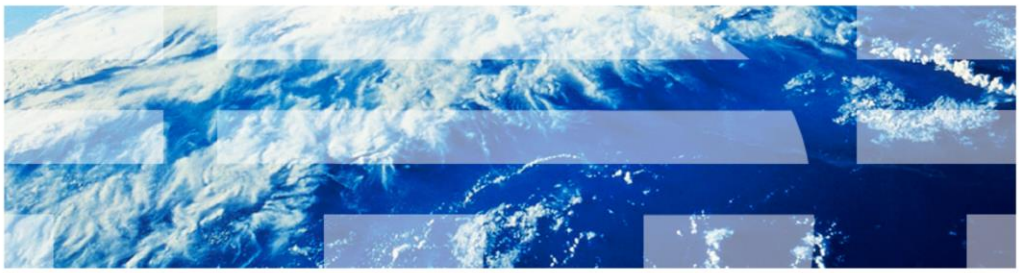
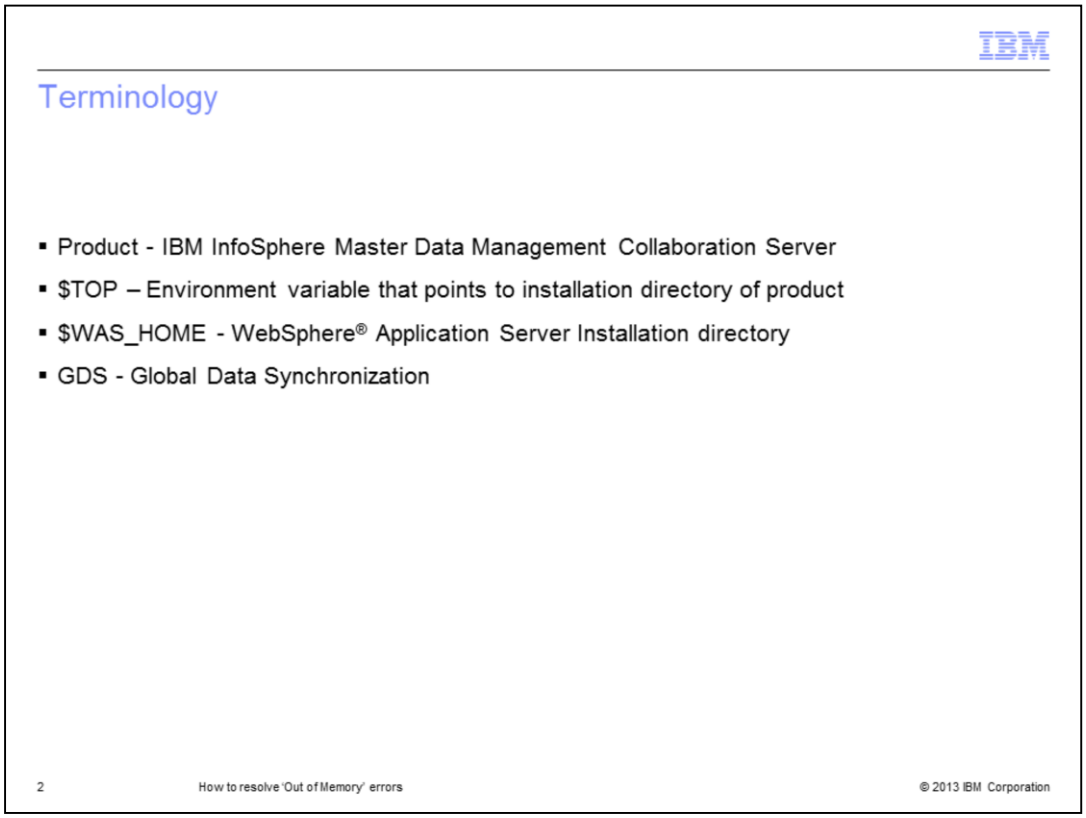
IBM

InfoSphere Master Data Management Collaboration Server

How to resolve 'Out of Memory' errors



This presentation covers what an 'Out of Memory' error is, factors that can contribute to this error, types of 'Out of Memory' errors and how to resolve them in the IBM InfoSphere® Master Data Management Collaboration server.

