Getting started with analysis of GC, Heapdumps and Javacores
For WebSphere on z/OS

Techdoc WP101612

This document can be found on the web at:
www.ibm.com/support/techdocs
Search for document number WP101612 under the category of "TechDocs"

Version 1.1

Author: Edward McCarthy
TechWorks
IBM AP Software Group
IBM
edwardmc@au1.ibm.com

Last Updated: 14 December 2009, 12:07
Table of Contents

1 PRACTICAL PROBLEM DETERMINATION FOR WEBSPHERE ON Z/OS .3

1.1 Introduction ...................................................................................................................................... 3
   1.1.1 What is supplied with this techdoc? ........................................................................................... 3

1.2 WebSphere and three main areas of problem investigation ......................................................... 4
   1.2.1 Garbage Collection analysis ....................................................................................................... 4
   1.2.2 Heapdumps ................................................................................................................................. 4
   1.2.3 Javacores...................................................................................................................... ............... 5
   1.2.4 Why more then one tool?........................................................................................................ .... 5

1.3 PD application toolkit......................................................................................................... .............. 6
   1.3.1 Install the ear file ........................................................................................................................ 6

1.4 JMeter.................................................................................................................................................. 6
   1.4.1 How to run JMeter .................................................................................................................... 6
   1.4.2 Supplied jmx script files ............................................................................................................. 7

1.5 Comment re ISA and the tools ........................................................................................................ 7

1.6 Installing ISA and the tools....................................................................................................... ........... 7
   1.6.1 Installing ISA ........................................................................................................................... 7
   1.6.2 Adding the tools described in this Techdoc ................................................................................ 8

1.7 How to use the tools ......................................................................................................................... 11

2 GARBAGE COLLECTION ANALYSIS ......................................................................................... 12

2.1 Enable verbose Garbage Collection .............................................................................................. 12

2.2 Generate and collect verbose GC output...................................................................................... 13
   2.2.1 Run JMeter ................................................................................................................................. 14
   2.2.2 Server Heap size considerations ............................................................................................... 15
   2.2.3 Monitor verbose GC ................................................................................................................. 16
   2.2.4 Understanding the GC output ................................................................................................... 17
   2.2.5 Collect GC output...................................................................................................................... 18

2.3 Analyse GC using Garbage Collection and Memory Visualizer (GCMV) ................................ 19

2.4 Analyse GC using PMAT .......................................................................................................... ..... 21

2.5 Summary ......................................................................................................................................... 25

3 HEAPDUMP ANALYSIS ................................................................................................................... 26

3.1 Run JMeter ......................................................................................................................................... 26

3.2 Collect heapdump........................................................................................................................... 27

3.3 Heapdump analysis using IBM Memory Dump Diagnostic for Java ( MDD4J ) ................. 28
3.4 HeapDump analysis using IBM Monitoring and Diagnostic Tools for Java™ - Memory Analyzer

3.5 Heapdump analysis using AlphaWorks HeapAnalyzer

3.6 Summary

4 JAVACORE ANALYSIS

4.1 Run JMeter

4.2 Collect javacores

4.3 javacore’s analysis using IBM Thread and Monitor Dump Analyzer

4.4 Summary
1 Practical problem determination for WebSphere on z/OS

1.1 Introduction

IBM has previously published an IBM whitepaper titled: “Introducing the IBM Support Assistant for WebSphere on z/OS”. That white paper described the IBM Support Assistant (ISA) product and the various problem diagnosis tools incorporated into it. The whitepaper can be downloaded from here:

http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101575

The aim of this techdoc is to demonstrate the process involved with using ISA to investigate three main types of problems associated with WebSphere Application Server on z/OS, namely:

1. Garbage collection analysis
2. Heap usage - Heapdumps
3. Poor response - Javacores

The above three types of problems that can occur with WebSphere on z/OS are the same as what can occur with WebSphere on any distributed platform.

If you are already experienced with using the ISA tools described in this document from working with WebSphere on distributed platforms, then this document will show how only a relatively small amount of additional knowledge is required to use the same tools for analysis tasks for WebSphere on z/OS.

If you are new to supporting WebSphere in general, then this document will also be of use in showing how to go about using these tools.

