

AVP-3225

Java Troubleshooting with ISA 5 using Thread and Monitor Dump Analyzer

Java Troubleshooting with ISA 5 using Thread and Monitor Dump Analyzer

- What this lab is about..... 2
- Lab requirements 3
- What you should be able to do 3
- Part 1:Lab Set Up 4
- Part 2:IBM Support Assistant 5 Beta 5
- Part 3:Advanced Javacore Analysis with IBM Thread and Monitor Dump Analyzer Tool20
- Part 4:(Optional) Mapping a Thread Id in a Javadump to the output of WebSphere logging or trace . 57
- Reference Links 61

What this lab is about

This lab is provided **AS-IS**, with no formal IBM support.

In this lab you will use IBM Support Assistant 5 Beta (ISA 5) to diagnose JVM issues experienced by a WebSphere Application Server 8.5.

A badly implemented web application will be used to simulate a hung thread scenario. Diagnostic data such as Javadumps (javacore.txt files) and system dumps will be used to determine the code causing the issue, and inspect the Java objects associated with the hung thread. The lab will also explain how to correlate thread id between the various logs and dumps.

The lab demonstrates how ISA 5 facilitates team team-based collaboration, and provides server-level diagnostic tools to carry out analysis. The ISA tools “Thread and Monitor Dump Analyzer (TMDA)” and “Memory Analyzer” will be used.

Lab requirements

List of system and software required for the attendee to complete the lab.

- WebSphere Application Server V8.5.0.1
- IBM Support Assistant V5 (Beta 2) with the following tools installed:
 - ▶ WebSphere Application Server Configuration Visualizer
 - ▶ Thread Dump and Monitor Analyzer – (ISA 5 desktop tool, and report)
 - ▶ IBM Monitoring and Diagnostic Tools for Java™ - Memory Analyzer – (ISA 5 desktop tool, and report tool)
 - ▶ IBM Extensions for Memory Analyzer (Packaged with the Memory Analyzer ISA 5 desktop tool)

What you should be able to do

At the end of this lab you should be able to:-

- Use ISA 5 to manage Java diagnostic data and utilize problem determination tools
 - Diagnose a hung WebSphere thread using Thread and Monitor Dump Analyzer for Java
 - Correlate thread information between WebSphere log, traces, Javadumps and system dumps
-

Part 1: Lab Set Up

_____ Login to the VMWare image with the username/password below:

Username :Administrator

Password : Impact2013AVP

NOTE:

Due to the physical memory on the VMware image being used for this lab please understand that certain operations may take time to perform – please be patient. The expected duration of this lab is **55 minutes** and each part has an estimated duration. The final lab section is marked as **optional** so you can attempt this if there is sufficient time.

Part 2: IBM Support Assistant 5 Beta

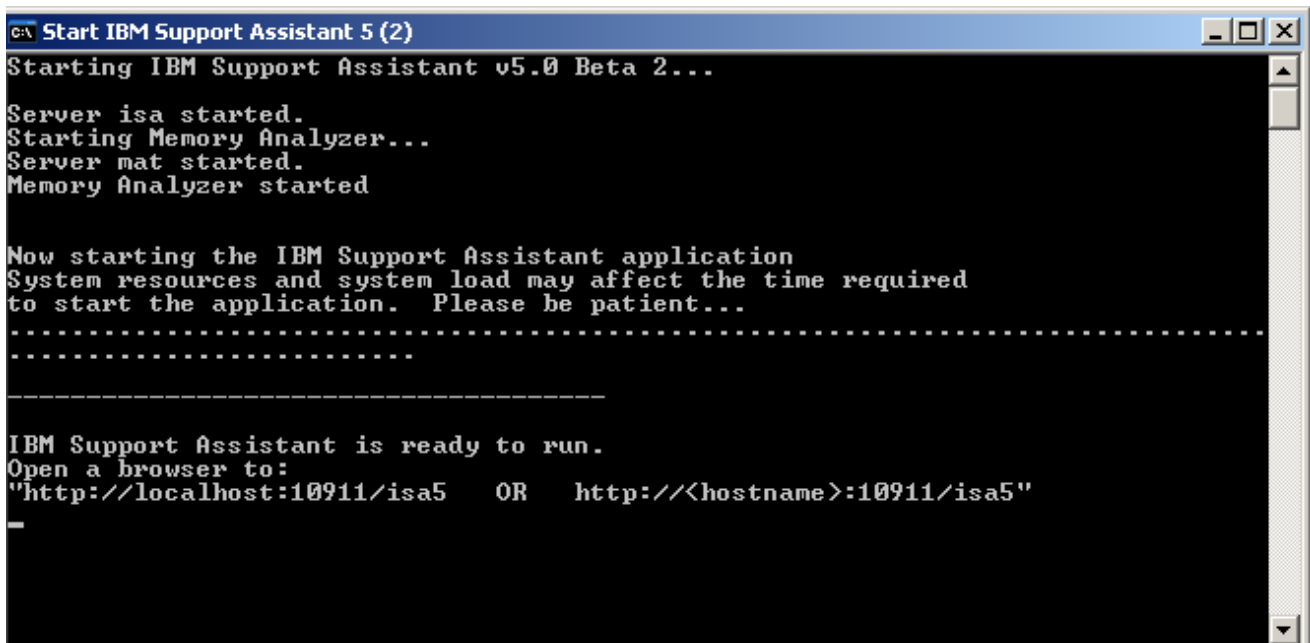
Note:

