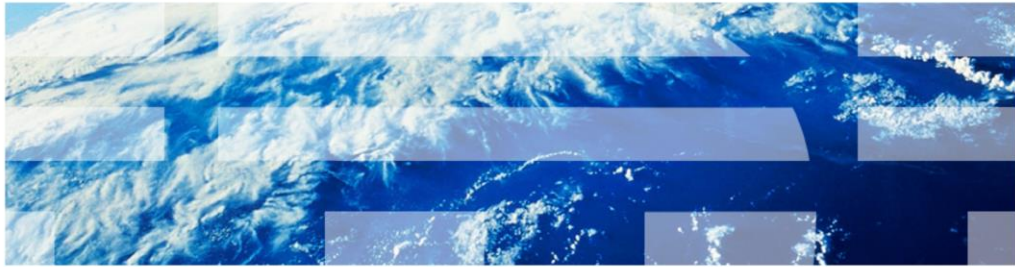


IBM Tivoli Application Dependency Discovery Manager 7.2.1

Learning to read a javacore created by Tivoli Application Dependency Discovery Manager



© 2013 IBM Corporation

In this module, you learn how to read a Tivoli® Application Dependency Discovery Manager javacore file.

Assumptions

- Basic operating system skills
- Basic knowledge of Tivoli Application Dependency Discovery Manager
- Basic understanding of hexadecimal-to-decimal conversion
- Basic text editing skills

Some assumptions before you begin this module are that you have the skills listed here.

Objectives

When you complete this module, you can perform these tasks:

- Determine the service name
Determine what service caused the Out of Memory error
- Determine whether the environment is running in 32-bit or 64-bit mode
- Determine the memory usage at the time of the Out of Memory error
Locate the available memory and the free memory that is available at a specific time
- Read the memory thread dump
Search and determine the thread that caused the Out of Memory error

When you have completed this module, you can perform several tasks:

- Determine the service name and the service that is running at the time of the Out of Memory error.
- Know that the service name provides a starting point for deciding the next actions to take to resolve the issue.
- Determine whether the environment is running in 32-bit or 64-bit mode, which is important because each mode has different memory maximums. You can also find the reason for the Out of Memory error.
- Determine the amount of memory that is available in the environment. This information is important to know. The Out of Memory error might be caused by the system not having enough available memory to perform the request from Tivoli Application Dependency Discovery Manager.

When you read the thread memory dump, you can see the .jar that was running at the time of the Out of Memory error. You can investigate the Tivoli Application Dependency Discovery Manager logs to acquire the root cause of the issue.

Out of physical memory

```

NULL -----
0SECTION  TITLE  subcomponent dump routine
NULL -----
1TISGINFO  Dump Event "systhrow" (00040000) Detail "Java/lang/OutOfMemoryError" received
1TIDATETIME  Date:      2010/10/27 at 01:04:10
1TIFILENAME  Javacore filename: /data/cmdb/dist/external/gigaspace-4.1/bin/javacore.20101027.010353.6970.0009.txt
NULL -----
0SECTION  GPNFO  subcomponent dump routine
NULL -----
2XHOSLEVEL  OS Level   : Linux 2.6.18-128.el5
2XHCPUS     Processors -
3XHCPUARCH  Architecture : amd64
3XHNUMCPUS  How Many  : 4
NULL
1XHERROR2   Register dump section only produced for SIGSEGV, SIGILL or SIGFPE.
NULL

1CICMDLINE  /data/cmdb/dist/external/jdk-1.5.0-Linux-x86_64/jre/bin/java -Dcom.collation.LogFile=/data/cmdb/dist/log/proxy.log -Dcom.collation.servicename=Proxy -
Dcom.collation.home=/data/cmdb/dist -Djava.security.policy=/data/cmdb/dist/etc/policy.all -Dcom.sun.management.jmxremote=true -
Djava.rmi.server.codebase=http://localhost:9430/d/api-dl.jar http://localhost:9430/d/guiserver-dl.jar http://localhost:9430/d/viewmgr-dl.jar http://localhost:9430/lib/sdm-dl.jar -
DCOLLATION -classpath /data/cmdb/dist/lib/admin-msgs.jar:/data/cmdb/dist/lib/anchor-msgs.jar:/data/cmdb/dist/lib/api-server.jar;

1CUAVAHOMEDIR  Java Home Dir: /data/cmdb/dist/external/jdk-1.5.0-Linux-x86_64/jre

2CIUSERARG   -Djava.rmi.server.codebase=http://localhost:9430/d/api-dl.jar http://localhost:9430/d/guiserver-dl.jar http://localhost:9430/d/viewmgr-dl.jar
http://localhost:9430/lib/sdm-dl.jar
2CIUSERARG   -Xms128M
2CIUSERARG   -Xmx2048M

```

This particular javacore is related to the environment running out of physical memory.