The application examples used to demonstrate how to use ISA, are freely available for you to download and try out on your own system. This allows you to try out these commonly used ISA tools on your own system, so that you can gain some familiarity with them, rather than trying to work out how to use them for the first time when there is some crisis in your production environment.

1.1.1 What is supplied with this techdoc?

The various ear files etc that are used in the following chapters to create the various problem scenarios are available for download from:

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101612

The zip file contains the following:

<table>
<thead>
<tr>
<th>Sub-directory</th>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>wasPdTools.zip</td>
<td>RAD Project Interchange file, contains the java source code used in the sample applications</td>
</tr>
<tr>
<td></td>
<td>wasPdTools.ear</td>
<td>Ear file to deploy into WebSphere</td>
</tr>
<tr>
<td>jmeter</td>
<td>wasPdTools-verboseGC.jmx</td>
<td>Script to generate verbose GC output</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>wasPdTools-causeOutOfMemory.jmx</td>
<td>Script to cause out of memory condition in WAS server</td>
<td></td>
</tr>
<tr>
<td>wasPdTools-Sleeper.jmx</td>
<td>Script to run long running requests in WAS server</td>
<td></td>
</tr>
</tbody>
</table>

### 1.2 WebSphere and three main areas of problem investigation

The purpose of WebSphere is to run customer application programs. During the running of these programs, it is possible that issues can arise that can cause the performance of the application to be less than hoped for. When this occurs, it is the responsibility of the WebSphere administrator to investigate what is the cause of the performance impact.

In this techdoc, the focus is on using ISA and the common tools it provides to help investigate three common areas of WebSphere performance.

Note the aim of this techdoc is to show you how to use the tools. Mastery of the tools and the information they can help provide is a skill, which can really only be obtained to a large extent via the school of actual experience.

#### 1.2.1 Garbage Collection analysis

The most important component of WebSphere is the Java Virtual Machine (JVM) that the application programs run in. As applications run in the JVM, they create objects of various size and periodically the JVM performs a garbage collection (GC) operation to remove objects that are no longer required so that applications can create new objects.

Analysis of how garbage collection is performing in the JVM is critical to determining what tuning adjustments, if any, need to be made to the JVM heap settings.

This techdoc will:

1. show how to enable verbose GC
2. show how to use the IBM Monitoring and Diagnostic Tools for Java™ - Garbage Collection and Memory Visualizer (GCMV) to analyse the GC output
3. show how to use the IBM Pattern Modeling and Analysis Tool for Java Garbage Collector (PMAT) to analyse the GC output

#### 1.2.2 Heapdumps

A common problem that occurs with WebSphere is the JVM heap getting an out of memory exception. This typically occurs when

1. applications create either too many objects and GC processing is unable to remove enough old objects
2. applications create very large objects that fill the JVM
3. both of the above
Alternatively the WebSphere server may not get an out of memory condition, but you may suspect that large amounts of memory in the JVM heap are being used up by objects that are not being freed. In such situations you could take a heapdump of the running WebSphere server and then analyse it to determine if your suspicions are correct.

This techdoc will:

1. show how to collect the heapdump in WebSphere on z/OS generated when an out of memory condition occurs
2. how to use the IBM Memory Dump Diagnostic for Java (MDD4J) tool to analyse the heapdump
3. how to use the IBM Monitoring and Diagnostic Tools for Java™ - Memory Analyzer tool to analyse the heapdump
4. how to use the IBM Heap Analyzer tool

1.2.3 Javacores

Finally another common problem that can occur when running applications in WebSphere, is that response time for the application is poor, for example response time under load may be in minutes rather than the desired sub-second response.

Typically this can be caused by long running requests. A common way to identify these long running requests is to take a series of javacores and then analyse them.

This techdoc will:

1. show how to generate a javacore in WebSphere on z/OS
2. how to use the IBM Thread and Monitor Dump Analyzer for Java tool to analyse the java cores

1.2.4 Why more than one tool?

In each of the three problem areas above, more than one tool is listed. You might be wondering why this is, why there is not just one tool for each type of problem. While each tool for the various problems has the same objective, they just provide different ways of analysing the problems. Some are more user friendly but perhaps do not provide the same capability for deep analysis that another tool may provide.

