info system” command early on to do a sanity check, and to make sure that you are looking at a dump from the system you expect.



Example: jdmpview session

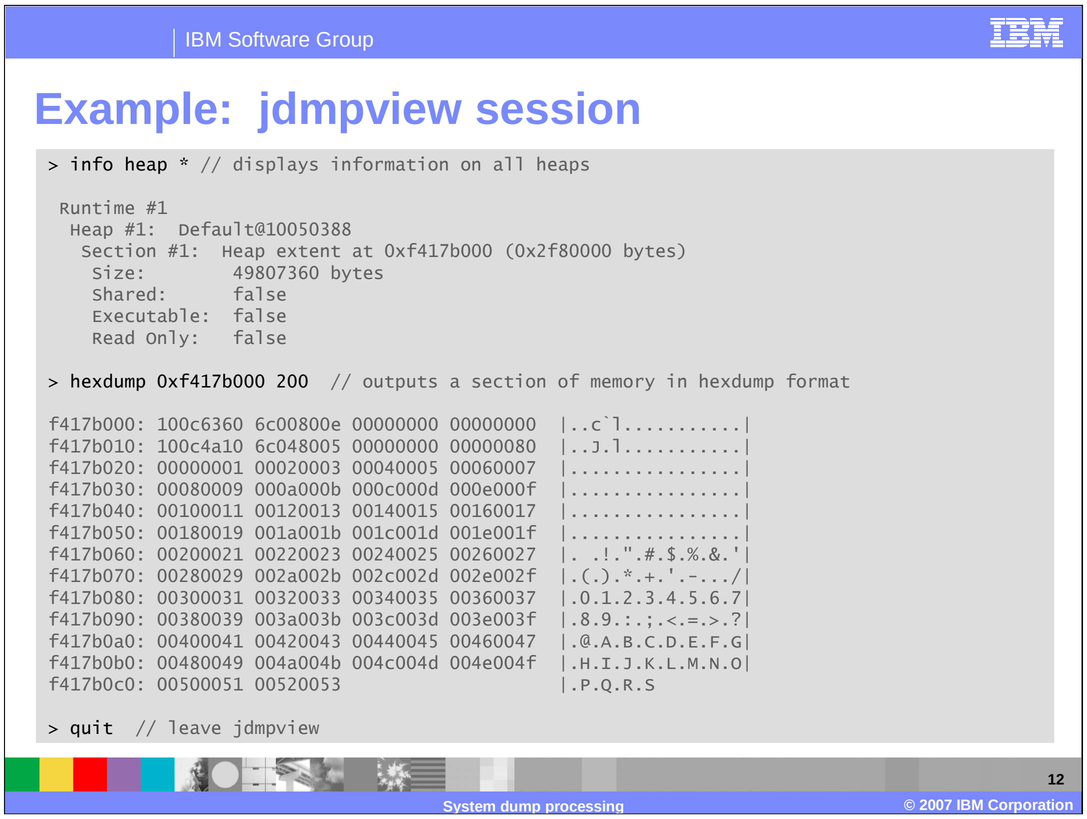
```
>info thread   // Displays info on current thread
               // Use "info thread *" for information on all threads

        native threads for address space # 0
         process id: 28836

          thread id: 28836
           registers:
            cs    = 0x00000073   ds    = 0x0000007b   eax   = 0x00000000   ebp   = 0xbfe32064
            ebx   = 0xb7e9e484   ecx   = 0x00000000   edi   = 0xbfe3245c   edx   = 0x00000002
            efl   = 0x00010296   eip   = 0xb7e89120   es    = 0xc010007b   esi   = 0xbfe32471
            esp   = 0xbfe31c2c   fs    = 0x00000000   gs    = 0x00000033   ss    = 0x0000007b
           stack sections:
            0xbfe1f000 to 0xbfe34000 (length 0x15000)
           stack frames:
            bp: 0xbfe32064   proc name: /home/test/sdk/jre/bin/java::_fini
                    .
                    .
            bp: 0x00000000   proc name: <unknown location>
           properties:

           associated Java thread: <no associated Java thread>
```

The “info thread” command will display information about the current thread. You can also display data about all of the threads in the process by using the command “info thread *.” The dump viewer provides data on native threads and Java threads.