- The 1TISGINFO line shows that the problem is an Out of Memory error.
- The 2XHOSLEVEL shows the operating system of the machine; in this case, it is Linux®. You also see the base level and any patches that are applied.
- The 3XHCPUARCH shows the architecture of the machine. In this case, it is amd64 bit CPU.
- The 3XHNUMCPUS shows the number of CPUs on the machine. In this case, it is four CPUs.
- The 1CICMDLINE shows the servicename that caused the Out of Memory error. In this case, it is Proxy service.
- The 2CIUSERARG shows the jvmargs. Be sure to note the `-Xms` and `-Xmx` settings.



Check memory consumption

- The 0SECTION shows the memory that is allocated for the service

```
0SECTION  MEMINFO subcomponent dump routine
NULL      -----
```

```
1STHEAPFREE  Bytes of Heap Space Free: 2b9f4108
```

```
1STHEAPALLOC Bytes of Heap Space Allocated: 80000000
```

- These are the hexadecimal values of the free and allocated memory. Use the calculator in programmer mode to convert. Subtract to determine how much is in use

```
2b9f4108 = 731857160 (700 MB free)
```

```
80000000 = 2147483648 (2 gigabit allocated for the JVM)
```

This check shows only 700 MB of free allocated memory

Note: Max memory is never returned when allocated

The 0SECTION shows the memory information, in hexadecimal. In this case, there is not enough allocated memory. There is only 700 MB of memory free, and that is not enough to complete the action that is requested by the proxy. Confirm that additional memory is available on the server, and increase the max (Xmx) memory that is allocated for the proxy service. Note that max memory is never given back after it is allocated. Keep that in mind when you allocate memory.

Logical connections (1 of 3)

```

NULL
-----
0SECTION  TITLE  subcomponent dump routine
NULL
-----
1TISGINFO  Dump Event "systhrow" (00040000) Detail "Java/lang/OutOfMemoryError" received
1TDATETIME Date:      2010/06/08 at 20:21:32
1TFILENAME Javacore filename: /tivolitaddm/dist/external/gigaspace-4.1/bin/javacore.20100608.201936.827646.bt
NULL

0SECTION  GPNINFO subcomponent dump routine
NULL
-----
2XHOSLEVEL OS Level   : AIX 5.3
2XHCPUS    Processors -
3XHCPUARCH Architecture : ppc
3XHNUMCPUS How Many  : 8
NULL
1XHERROR2  Register dump section only produced for SIGSEGV, SIGILL or SIGFPE.
NULL

1CICMDLINE /tivolitaddm/dist/external/jdk-1.5.0-AIX-powerpc/jre/bin/java -Dcom.collation.LogFile=/tivolitaddm/dist/log/topology.log -Dcom.collation.servicename=Topology -
Dcom.collation.home=/tivolitaddm/dist -Djava.security.policy=/tivolitaddm/dist/etc/policy.all -Dcom.sun.management.jmxremote=true -
Djava.rmi.server.codebase=http://localhost:9430/dl/topomgr-dl.jar

1CUAVAHOMEDIR Java Home Dir: /tivolitaddm/dist/external/jdk-1.5.0-AIX-powerpc/jre

2CIUSERARG  -Djava.rmi.server.codebase=http://localhost:9430/dl/topomgr-dl.jar http://localhost:9430/dl/topomgr-events-dl.jar
http://localhost:9430/dl/topobuilder-dl.jar http://localhost:9430/dl/topobuilder-events-dl.jar http://localhost:9430/dl/reports-dl.jar http://localhost:9430/dl/monitor-
statemanager-dl.jar http://localhost:9430/dl/monitor-engine-dl.jar http://localhost:9430/dl/changemgr-dl.jar http://localhost:9430/lib/sdm-dl.jar
2CIUSERARG  -Xmx1024M

You see the service name, the architecture of the machine, the Java that is used, the last .jar file that is used when the Out of Memory error
occurs, and the jvmarg setting for it when it was out of memory

```

6

Learning to read a javacore created by Tivoli Application Dependency Discovery Manager

© 2013 IBM Corporation

This particular javacore is related to logical connections.