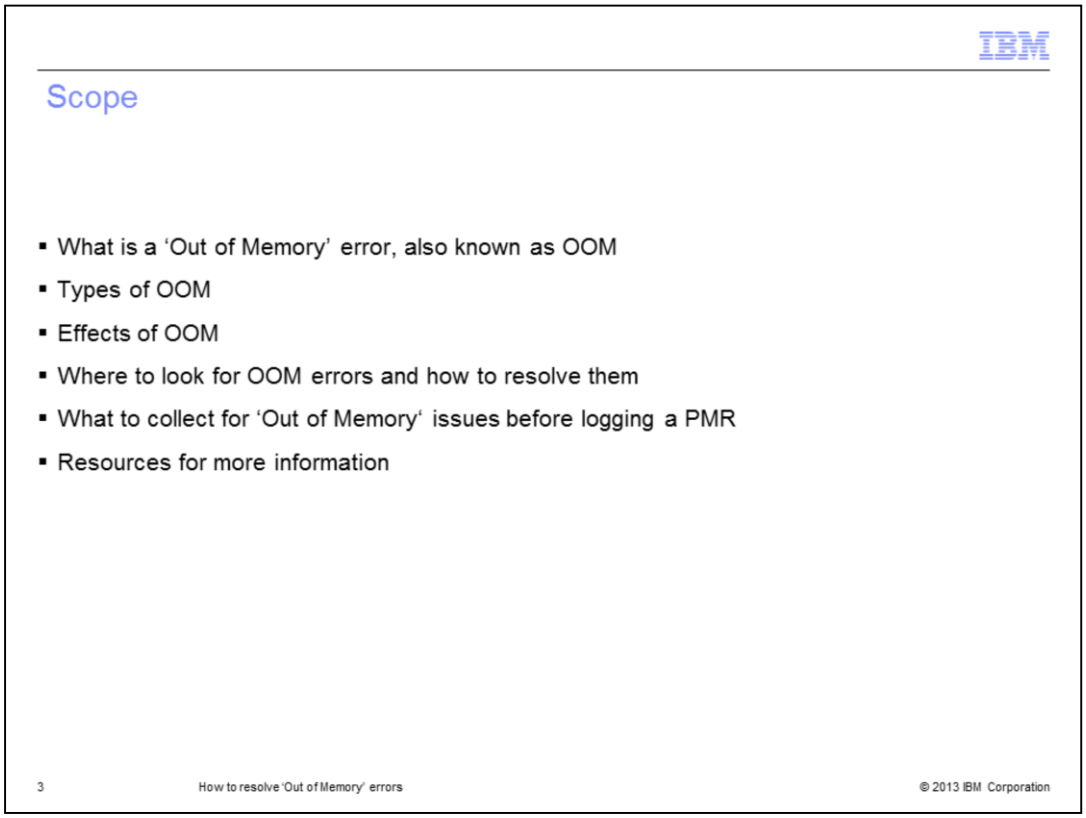


Before going into details, there is some terminology you need to be aware of. The official name of the product referenced in this presentation is IBM InfoSphere Master Data Management Collaboration Server, referred to as MDMCS.

The term $TOP is an environment variable that points to the installation directory of the product.

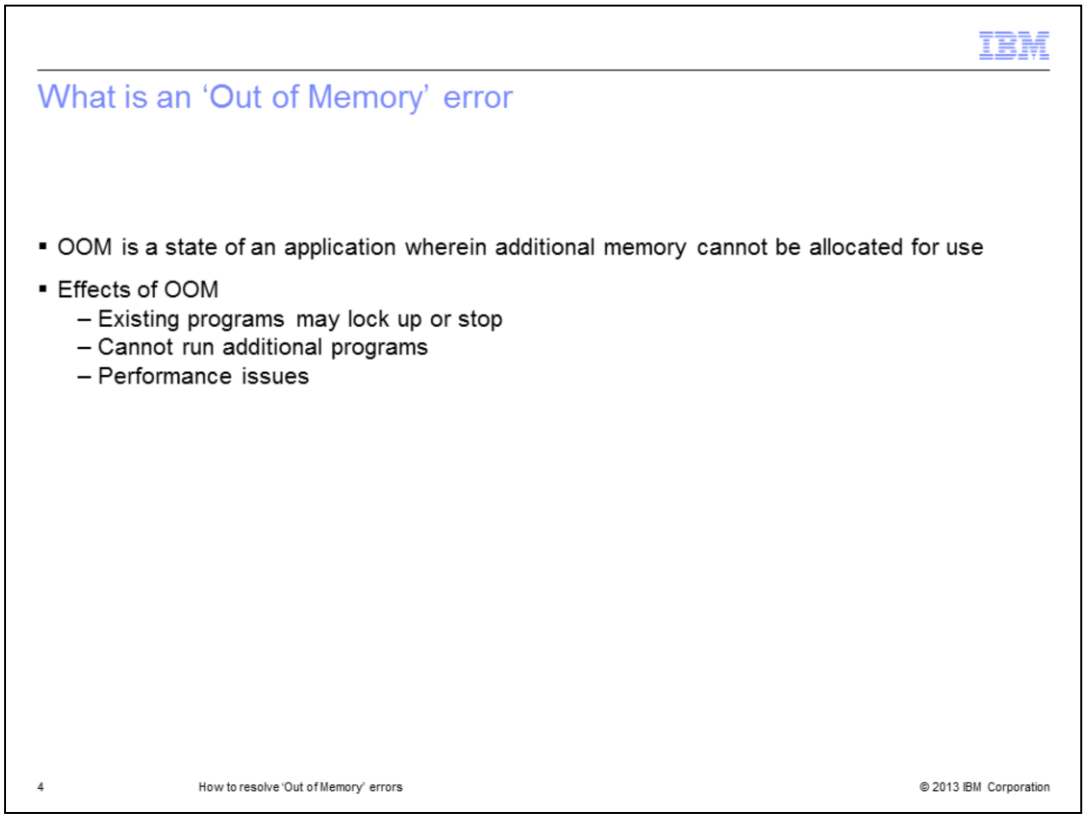
WAS_HOME is the directory where the WebSphere Application Server is installed.

GDS refers to MDMCS with Global Data Synchronization.



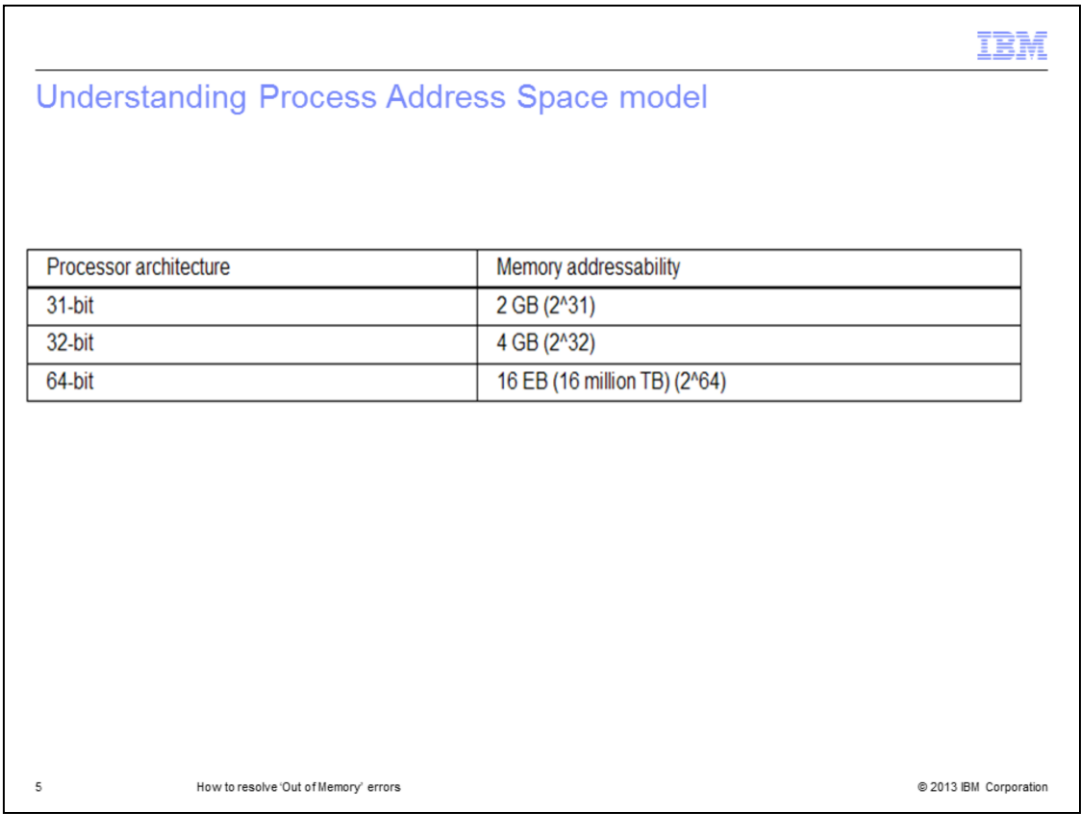
In this presentation you will learn what an 'Out of Memory' error is, also known as OOM error. You will understand the various types of OOM errors and what the effects are on the application. In addition, you will know which logs you need to look at to see the OOM error and what you can do to resolve them.