IBM Software Group

Example: jdmpview session

```
> info heap * // displays information on all heaps

 Runtime #1
  Heap #1:  Default@10050388
   Section #1:  Heap extent at 0xf417b000 (0x2f80000 bytes)
    Size:        49807360 bytes
    Shared:      false
    Executable:  false
    Read Only:   false

> hexdump 0xf417b000 200  // outputs a section of memory in hexdump format

f417b000: 100c6360 6c00800e 00000000 00000000  |..c`l...........|
f417b010: 100c4a10 6c048005 00000000 00000080  |..J.l...........|
f417b020: 00000001 00020003 00040005 00060007  |................|
f417b030: 00080009 000a000b 000c000d 000e000f  |................|
f417b040: 00100011 00120013 00140015 00160017  |................|
f417b050: 00180019 001a001b 001c001d 001e001f  |................|
f417b060: 00200021 00220023 00240025 00260027  |. .!.".#.$.%.&.'|
f417b070: 00280029 002a002b 002c002d 002e002f  |.(.).*.+.'.-.../|
f417b080: 00300031 00320033 00340035 00360037  |.0.1.2.3.4.5.6.7|
f417b090: 00380039 003a003b 003c003d 003e003f  |.8.9.:.;.<.=.>.?|
f417b0a0: 00400041 00420043 00440045 00460047  |.@.A.B.C.D.E.F.G|
f417b0b0: 00480049 004a004b 004c004d 004e004f  |.H.I.J.K.L.M.N.O|
f417b0c0: 00500051 00520053                    |.P.Q.R.S

> quit  // leave jdmpview
```