- The 1TISGINFO shows the Out of Memory error.
- The 2XHOSLEVEL shows the operating system of the machine. In this case, it is AIX® 5.3.
- The 3XHCPUARCH shows the architecture of the machine. In this case, it is a Power PC architecture.
- The 3XHNUMCPUS shows the number of CPUs of the machine. In this case, it is eight CPUs.
- The 1CICMDLINE shows the service name that is causing the Out of Memory error. In this case, it is the Topology Service.
- The 2CIUSERARG shows the Java arguments that are used.

Logical connections (2 of 3)

```

3XMTHEADINFO      "TopologyBuilderEngineThread" (TID:0x33F8AC00, sys_thread_t:0x33B1D560, state:CW, native ID:0x0065800B) prio=5
4XESTACKTRACE     at javax/jdo/spi/JDOImplHelper.newObjectInstance(Bytecode PC:21(Compiled Code))
4XESTACKTRACE     at kodo/util/ApplicationIds.fromPKValues(ApplicationIds.java:126(Compiled Code))
4XESTACKTRACE     at kodo/jdbc/meta/AbstractClassMapping.getObjectId(AbstractClassMapping.java:243(Compiled Code))
4XESTACKTRACE     at kodo/jdbc/runtime/JDBCStoreManager.loadMappings(JDBCStoreManager.java:931(Compiled Code))
4XESTACKTRACE     at kodo/jdbc/sql/AbstractResult.load(AbstractResult.java:184(Compiled Code))
4XESTACKTRACE     at kodo/jdbc/sql/Select$SelectResult.load(Select.java:2181(Compiled Code))
4XESTACKTRACE     at kodo/jdbc/meta/ManyToManyFieldMapping.loadProjection(ManyToManyFieldMapping.java:333(Compiled Code))
4XESTACKTRACE     at kodo/jdbc/meta/AbstractCollectionFieldMapping.load(AbstractCollectionFieldMapping.java:596(Compiled Code))
4XESTACKTRACE     at kodo/jdbc/runtime/JDBCStoreManager.load(JDBCStoreManager.java:526(Compiled Code))
4XESTACKTRACE     at kodo/runtime/DelegatingStoreManager.load(DelegatingStoreManager.java:133(Compiled Code))
4XESTACKTRACE     at kodo/runtime/ROPStoreManager.load(ROPStoreManager.java:79(Compiled Code))
4XESTACKTRACE     at kodo/runtime/StateManagerImpl.loadFields(StateManagerImpl.java:3131(Compiled Code))
4XESTACKTRACE     at kodo/runtime/StateManagerImpl.loadField(StateManagerImpl.java:3230(Compiled Code))
4XESTACKTRACE     at kodo/runtime/StateManagerImpl.isLoaded(StateManagerImpl.java:1368(Compiled Code))
4XESTACKTRACE     at com/collation/topomgr/jdo/topology/sys/RuntimeProcessJdo.jdoGetports_(RuntimeProcessJdo.java(Compiled Code))
4XESTACKTRACE     at com/collation/topomgr/jdo/topology/sys/RuntimeProcessJdo.hasPorts(RuntimeProcessJdo.java(Compiled Code))
4XESTACKTRACE     at
com/libm/cdb/topomgr/topobuilder/agents/ConnectionDependencyAgent.populateCacheMaps(ConnectionDependencyAgent.java:498(Compiled Code))
4XESTACKTRACE     at com/libm/cdb/topomgr/topobuilder/agents/ConnectionDependencyAgent.runInternal(ConnectionDependencyAgent.java:106(Compiled
Code))
* 4XESTACKTRACE   at com/libm/cdb/topomgr/topobuilder/agents/ConnectionDependencyAgent.run(ConnectionDependencyAgent.java:84)
4XESTACKTRACE     at com/libm/cdb/topomgr/topobuilder/engine/TopologyBuilderEngine.buildTopology(TopologyBuilderEngine.java:208(Compiled Code))
4XESTACKTRACE     at com/libm/cdb/topomgr/topobuilder/engine/TopologyBuilderEngine.access$100(TopologyBuilderEngine.java:59)
4XESTACKTRACE     at com/libm/cdb/topomgr/topobuilder/engine/TopologyBuilderEngine$1.run(TopologyBuilderEngine.java:170)
4XESTACKTRACE     at java/lang/Thread.run(Thread.java:810(Compiled Code))