The IBM® Support Assistant (ISA) is a free application that provides features for problem determination, and a platform for obtaining diagnostic tools. The most recent release of ISA (version 5.0), which is currently in beta, brings these capabilities into a server environment. This enables an administrator to install a single instance of ISA that can be used by a group of users and accessed via a web browser. Therefore resources, files, information, and server-level tools can be shared. This facilitates team based collaboration and avoids the need for each team member to install diagnostic tools on their local workstation.

ISA v5.0 can be installed from an EAR file into an existing WebSphere Application Server, or using a simple “all-in-one” unzip install which contains everything required, including a lightweight application server and Java runtime. This lab uses the latter approach.

In this part of the lab, you will use ISA to understand its key concepts, and test some of the core functions.

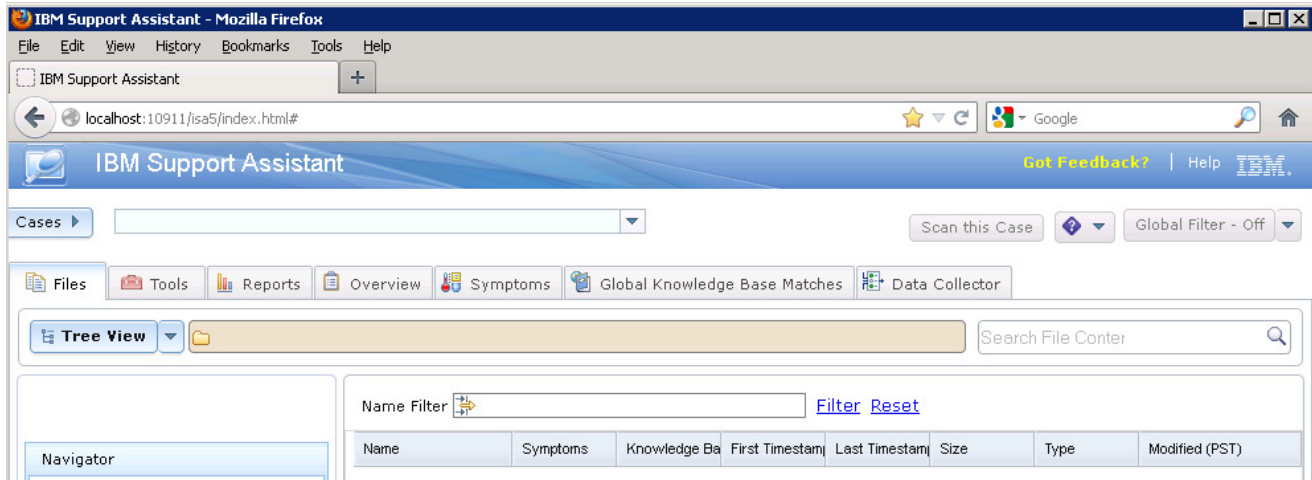
_____ Validate that ISA is fully started. ISA was started in a command prompt window. Check that window and look for “IBM Support Assistant is ready to run”. If this is not displayed wait for the startup to complete.



```
C:\> Start IBM Support Assistant 5 (2)
Starting IBM Support Assistant v5.0 Beta 2...
Server isa started.
Starting Memory Analyzer...
Server mat started.
Memory Analyzer started

Now starting the IBM Support Assistant application
System resources and system load may affect the time required
to start the application. Please be patient...
.....
-----
IBM Support Assistant is ready to run.
Open a browser to:
"http://localhost:10911/isa5 OR http://<hostname>:10911/isa5"
```