If you still have to log a PMR, you will need to know the diagnostic information you need to collect to help investigate the root cause of the issue. Finally, you will look into some Information Center links and Technotes which will help you understand the 'Out of Memory' error better and diagnose it further.



OOM is a state of an application wherein additional memory cannot be allocated for use to the existing program, a new program or the operating system. This occurs because all available memory, including disk swap space, has been allocated.

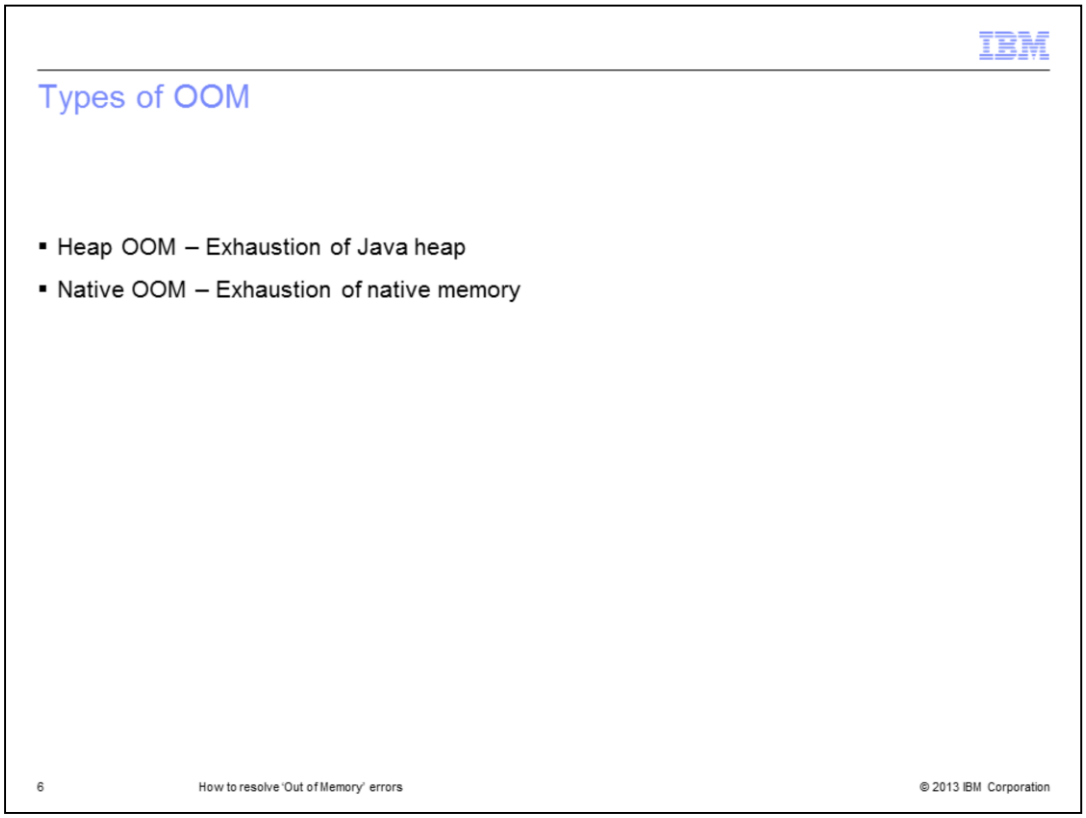
The effects of OOM include existing programs may be locked or ending abruptly due to lack of additional memory allocation. Another effect of OOM is if you cannot run additional programs as the new programs need memory to function correctly. Also, a lack of sufficient memory can cause performance issues. Finally, JVM may start to spend too much time managing within the available resources and in performing garbage collection.



The memory in which the Java process runs is called the Process Address Space. This memory area is shared by both the Java and native heaps and the operating system. The table displayed on this slide shows the maximum address space available by hardware architecture.