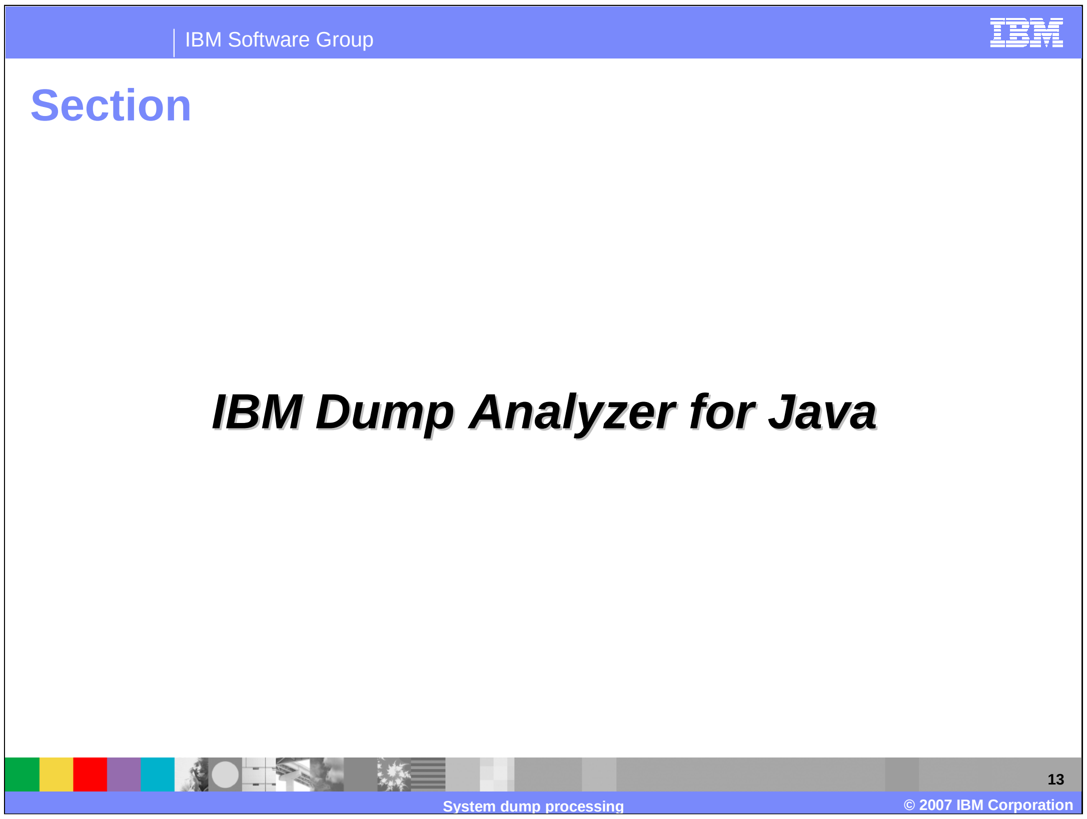
12

System dump processing

© 2007 IBM Corporation

Using the “info heap *” command will display all of the heap spaces associated with this JVM process. In this case, there is a default heap, and its name, size, location, and other attributes are shown. Once you know the memory location of the heap, you can use the “hexdump” command to dump the contents of the heap to the terminal display in hexadecimal format with corresponding ASCII values. In this example, the dump viewer is starting at the beginning of the default heap and displaying 200 bytes of data.

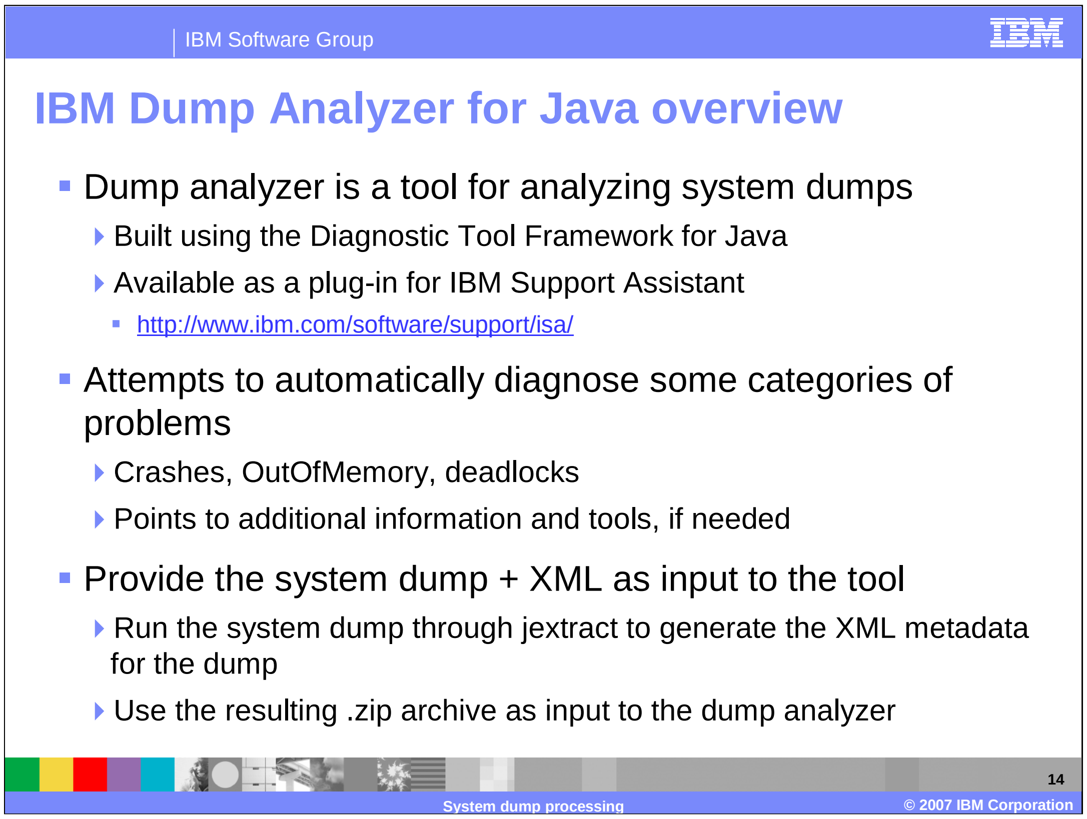
When you have finished using the dump viewer, leave the debugging environment using the “quit” command.