Which tool you use may well depend on your technical capability. There is no right or wrong tool, try them all and use the one you feel most comfortable with.
1.3 **PD application toolkit**

An ear file called wasPdTools.ear is supplied with this techdoc. This ear file contains two servlets, one is used to help demonstrate analysis of verbose GC and heapdumps, and the other used to help demonstrate javacore analysis.

1.3.1 **Install the ear file**

Use the WebSphere Admin GUI to install the wasPdTools.ear into a WAS server.

1.4 **JMeter**

JMeter is an open source tool that can be used to generate load onto any target server. Load generation scripts can be graphically built in JMeter and then saved to a file, typically with a jmx suffix.

It is used in each of the following chapters to create load on WebSphere server to help cause the problem the associated tool helps to investigate.

Download JMeter from the Apache web site for JMeter. The version used in association with this tech doc was downloaded from:

[http://apache.wildit.net.au/jakarta/jmeter/binaries/jakarta-jmeter-2.3.4.zip](http://apache.wildit.net.au/jakarta/jmeter/binaries/jakarta-jmeter-2.3.4.zip)

Unzip the zip file to a directory, there is no product installer to run.

1.4.1 **How to run JMeter**

From a DOS prompt, change to the directory where JMeter is located and then to the bin sub-directory. Type in jmeter and press Enter and the initial JMeter window similar to that shown below will display:

![JMeter Window](image)
1.4.2 Supplied jmx script files

The following three JMX files are supplied with this techdoc for use in the following chapters:

wasPdTools-verboseGC.jmx
wasPdTools-causeOutOfMemory.jmx
wasPdTools-Sleeper.jmx

1.5 Comment re ISA and the tools

In this document we describe how to use ISA to launch the various problem analysis tools. Some of these tools can be downloaded from IBM sites like Alphaworks, installed on your PC and launched from a DOS prompt, without needing to use ISA at all.

This approach is not described in this document.

ISA is the IBM strategic product for providing a single point to integrate the various tools used to assist with analysis of WebSphere problems.

ISA provides a simple approach for obtaining the tools you want to use, getting updates for the installed tools and launching the tools.

1.6 Installing ISA and the tools

A number of tools are described in this techdoc. To use them on your system you will need to install ISA on your PC.

1.6.1 Installing ISA

Techdoc WP100575 provides references to information about how to obtain and install ISA. In case you do not have that document handy, some of that detail from that document is repeated here:

The installation of the ISA and its analysis tools is well documented in several websites shown below, so we will not repeat the instructions here.

Download from http://www.ibm.com/software/support/isa/

The IBM Education Assistant (IEA) has several tutorials on installing and using the ISA at:

Once you have the ISA installed on your workstation, it is quite easy to download specific diagnostic tools as “add-ons”. There are over 100 Product Add-ons and a growing list of tool add-ons.

### 1.6.2 Adding the tools described in this Techdoc

To add the tools to ISA, select Update -> Find New -> Tools Add-ons as shown below:

![Image of the Support Assistant Workbench interface](image)

On the next screen displayed, expand the JVM-based Tools tree and select the products to be installed as shown below:

![Image of the JVM-based Tools tree](image)
Note HeapAnalyzer is not shown in the above display but should also be selected.

When you click Next, you will get a display of the product license conditions. Read these and then click Accept to accept the license conditions and then click Next. A screen is then displayed showing a list of the products and their versions you selected to add to ISA. Click Finish and a window will be displayed which shows that installation of the tools is underway. An internet connection is required, as the software is downloaded from an IBM site.

This process will take some time depending on your line speed, at the end of which a screen similar to the following appear:
Click Finish and then ISA will prompt you to do a restart. The tools will now be available for use.

To use these tools, click on ‘Analyze Problem’ on the ISA home page as shown below, which will bring up a list of the various tools that can be used.
1.7 How to use the tools

This document does not provide in-depth detail of how to use each tool described, the focus is on how to use each tool in association with WebSphere on z/OS. A basic overview is provided for each tool. Full documentation about each tools capability is available online.