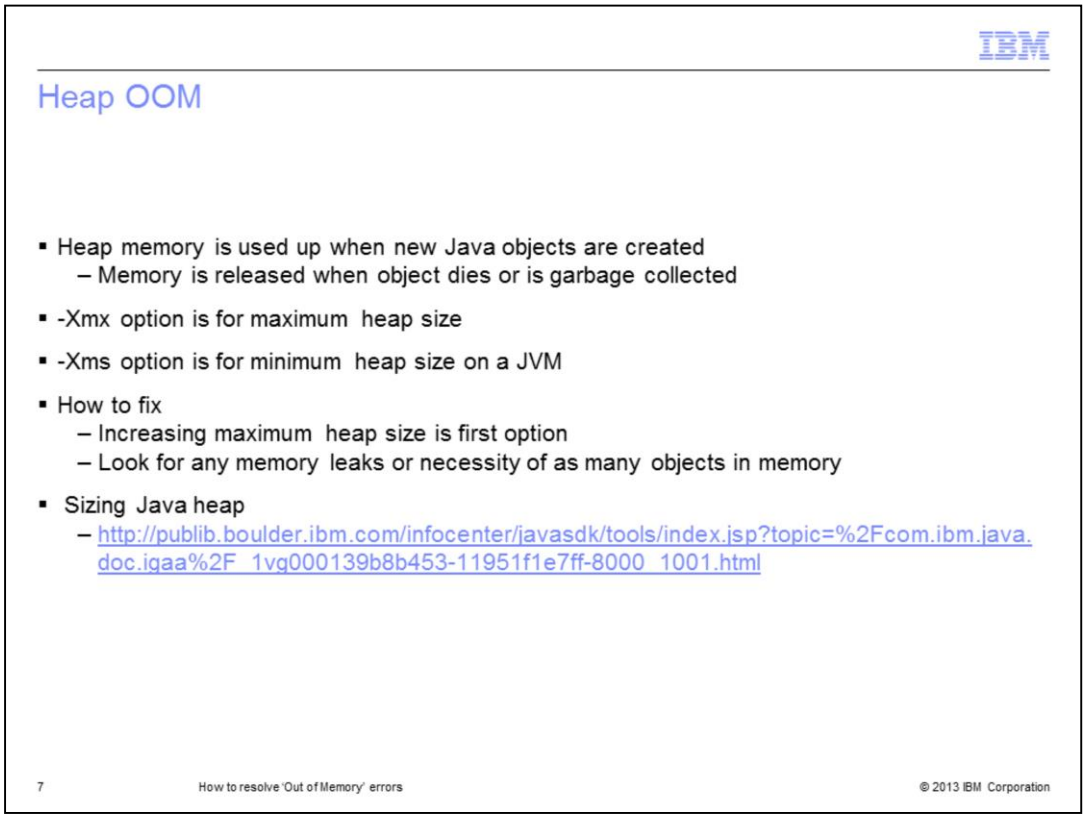
On a 31 or 32 bit processor, to increase the total space available to the native heap, either reduce the size of the Java heap or increase the Process Address Space size. You can increase the Process Address Space only on some operating systems. For example, if the total memory available to Java heap and native heap is approximately two gigabytes, the maximum heap has been set to 1792 megabytes, then 1792 megabytes of address space is reserved and there will only be just over 256 megabytes available for native memory use.

Moving to 64 bit architecture removes the Java heap sizing constraints.



From the previous slide, you should now understand the memory available to a Java process varies by operating system and is used for two separate memory areas, the “native heap” and the “Java heap”. Based on this, OOM errors can be broadly classified into two types, heap OOM and native OOM.

Heap OOM is when Java heap has been exhausted wherein no memory can be allocated to a Java object. Native OOM is when native memory has been exhausted such that memory cannot be allocated to a thread.

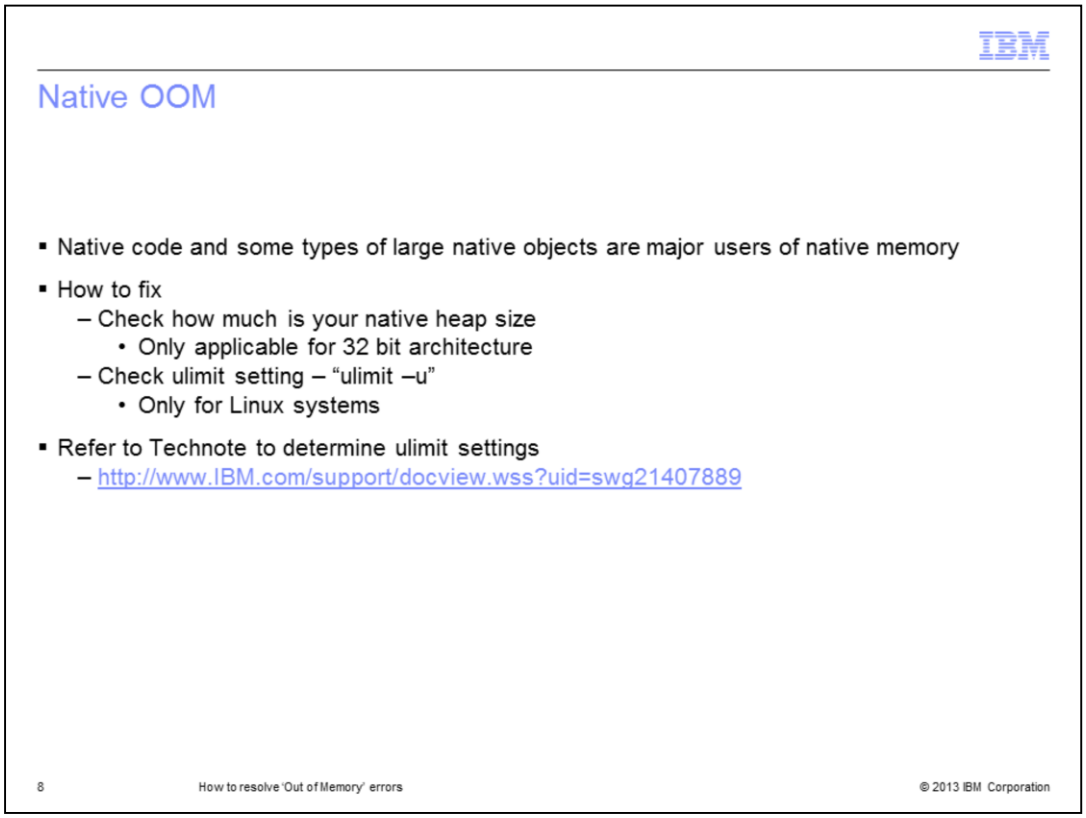


Heap OOM happens when the current heap size settings is insufficient to handle the existing load either because there are many new Java objects being created, or because the current objects are using up more memory, or if there is a memory leak. Memory is freed when objects in memory are no longer referenced by others and can be garbage collected.

The -Xmx and -Xms options on a JVM, define the maximum and minimum Java heap size.

Increasing the maximum heap size for this JVM to permissible limits is the first option to try. Also, look through your code to see if you need as many objects in memory or, if there are some objects that can be released to free up memory. Finally, check for any memory leaks within the application.
As a general guideline on sizing the Java heap, set the maximum Java heap size using the -Xmx command-line option to a value that allows the application to run with 70% occupancy of the Java heap. Set the minimum heap size using the -Xms command-line option so that the Java heap is 40% occupied at its lowest memory usage.