____ Launch the browser and use the bookmarks to load the ISA web interface.

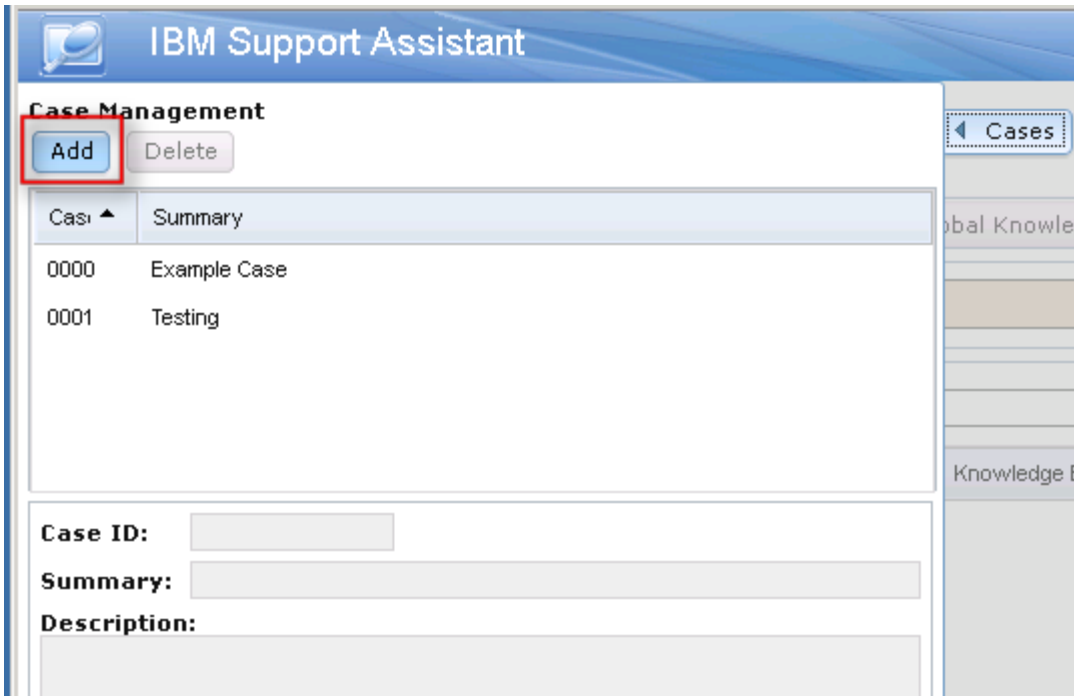


Note:

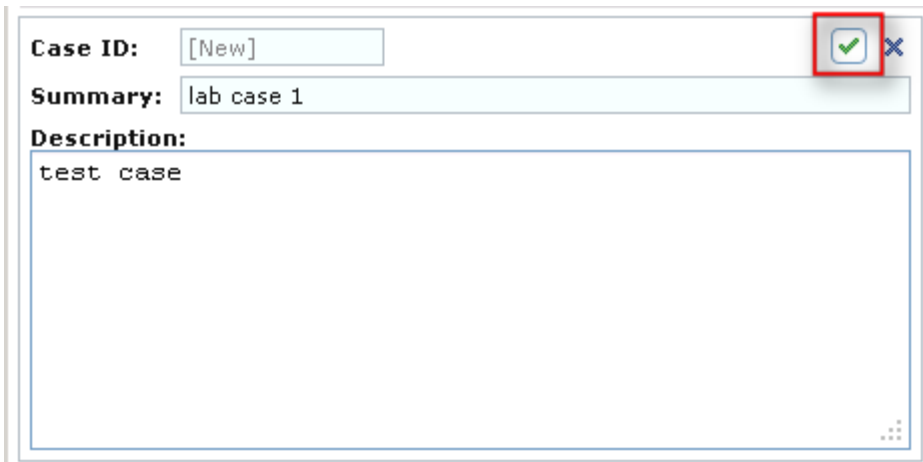
Like previous releases of ISA, the tool can be used to find problems, analyze data, and send information to IBM support. As it is a multi-user installation, ISA provides a case management component to help manage various problem determination activities that a team might require. A case is simply a container for a logical grouping of files and information. A typical practice would be to group artifacts pertaining to a single issue.