What is supplied with this Techdoc

The zip file that you can download from
2 Garbage Collection analysis

This chapter will demonstrate how analysis of Garbage Collection output can be done using the following tools:

1. IBM Monitoring and Diagnostic Tools for Java™ - Garbage Collection and Memory Visualizer (GCMV)
2. IBM alphaWorks Modeling and Analysis Tool for Java Garbage Collector

2.1 Enable verbose Garbage Collection

To be able to analyse how GC is performing in the JVM, the WebSphere server must be configured to produce verbose GC output. There is very little overhead in having a WebSphere server produce verbose GC.

Enabling verbose GC output in a WebSphere server is the same process for WebSphere on all platforms. The only difference for WebSphere on z/OS is that a server consists of a control region, servant region and optionally an adjunct region if the server is running a messaging engine. This means that you need to specifically select the servant region as the one you want to enable verbose GC output on. You can also enable verbose GC output for the control and adjunct region as well if desired, but most tuning of the JVM heap occurs in the servant region where the applications run.

To configure a server to produce verbose GC output, in the WebSphere Admin GUI select Application servers -> <serverName> -> Process definitions -> Servant -> Java Virtual Machine as shown below, and then click on the box to the left of the string "Verbose garbage collection".
Click OK and then save the change. The server must be restarted to pick up this change.

In a WebSphere server on z/OS, set up in the default way, the verbose GC output is written to the SYSOUT part of the STC output. To verify that, you would enter a question mark beside the server in the SDSF DA display to get a display similar to that shown below and then select the SYSOUT display.

```
SDSF JOB DATA SET DISPLAY - JOB WYSR01AS (STC21345)
COMMAND INPUT ===>  
NP   DDNAME   StepName ProcStep DSID Owner    C Dest  
JESMSGLG JES2                 2 WYASRU   S  
JESJCL    JES2                 3 WYASRU   S  
JESYSMSG  JES2                 4 WYASRU   S  
SYSOUT    WYSR01AS           103 WYASRU   S  
SYSPRINT  WYSR01AS           105 WYASRU   S  
```

### 2.2 Generate and collect verbose GC output

Having all the required software in place, the next step is to use JMeter to generate some load on the server, by running the heapEater application, which will result in verbose GC output being generated.
2.2.1 Run JMeter

In the initial JMeter window select File -> Open to bring up a display that you can use to locate the file called wasPdTools-verboseGC.jmx and select it. The display will then look similar to this:

![JMeter Display](image.png)

This jmx file is a pre-configured load script to run the wasPdTools application. You need to modify the target TCPIP address and TCPIP port set in this script, do this by clicking on ‘wasV7ZosServer’, and change the values in the ‘Server Name or IP’ and ‘Port Number’ fields as shown below:

![Modified JMeter Display](image2.png)

The ‘Port Number’ will be the TCPIP port assigned to the ‘WC Default Host’ in the WebSphere server.
There is no save button to save your changes, the values you type in take effect straightaway.

Once you have updated these fields, click on Run -> Start to have JMeter start executing the load script.

### 2.2.2 Server Heap size considerations

The aim of this exercise is to generate verbose GC output and then analyse it. How frequent verbose GC messages are generated will depend on the heap size settings of the JVM in the server. When we tested this process, the heap size for the server was set to a minimum size of 256M and maximum of 512M. There are five types of requests, allocating objects of approximate sizes of 10K, 20K, 100K, 250K and 500K.

The requests for objects of size 10K and 20K will allocate objects in the Small Object Area (SOA) of the heap, while the 100K, 250K and 500K will be allocated out of the Large Object Area (LOA) part of the heap.

If your server has a much larger heap then you would either need to reduce it, or modify the JMeter load script.

You can change the size of the objects created for each request issued, by clicking on ‘HTTP Request’ type entries, double click in the field under the Value heading, for the Parameter called objectSize and then type in a new value as shown below:
Alternatively you can change the rate at which requests are sent to the server by clicking on ‘loadDriver’ and adjusting the ‘Number of Threads’ and/or ‘Loop Count’ as shown below:

2.2.3 Monitor verbose GC
Once JMeter has started, go to the bottom of the SYSOUT part of the WebSphere server STC. You should start to see new verbose GC messages being written.