More details on how to set the heap size can be found at the link displayed on this slide.

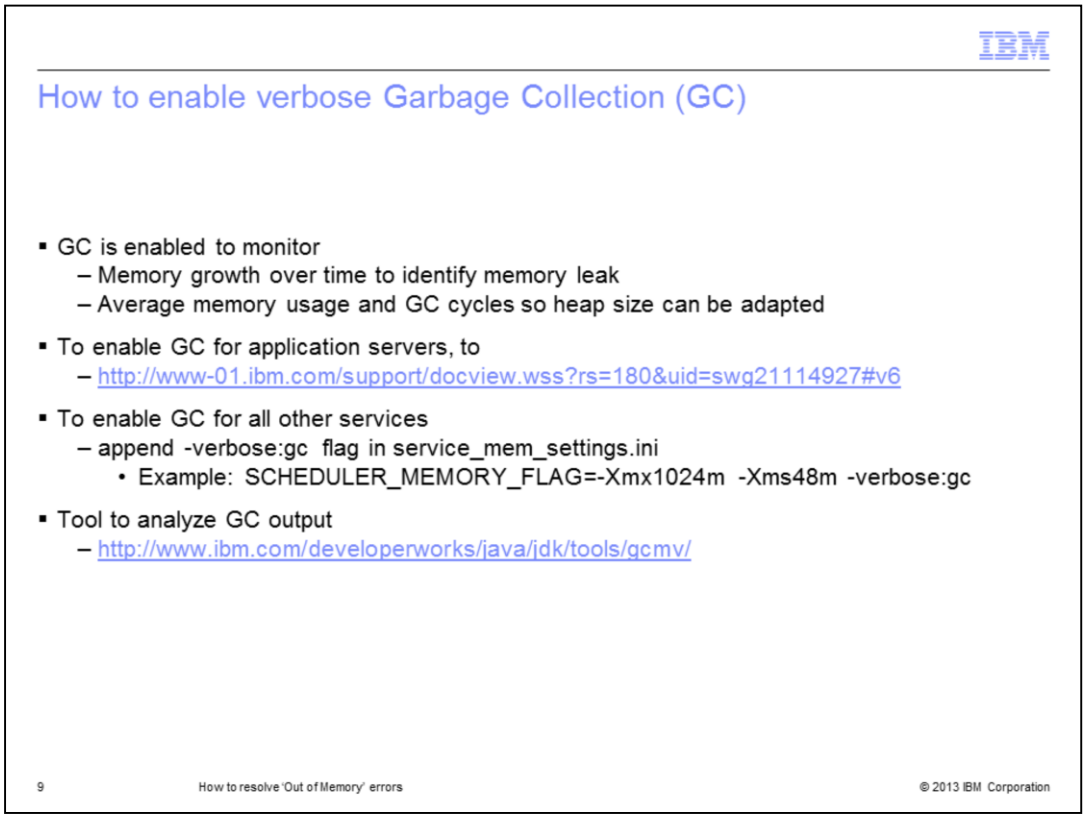


Native heap has objects whose lifetime is the same as the JVM. This memory is not controlled by the garbage collector.

The 'Cannot create thread' error message is often associated with native OOM. If you have a 64 bit operating system, then heap size cannot cause a native OOM situation.

To fix native OOM errors, check how much native memory is left for the JVM. The native memory area is the memory for the process left over from the Java heap. So if your heap size is large, then the native memory will be small causing native OOM errors. Also, if the native OOM occurs on a Linux system related to the ulimit settings, you may also see a "java.net.ConnectException: Connection refused” error. Check the ulimit value for Linux using the "ulimit -a" command. If the value of the “max user processes” parameter is low (the default is 1024), then either make it unlimited or raise it to a high value. Use the command displayed on this slide to set it to unlimited. Refer to the link displayed on this slide to view the Technote "How to determine the ulimit settings of a running WebSphere Application Server process on Linux”.

In summary, if insufficient Process Address Space is available to cover the memory requirements of both the Java heap and the native heap, you must reduce the overall memory usage of the application, typically by scaling the application load over multiple Java processes. For example, load balancing, or by using 64-bit Java.