Section

IBM Dump Analyzer for Java

In addition to the dump viewer for processing system dumps, you can also use the IBM Dump Analyzer for Java to gain insight into the contents of your dump.


IBM Software Group

IBM Dump Analyzer for Java overview

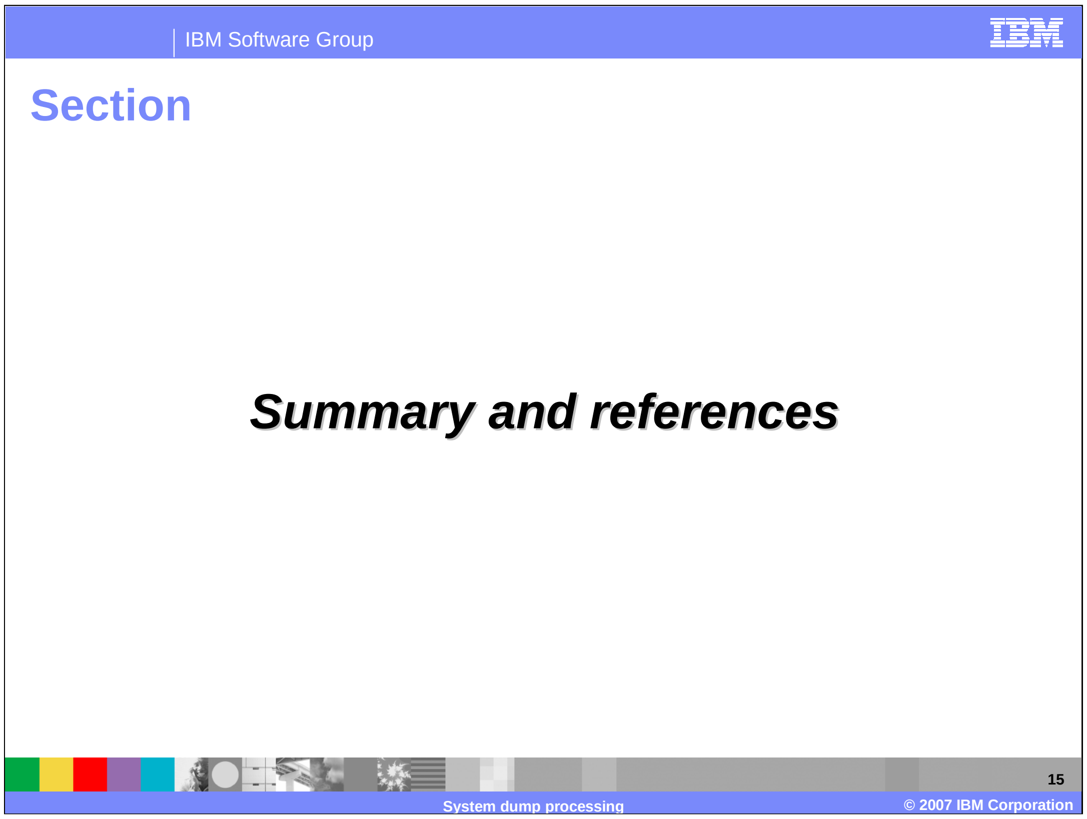
- Dump analyzer is a tool for analyzing system dumps
  - Built using the Diagnostic Tool Framework for Java
  - Available as a plug-in for IBM Support Assistant
    - http://www.ibm.com/software/support/isa/
- Attempts to automatically diagnose some categories of problems
  - Crashes, OutOfMemory, deadlocks
  - Points to additional information and tools, if needed
- Provide the system dump + XML as input to the tool
  - Run the system dump through jextract to generate the XML metadata for the dump
  - Use the resulting .zip archive as input to the dump analyzer