2.2.4 Understanding the GC output

Below is shown a sample output written for one GC occurrence and what various parts mean:

```
<gc type="global" id="68" totalid="68" intervalms="551.118">
    <timestamp mark="24.470" sweep="1.413" compact="0.600" total="26.002" />
    <tenured freebits="176051448" totalbytes="268435456" percent="63" />
    <heapfreebytes="157706320" totalbytes="217433088" percent="72" />
    <heapfreebytes="13383968" totalbytes="81002384" percent="16" />
</tenured>

<expansion type="tenured" amount="1048576" newsize="99490304" timetaken="0.016" reason="excessive time being spent in gc" gctimepercent="13" />
```

This type of entry, indicates that Java has decided to increase the heap size because it has detected that too much time has been spend in GC activity.

Further explanation about the GC output can be found in the IBM Developer Kit and
Runtime Environment, Java Technology Edition, Version 6, Diagnosis Guide, which can be downloaded from:


2.2.5 Collect GC output

Use ISPF option 3.2 to allocate a dataset called <yourUserid>.SYSLOG with attributes as shown below:

<table>
<thead>
<tr>
<th>General Data</th>
<th>Current Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management class . . : STANDARD</td>
<td>Allocated cylinders : 26</td>
</tr>
<tr>
<td>Storage class . . . : BASE</td>
<td>Allocated extents . : 1</td>
</tr>
<tr>
<td>Volume serial . . . : DMTP21</td>
<td></td>
</tr>
<tr>
<td>Device type . . . . : 3390</td>
<td></td>
</tr>
<tr>
<td>Data class . . . . : <strong>None</strong></td>
<td>Current Utilization</td>
</tr>
<tr>
<td>Organization . . . : PS</td>
<td>Used cylinders . . : 5</td>
</tr>
<tr>
<td>Record format . . . : VB</td>
<td>Used extents . . : 1</td>
</tr>
<tr>
<td>Record length . . . : 155</td>
<td></td>
</tr>
<tr>
<td>Block size . . . : 15500</td>
<td></td>
</tr>
<tr>
<td>1st extent cylinders: 26</td>
<td></td>
</tr>
<tr>
<td>Secondary cylinders : 9</td>
<td></td>
</tr>
<tr>
<td>Data set name type :</td>
<td>SMS Compressible : NO</td>
</tr>
</tbody>
</table>

| Creation date . . . : 2009/11/05     | Referenced date . . . : 2009/11/17 |
| Expiration date . . : ***None***     |                                     |

Once the JMeter load test has completed, in SDSF, enter DA to see the running STCs, put a question mark beside the STC of the WebSphere server and select SYSOUT. Then enter these commands:

```
print odsn 'yourUserid.syslog'
print
print close
```

From a DOS prompt, use FTP to download the collected output to your PC, an example
is shown below:

```
C:\zIBM-works\projects\techdoc-was-v7-gc>ftp
wtsc55.itso.ibm.com
Connected to wtsc55.itso.ibm.com.
220-FTP Server (user 'edwardmc@au1.ibm.com')
220
User (wtsc55.itso.ibm.com:(none)): edmcar
331-Password:
331
Password:
230-220-FTPMVS1 IBM FTP CS V1R9 at wtsc55.itso.ibm.com,
230-EDMCAR is logged on. Working directory is "EDMCAR.".
230
ftp> get
Remote file 'edmcar.syslog'
Local file was-v7-gc-log.txt
200 Port request OK.
125 Sending data set EDMCAR.SYSLOG
250 Transfer completed successfully.
```

2.3 Analyse GC using Garbage Collection and Memory Visualizer (GCMV)

From the ISA home page, click on Analyze Problems and then select:

IBM Monitoring and Diagnostic Tools for Java™ - Garbage Collection and Memory Visualizer (GCMV)

A display similar to that shown below will appear, just click Next:
The main display for GCMV will then be displayed as shown below:

To start analysing the GC output select File -> Open File, this will bring up a file dialog box, use it to locate the output file. You can then start analysing the GC output. Below is a screen shot showing an example of GC analysis:
2.4 Analyse GC using PMAT

From the ISA home page, click on Analyze Problems and then select:

IBM Pattern Modeling and Analysis Tool for Java Garbage Collector (PMAT)

A display similar to that shown below will appear, just click Next:

PMAT does not run in the ISA workspace, when you click Next, a separate display will appear as shown below:
The initial display for PMAT is similar to that shown below:

```
Select File -> Open verbosegc Files ( IBM SDK ) as shown below:
```

Select File -> Open verbosegc Files ( IBM SDK ) as shown below:
Use the file location dialog box to locate the file you downloaded previously and select it. PMAT will then analyse the file and present a display similar to that shown below:

- **File name**: C:\IBM\works\projects\techdoc-was-v7-gc\was-v7-gc-log.txt
- **Number of verboseGC cycles**: 1
- **Number of Garbage Collections**: 8
- **Number of Allocation failures**: 4

To graphically view the verbose GC output, right click on the file and select ‘Graph View All’ as shown below:
This will produce a display similar to that shown below:
You are now at a point where you can proceed toanalyse the GC output. There are various buttons on the right hand side of the display that will toggle various displays in the graphical area.

In the above display, the buttons that have 'LOA' in their name and the 'GC Completed' button have been selected. The grey line is the 'GC Completed' line. The test application used in this example, created large objects, which were all allocated out of the LOA, or Large Object Area of the heap.

### 2.5 Summary

In this chapter we have shown how to generate verbose GC, collect it, how to transfer it to your PC and shown how to get started with analysis of this output using the two tools. Each tool performs GC analysis in a slightly different way, use the one you like the most or both if needs be.

If this was a real life situation, once you had completed your analysis, you could then consider what JVM adjustments you might make, such as increasing or decreasing the size, or trying different GC options.

The purpose of this chapter was to take you through the “how” of collecting and analysing GC output for WebSphere on z/OS. As mentioned this techdoc is not a tutorial on “how” to analyse GC output, the focus is on getting you to this stage where you can start the analysis.

Becoming competent at analysis of GC output is a skill that requires lots of practice.
3 Heapdump analysis

This chapter will demonstrate how analysis of heapdumps can be done using the following tools:

1. IBM Memory Dump Diagnostic for Java (MDD4J)
2. IBM Monitoring and Diagnostic Tools for Java™ - Memory Analyzer
3. IBM HeapAnalyzer

3.1 Run JMeter

In the initial JMeter window select File -> Open to bring up a display that you can use to locate the file called wasPdTools-causeOutOfMemory.jmx and select it. The display will then look similar to this:

![JMeter screenshot]

This jmx file is a pre-configured load script to run the wasPdTools application.

You need to modify the target TCPIP address and TCPIP port set in this script, do this by clicking on ‘wasV7ZosServer’, and change the values in the ‘Server Name or IP’ and ‘Port Number’ fields.

Once you have made the change, start the load run in JMeter.

When the requests are run, they will create objects but not free them, and will eventually lead to the JVM getting an out of memory condition.
3.2 Collect heapdump

One option would be that after a few minutes, you could issue a command similar to the following to have WebSphere on z/OS generate a heap dump:

F <wasServerId>,heapdump

If you issued such a command, in the SYSOUT part of the servant STC you would see messages similar to this:

JVMDUMP034I User requested Heap dump using '/SYSTEM/var/WebSphere/ /home/WYCell/WYSRVG/heapdump.20091117.015523.16777288.0001.phd' through JVMRI

JVMDUMP010I Heap dump written to /SYSTEM/var/WebSphere/ /home/WYCell/WYSRVG/heapdump.20091117.015523.16777288.0001.phd

In this exercise however, after several minutes, the JVM will get an out of memory condition, WebSphere will automatically generate a heapdump and the servant STC will be terminated and a new one automatically restarted.

The heapdump will be written to the home directory of the userid the WebSphere servant STC is running under. You can determine what this directory will be by first determining what userid the servant STC is running under, the SDSF DA display will show that, for example:

SDSF DA MVST TESTMVS PAG 0 CPU/L 0/ 0
NP JOBNAME StepName ProcStep JobID Owner
WYSR01AS WYSR01AS BBOPASR STC06991 WYASRU
WYSR01A WYSR01A BBOPACR STC06981 WYACRU
WYSR01AA WYSR01AA BBOPCRA STC06987 WYASRU

The above display shows that the servant region, WYSR01AS is running under the userid WYASRU. Then issue the following command:

tso lu wyasru omvs

In the output displayed will be lines similar to the following showing the home directory for the userid:

...
Use FTP to download the heapdump file to your PC. Note be sure to use the binary transfer method.