```

- The TopologyBuilderAgent is single-threaded
- Current threads are not typically what cause the issue; it is typically in the All Threads Details
- Anything ending in jdo is automatically generated dynamically or listed as (Compiled Code)

7

Learning to read a javacore created by Tivoli Application Dependency Discovery Manager

© 2013 IBM Corporation

The 3XMTHEADINFO shows that TopologyBuilderEngineThread is the point of failure.

The 4XESTACKTRACE shows the code that failed. In this case, it is highlighted in red and an asterisk has been added to the trace to help you locate the line.

Note that the TopologyBuilderAgent is single-threaded.

Current threads are not typically what cause the issue. It is typically in the All Threads Details.

Anything ending in jdo is automatically generated dynamically or listed as (Compiled Code). This type of code requires a code change because it is compiled.

Logical connections (3 of 3)

The conclusion is as follows:

- The logical connections are legitimate
- You must increase the topology Java virtual machine (JVM) to 2 gigabit or move to a 64-bit operating system to increase beyond 3 gigabit

The reason for the logical connection Out of Memory error is that the logical connections were legitimate, but the allocated memory did not allow for the number of logical connections. Increase the Topology JVM Xmx setting. You must have a 64-bit architecture if you have to increase memory beyond 3 gigabit.

Discovery javacore

- Using javacore for a discovery issue is much more difficult
- You look for the servicename as you do in any other javacore
 - If there are multiple servicenames, the last one is the one that is running
- The CURRENT THREADS section provides a clue for where to look in the logs
- If there is a DiscoverWorker, you must look in the logs
- ALL THREADS details
 - When you see DONE_DISCOVER_SENSOR_CLEANUP, it timed out, but it is still running
- dist/log/services/DiscoverManager.log
 - After waiting for threads to finish and before the JVM restarts, if you do not see JVM restart, look in this log for the issue
- dist/log/sensors
 - Look in the sensors directory for logs that did not complete

Discovery out-of-memory is more difficult to diagnose. You can find the service name. However, if you see multiple service names, the last one is the one that is running at the time of the Out of Memory error.

The current thread at the bottom of the javacore provides a clue to where to look in the dist/log log files. Search on 4XE and look for a heading of Current threads.

If there is a DiscoverWorker, then look in the dist/log log files.

In the All Threads Details, look for DONE_DISCOVER_SENSOR_CLEANUP. This is where it timed out, but is still running.

DiscoverManager.log shows everything after waiting for the threads to finish the message and before the JVM restarts. If JVM does not restart, then look for the issue in the dist/log/sensors directory.

In the dist/log/sensors directory, look for sensors that did not complete.

Summary

Now that you have completed this module, you can perform these tasks:

- Determine the servicename
 - Determine what service caused the Out of Memory error
- Determine whether the environment is running in 32-bit or 64-bit mode
- Determine the memory usage at the time of the Out of Memory error
 - Locate the available memory and the free memory that is available at a specific time
- Read the memory thread dump
 - Search and determine the thread that caused the Out of Memory error

Now that you completed this module, you can determine the servicename and the service that is running at the time the Out of Memory error occurred.

You know whether the environment is running in 32-bit or 64-bit mode, which is important because each mode has different memory maximums. You know the amount of memory allocated to the service.

Reading the thread memory dump provides the .jar file that is running at the time of the Out of Memory error. You can investigate the Tivoli Application Dependency Discovery Manager logs when needed to acquire the root cause of the issue.

Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, AIX, and Tivoli are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at <http://www.ibm.com/legal/copytrade.shtml>

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.

© Copyright International Business Machines Corporation 2013. All rights reserved.