14

System dump processing

© 2007 IBM Corporation

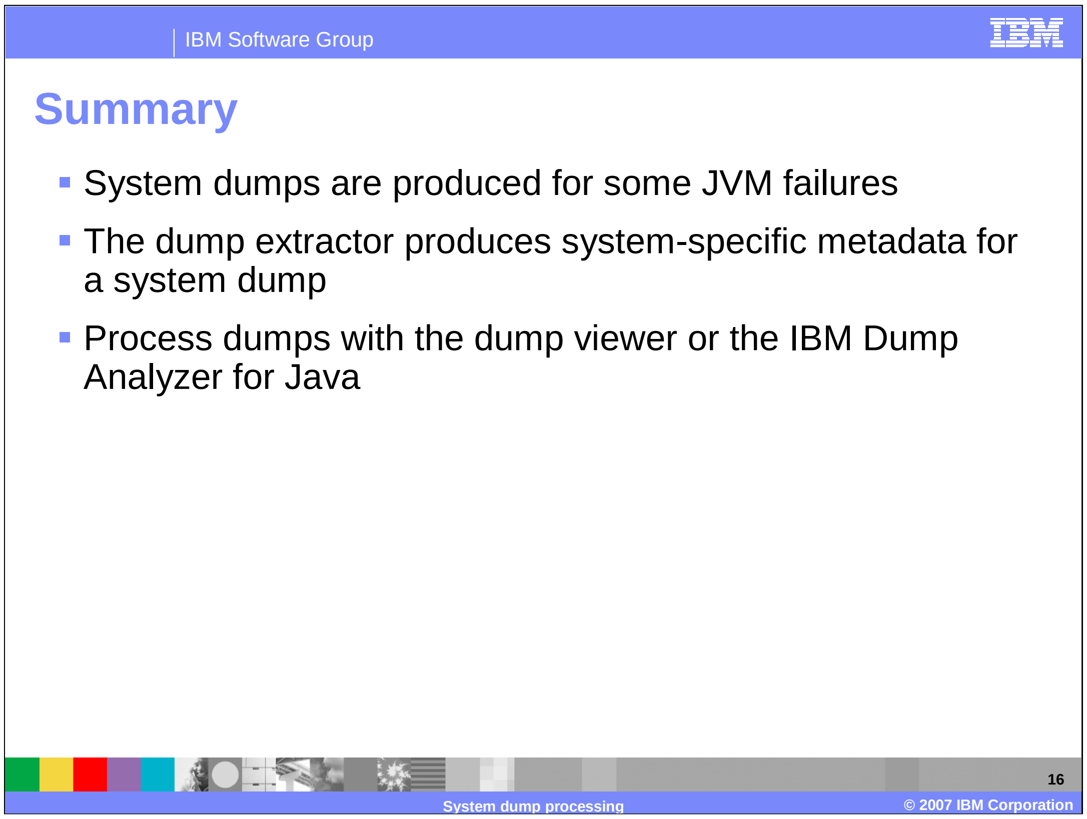
The IBM Dump Analyzer for Java is a graphical tool that is available as a free plug-in for IBM Support Assistant. Like the dump viewer, the IBM Dump Analyzer for Java is built on the Diagnostic Tool Framework for Java APIs and takes as input the .zip archive that was produced by the dump extractor. The dump analyzer is intended to perform automated analysis of system dump files produced by the IBM Virtual Machine for Java Platforms. The tool will use the information in the .zip archive input file to perform some automatic diagnosis, trying to identify the underlying cause of your problem and why the system dump was produced. The dump analyzer will offer you suggestions on how to overcome your issue, if possible, or it will point you to additional resources and tools that can help you solve your problem.



Section

Summary and references

This section contains a summary and reference.