### 3.3 Heapdump analysis using IBM Memory Dump Diagnostic for Java (MDD4J)

From the ISA home page, click on Analyze Problems and then select:

**IBM Memory Dump Diagnostic for Java (MDD4J)**

A display similar to that shown below will appear, just click Next:

![Tool Input Parameter Values](image)

The main display for MDD4J will then be displayed as shown below:
Select ‘Single Heapdump Analysis’ and click the Browse button, this will bring up a file dialog window, use it to locate your heapdump and select it. Then click the Analyze button which will produce a display similar to this:
Clicking on the ‘Reach Size Leak Suspects’ tab brings up a report which tries to identify what objects are contributing to a memory leak. In our case, the report identifies, as expected our application. It lists the HeapeaterBigObject as a leak suspect, as shown below:

<table>
<thead>
<tr>
<th>Leaking Container Object Type</th>
<th>Leak Unit Object Type</th>
<th>Number of Instances</th>
<th>Heap Size of Leak Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tools.HeapEaterBigObject</td>
<td>java.util.Vector</td>
<td>1</td>
<td>457MB</td>
</tr>
<tr>
<td>java.util.Vector</td>
<td>java.lang.StringBuffer</td>
<td>2355</td>
<td>457MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Leaking Container Object Type</th>
<th>Leak Unit Object Type</th>
<th>Number of Instances</th>
<th>Heap Size of Leak Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.ws.session.store.memory.MemoryStore</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java.util.HashMap</td>
<td>com.ibm.ws.session.store.memory.MemorySession</td>
<td>13384</td>
<td>6.32MB</td>
</tr>
</tbody>
</table>
In this case we know this is from our application so we have identified that this application is using a large amount of the heap.

MDD4J provides a number of features to help you analyse the contents of the heapdump.

3.4 HeapDump analysis using IBM Monitoring and Diagnostic Tools for Java™ - Memory Analyzer

From the ISA home page, click on Analyze Problems and then select:

IBM Monitoring and Diagnostic Tools for Java™ - Memory Analyzer

A display similar to that shown below will appear, just click Next:

![Tool Input Parameter Values](image)

The main display for Memory Analyzer will then be displayed as shown below:
Select File -> Open Heap Dump, this will bring up a file dialog window, use it to locate your heapdump and select it. Analysis of the dump will then automatically commence, and eventually a display similar to that shown below will appear:
Take the default option of the ‘Leak Suspects Report’ and click Finish Then click the Analyze button which will produce a display similar to this:

This display shows that Memory Analyzer has indentified that the HeapEaterBigObject is occupying over 479M of the heap. In this case we know this is from our application so we have identified that this application is using a large amount of the heap.

Memory Analyzer provides a number of features to help you analyse the contents of the heapdump.

### 3.5 Heapdump analysis using AlphaWorks HeapAnalyzer

From the ISA home page, click on Analyze Problems and then select:

Heap Analyzer

A display similar to that shown below will appear, just click Next:
HeapAnalyzer does not run in the ISA workspace, when you click Next, a separate display will appear as shown below:

3.5.1 Using HeapAnalyzer

Select File -> Open and use the file dialog box to locate the heapdump file you downloaded.

When you select the heapdump file, a progress bar will be displayed showing you that HeapAnalyzer is analysing the file. Once this analysis completes, a summary window is displayed, which will be similar to that shown below:
The typical approach is to try and identify what object is using up a large proportion of the heap. In this case, the initial display is enough to identify the cause of the out of memory condition. We can see that heapAnalyzer has highlighted a particular row. This row indicates that approximately 479M of the heap is associated with this object, and that it contains 20504 other objects. The row also indicates that this object accounts for 89% of the heap.

The reason it is highlighted is because there is a large drop in size between this row and the next row, which can be seen by clicking on the small symbol at the start of the row as shown below:
A large drop like this typically indicates a problem, in that some object is using up large amounts of the heap, because it contains a large number of smaller objects.