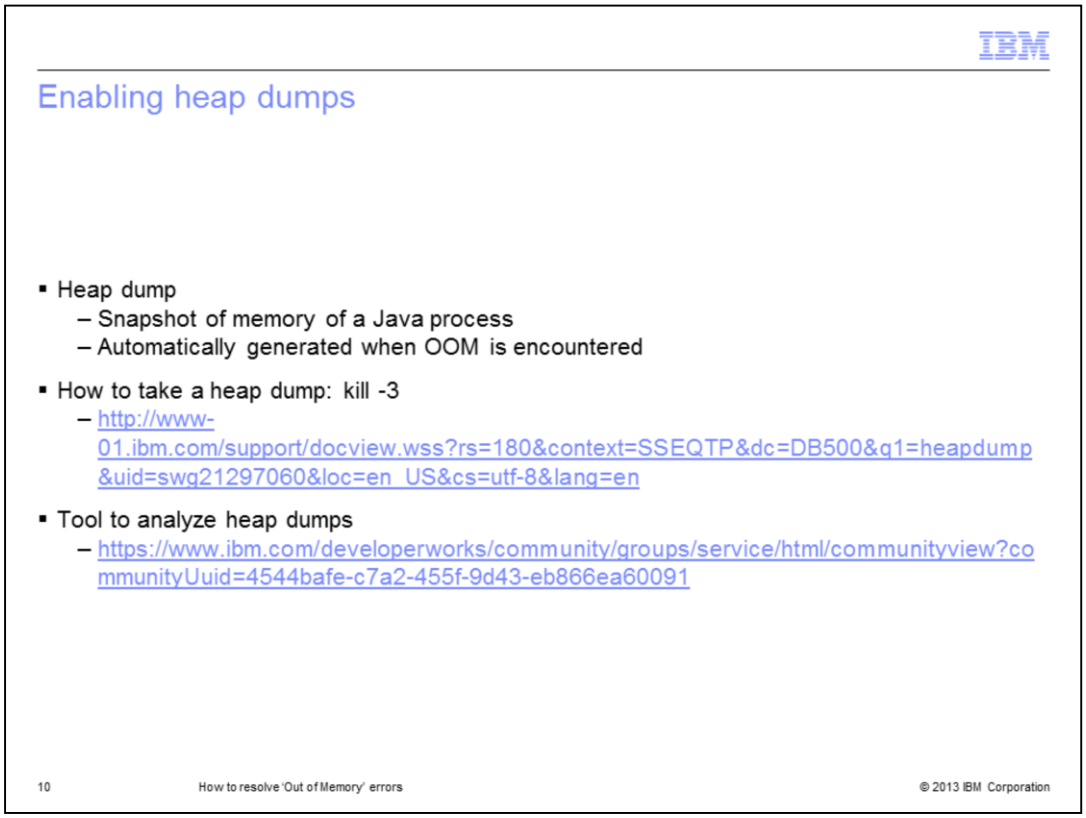


GC refers to Garbage Collection. Sometimes to diagnose issues with how the garbage collector is working, you have to enable verbose GC to know in detail how memory is being used over a period of time. Note that if threads have run out of memory or GC cycles take too long, it will manifest in slower performance of the product. You will see more delayed queries being logged in the corresponding service's db.log files.

To enable GC for application servers service, follow instructions at the link displayed on this slide. Depending on the operating system in use, the verbose GC output is written to a file by default. The file for AIX® and Linux is in native_stderr.log. The file for Solaris is in the native_stdout.log file.

To enable verbose GC in all other product services, edit the service_mem_settings.ini file located in $TOP/bin/conf directory and append -verbose:gc flag. For example, SCHEDULER_MEMORY_FLAG=-Xmx1024m -Xms48m -verbose:gc. You will have to restart the service for the property to take affect.

For enablement of verbose GC in GDS Messaging service, edit the gdsmsg.sh file located in $TOP/bin directory and add verbose:gc to CCD_JMS_JVM_DEBUG_OPTS. For example, CCD_JMS_JVM_DEBUG_OPTS="-Xmx1024m -Xms512m -verbose:gc". Again, a restart of the service is necessary.

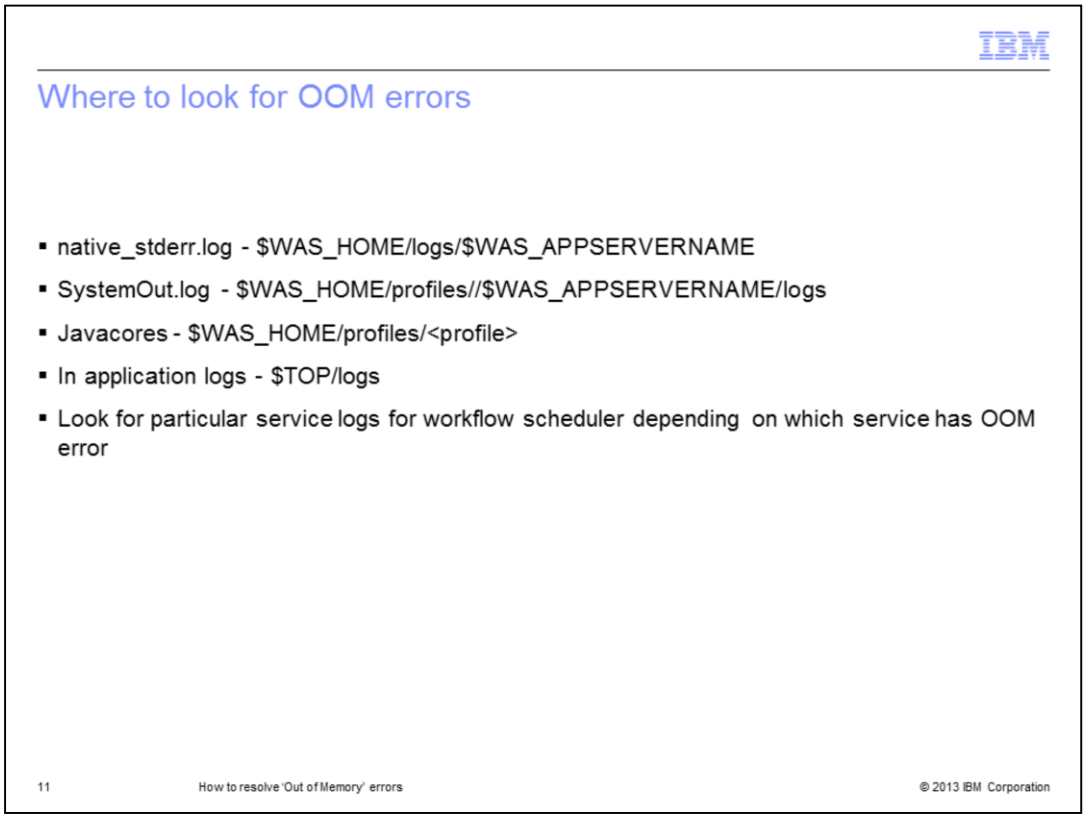


A heap dump is a snapshot of the memory of a Java process. The snapshot contains information about the Java objects and classes in the heap at the moment the snapshot is triggered. Typically, a full GC is triggered before the heap dump is written, so the dump contains information about the remaining objects in the heap. It is typically automatically generated when an OOM condition is reached.

But, in case you have to generate it, you can enable a heap dump by using the Signal SIGQUIT: command. If there is indication of a memory leak which requires heap dumps collected over time, you can enable generation of heap dumps by running kill -3 on the respective service. The best scenario is to generate the heap dump just before the OOM error is seen so that you can analyze which object is taking the maximum memory.

Further instructions on how to take a heap dump using kill -3 are found at the link displayed on this slide.

You can use various tools on the heap dump output to analyze the composition of the objects on the heap and find the objects that are controlling large amounts of memory on the Java heap and the reason why the GC cannot collect them.



Some files which have more diagnostic information printed out related to the ‘Out of Memory’ error are displayed on this slide. There is the native_stderr.log and the SystemOut.log.