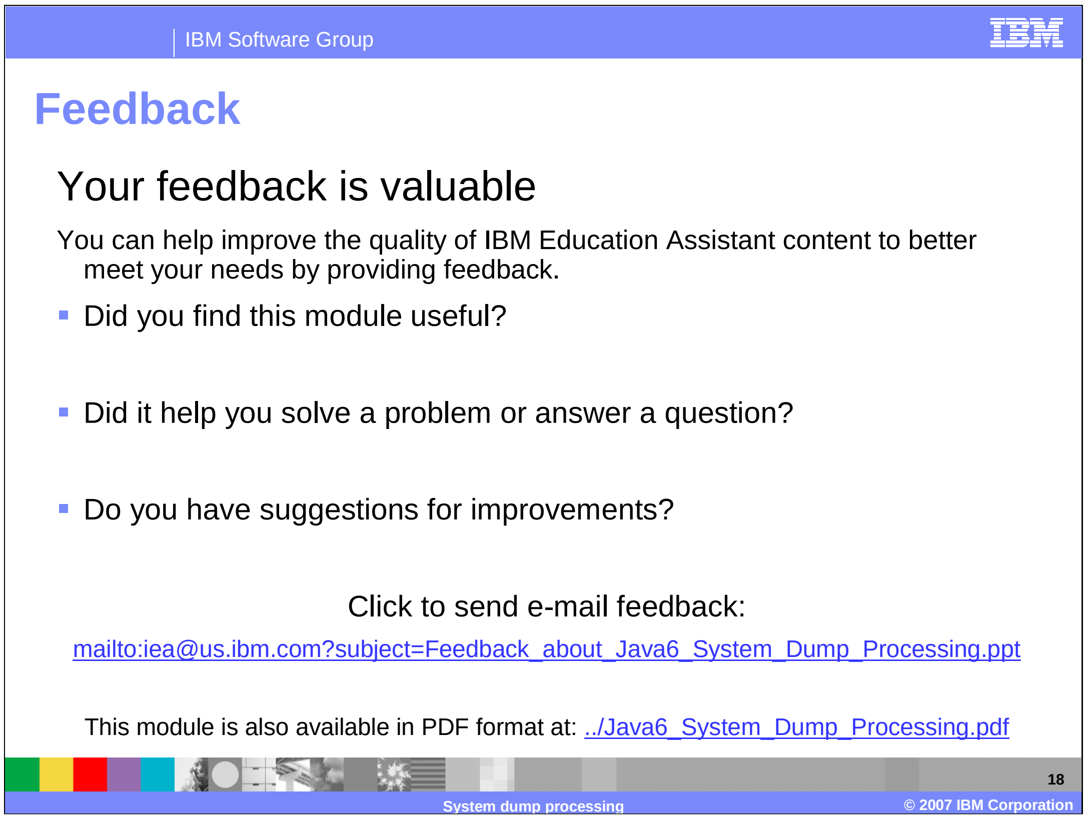
Summary

- System dumps are produced for some JVM failures
- The dump extractor produces system-specific metadata for a system dump
- Process dumps with the dump viewer or the IBM Dump Analyzer for Java

System dumps are important diagnostic artifacts, produced by the JVM when a general protection fault or a stop signal occurs. The dump contains a binary image of the JVM process, and should always be pre-processed by the dump extractor before using any other system dump analysis tools. There are two main tools that you can use to process system dumps. The command-line dump viewer, jdmpview, which is packaged with the SDK, and the IBM Dump Analyzer for Java, which is available as a free downloadable plug-in for IBM Support Assistant.

References

- IBM Support Assistant
  - http://www.ibm.com/software/support/isa/
- Article on the IBM Dump Analyzer for Java
  - http://www.ibm.com/developerworks/java/library/j-ibmtools1/
- 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_System_Dump_Processing.ppt

This module is also available in PDF format at: ../Java6_System_Dump_Processing.pdf

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

Trademarks, copyrights, and disclaimers