The object name of java/lang/Object does not however readily identify who owns this object. However by looking back up the tree, we can see that two rows up is an object called Tools/HeapEaterBigObject. Since Tools/HeapEaterBogObject is not a standard Java class, it is clearly an object related to an application. In this case we know this is from our application so we have identified that this application is using a large amount of the heap.

### 3.6 Summary

In this chapter we have shown how to collect a heapdump generated by WebSphere on z/OS, how to transfer it to your PC and shown how to get started with analysis of the heapdump using the three tools. Each tool performs heapdump analysis in a slightly different way, use the one you like the most or all three if needs be.

If this was a real life situation, once you had completed your analysis and identified the suspect application, you would then take this evidence off to the owner of the application and ask them to investigate why their application is using such a large amount of the heap.

The purpose of this chapter was to take you through the “how” of collecting and analysing a heapdump in WebSphere on z/OS. Its purpose is not to school you in the technique of heapdump analysis. Real world situations would typically not be as obvious as this simple example. Becoming proficient at heapdump analysis is a skill that is typically gained through much experience.
4 javacore analysis

This chapter will demonstrate how to use the IBM Thread and Monitor Dump Analyzer for Java tool to analyse a javacore.

4.1 Run JMeter

We will use JMeter to run a servlet called Sleeper.

Select File -> Open to bring up a display that you can use to locate the file called wasPdTools-Sleeper.jmx and select it.

Once the file is opened in JMeter, modify the target TCPIP address and TCPIP port set in this script, do this by clicking on ‘wasV7ZosServer’, and change the values in the ‘Server Name or IP’ and ‘Port Number’ fields to reflect the location of the WebSphere server you are using.

Once you have updated these fields, click on Run -> Start to have JMeter start executing the load script.

4.2 Collect javacores

As soon as JMeter starts running the load script, issue the following command via SDSF:

```
F <wasServerId>,javacore
```

This command will cause WebSphere to write a javacore to the home directory. You will see messages detailing were the javacore has been written to in the STC log, it will look similar to this:

```
JVMDUMP034I User requested Java dump using
'/SYSTEM/var/WebSphere/home/WYCell/WYSRVG/javacore.20091117.051119.16777301.0001.txt' through JVMRI
```

Issue the above command three more times, thirty seconds apart.

Then FTP the four javacore files down to your PC, using the ASCII option.

4.3 javacore’s analysis using IBM Thread and Monitor Dump Analyzer

From the ISA home page, click on Analyze Problems and then select:
The IBM Thread and Monitor Dump Analyzer for Java (TMDA)

The main display for TMDA will then be displayed as shown below:

Select File -> Open and use the file dialog box to locate the javacore files you downloaded. Select all the javacore files in one go by holding down the control key while selecting the files as shown below:
Click Open and you will then get a display similar to this:

![Thread Dump List](image)

Hold the control key down and select all the javacore files in the Thread Dump List, and then right click, and select Compare Threads as shown below:
This option then produces a display that shows information from threads running in the WebSphere server from all the javacore dumps as shown below:

The threads called 'WebSphere:ORB.thread.pool...' are the threads in the WebSphere JVM that run servlets requests. You will see many other threads in the list, these run various others parts of a WebSphere server.

Typically you will look at these WebSphere ORB threads. In a situation where the complaint is of poor response, what you are looking for are threads that are running the same request across multiple javacore dumps. Finding one or more threads running the same application request across the time period that the javacores were taken across, identifies an application that probably needs to be investigated as to why it is taking so
In the above example, we can see that the thread is in a wait caused by Thread.sleep, which has been issued by the Tools/Sleeper servlet from the doPost method. Checking other threads will show that a number of them are in the same wait state.

4.4 Summary

In this chapter we have shown how to generate a javacore, how to transfer it to your PC and shown how to get started with analysis.

If this was a real life situation, the analysis may identify an application which has requests that are running for a long period of time. The java stack trace on the thread gives you an indication of what the request is doing at the time the javacores were taken. You can then discuss this with the application developers to try and determine why the request is taking a long time.

The purpose of this chapter was to take you through the “how” of collecting and analysing a javacore in WebSphere on z/OS. Its purpose is not to school you in the technique of javacore analysis. Real world situations would typically not be as obvious as this simple example. Becoming proficient at javacore analysis is a skill that is typically gained through much experience.