The Javacore file, if OOM is from appserver, is found in the first viable location of the directory specified on the IBM_JAVACOREDIR environment variable, the <WAS_install_root>/profiles/<*profile*> directory, the directory specified on the TMPDIR or TEMP environment variable, and the directory where the Java command was executed.

If the Javacore is generated because of OOM from any other product services then look for the file in $TOP.

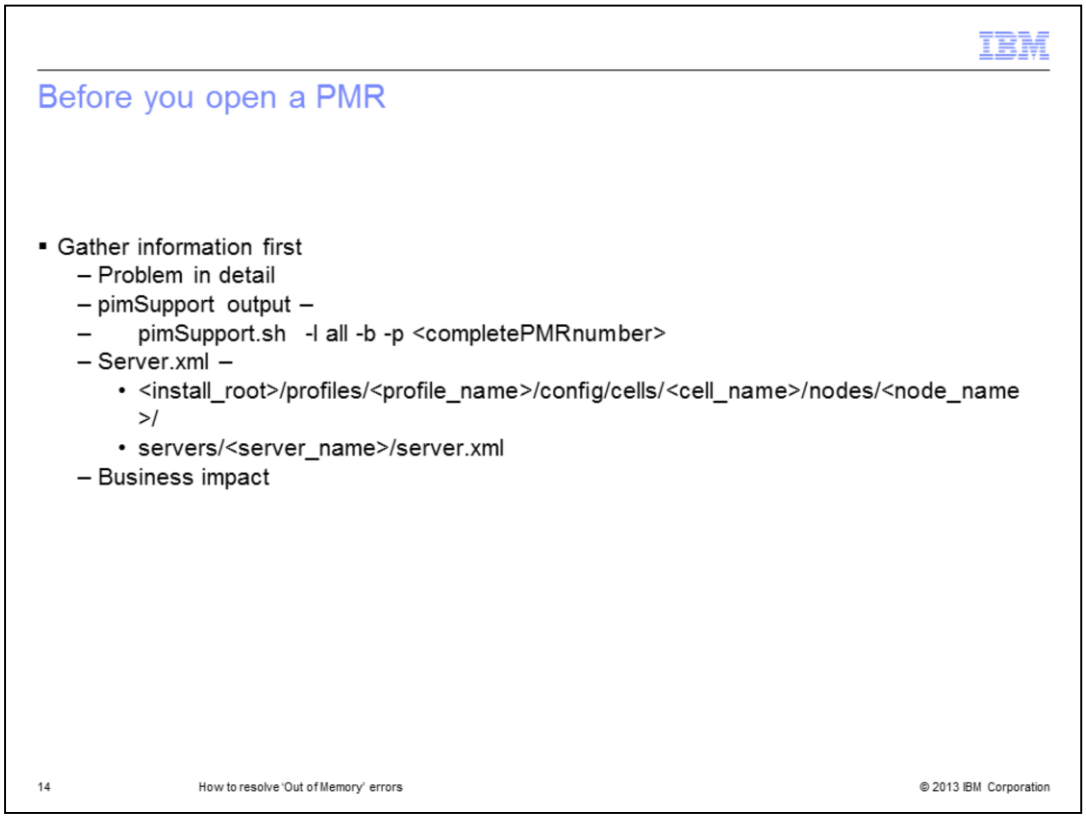
OOM errors are also seen in the product logs and are in the respective directory of the service under $TOP/logs.

This slide displays some sample snippets of WebSphere Application Server logs showing the OOM error.

Sample Product logs

- Scheduler's default.log
- [sch_worker_0] ERROR WPCWPCProfile13245719820840 JOB_ID:840-CWPCM0001E:Generic error / Exception : , Exception:java.lang.OutOfMemoryError
  at java.lang.StringBuffer.ensureCapacityImpl(StringBuffer.java:335)
  at java.lang.StringBuffer.append(StringBuffer.java:201)
  at com.ibm.ccd.common.interpreter.operation.generated.GenDumpSystemLogOperation.execute(Unknown Source)
  at WPCWPCProfile13245719836570.ScriptFunction__splitLogLines
- Error in appsvr's exception.log while running migrateDataToXml.sh
  Caused by: java.sql.SQLException: ORA-27163: out of memory.
  at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTIoer.java:445)
  at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTIoer.java:396)
  at oracle.jdbc.driver.T4C8Oall.processError(T4C8Oall.java:879)
  at oracle.jdbc.driver.T4CTTIfun.receive(T4CTTIfun.java:450)
- Workflow's exception.log
  [WorkflowEventProcessorEventId46193] ERROR com.ibm.ccd.common.error.AustinException - Generic Error, Exception:java.lang.OutOfMemoryError
  at java.io.StringWriter.write(StringWriter.java(Compiled Code))
  at com.ibm.ccd.common.interpreter.engine.ScriptCache.getScript(ScriptCache.java(Compiled Code))

This slide displays some sample snippets of product logs showing the OOM error.



Before opening a PMR and contacting product Support, there is some information you need to have available. You need to have a detailed problem description including how and when you found out that there is a memory issue. Which service did you see having this issue and what part of the product were you using? How often is the OOM situation encountered and if something has changed recently like type of customization or workload.

Also useful for further diagnosis is to upload to the PMR all created javacore files and at least two heap dump files, especially the first and last heap dump files.

Attach to the PMR, the pimSupport output. This will give you the product and the WebSphere Application Server logs in addition to the configuration files. The command to run is displayed on this slide.

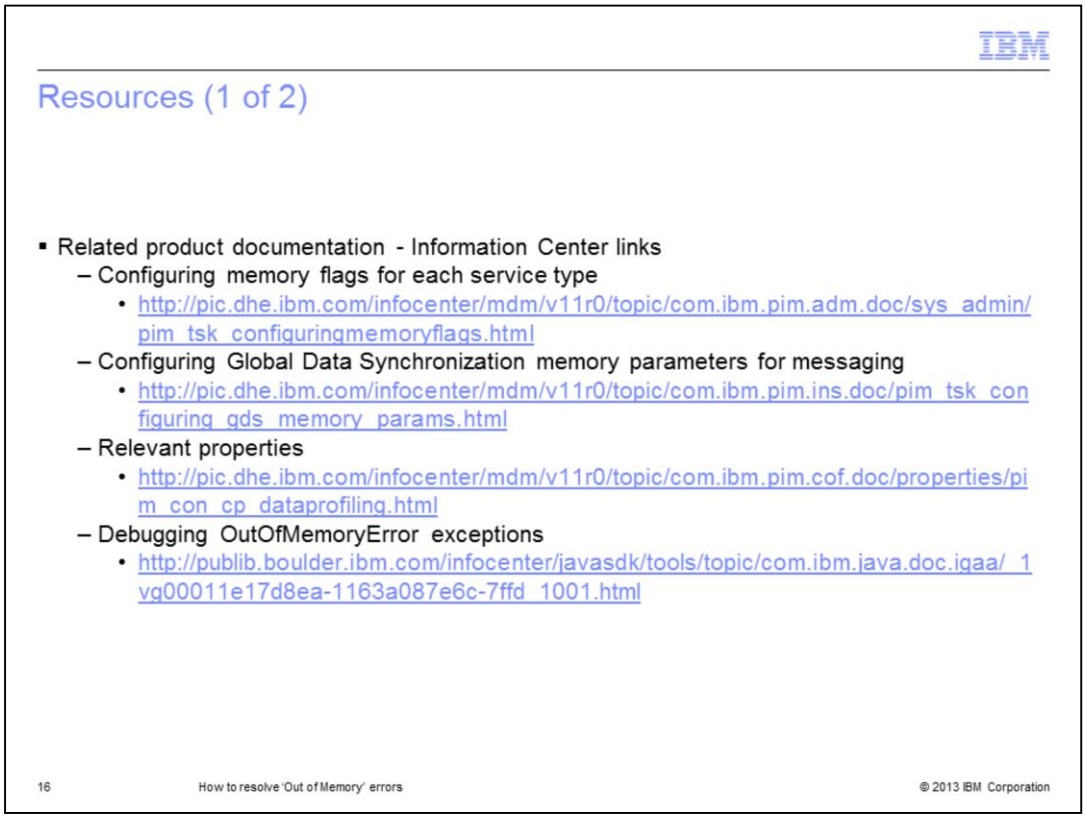
Upload to the PMR the server.xml from the problematic server. The location of this file is as displayed on the slide.

Finally, let Support know what the business impact is of the OOM on your application.

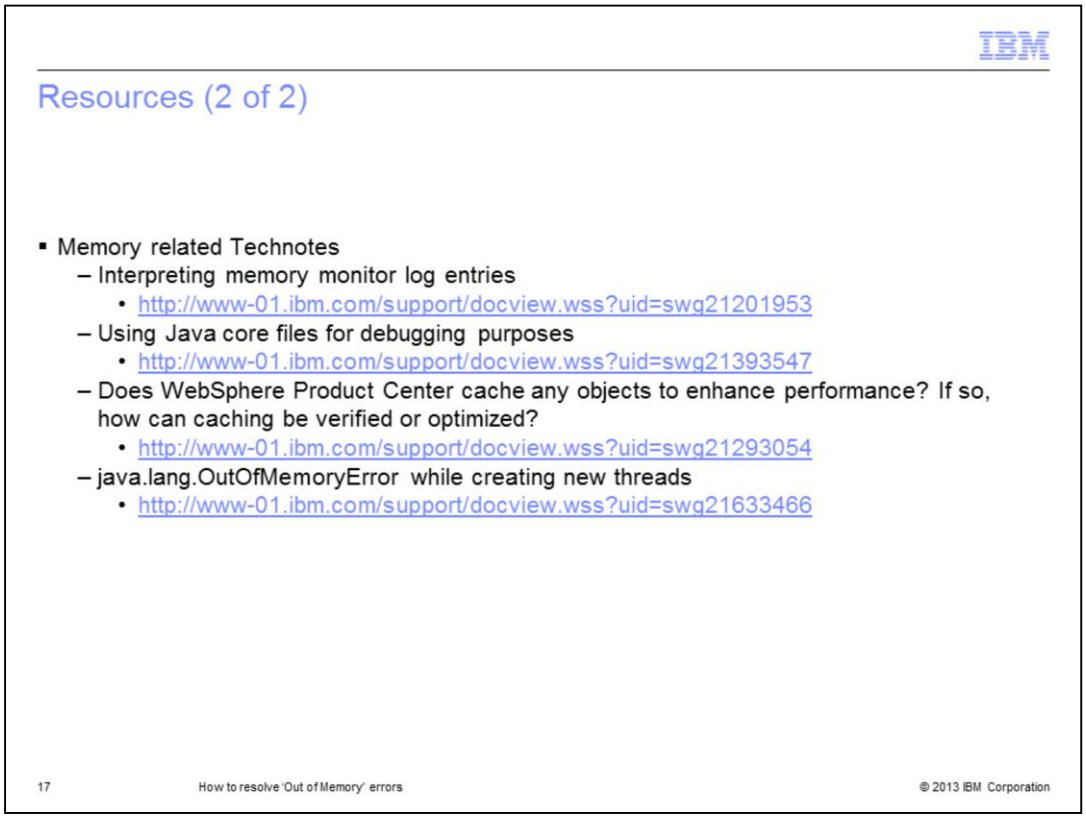
'MustGather' for OOM issues

- What to collect for OOM issues for further diagnosis
  - "MustGather: Out of Memory errors on AIX, Linux, or Windows"
    - http://www.IBM.com/support/docview.wss?uid=swg21138587
  - "MustGather: Native Memory Issues on AIX"
    - http://www.IBM.com/support/docview.wss?uid=swg21405353
  - "MustGather: Native Memory Issues on Linux"
    - http://www.IBM.com/support/docview.wss?uid=swg21138462
  - "MustGather: Out of Memory errors on Solaris - Heap Leak"
    - http://www.IBM.com/support/docview.wss?uid=swg21145349
  - "MustGather: Out of Memory errors on Solaris - Native Leak"
    - http://www.IBM.com/support/docview.wss?uid=swg21104470

This slide displays links for 'MustGather' documents that are required to further diagnose OOM issues.



For reference, this slide displays links that you might find useful.



This slide also displays Technote links that you might find useful.

Trademarks, disclaimer, and copyright information