____ Create a new case by clicking the **Cases** button, and then **Add**.

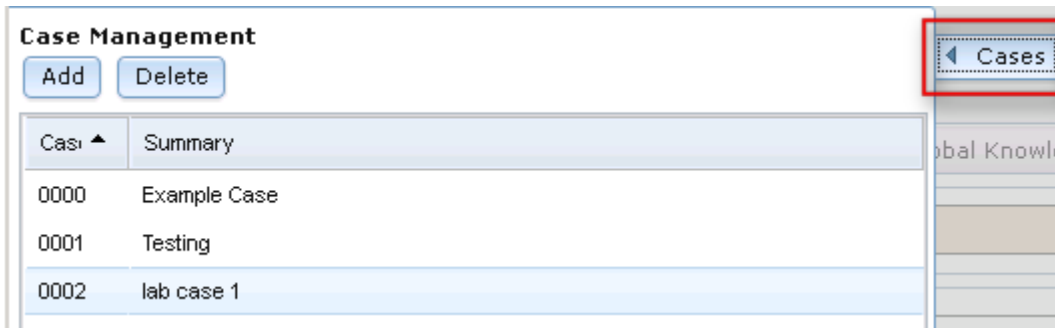




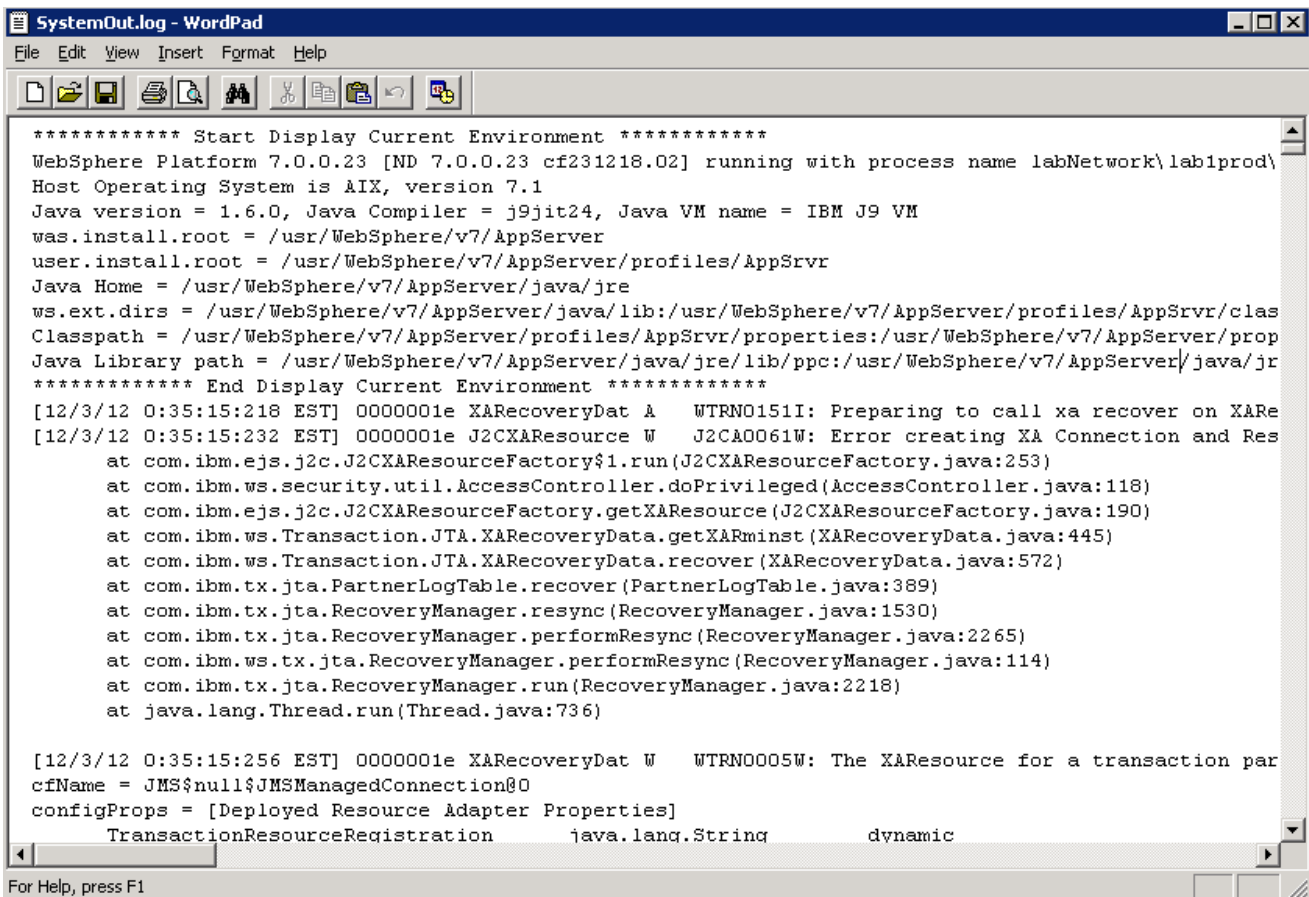
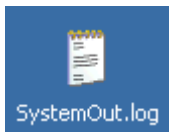
_____ Complete the summary and description, and click the **green** tick.



_____ Shrink the cases dialog by clicking **Cases**.

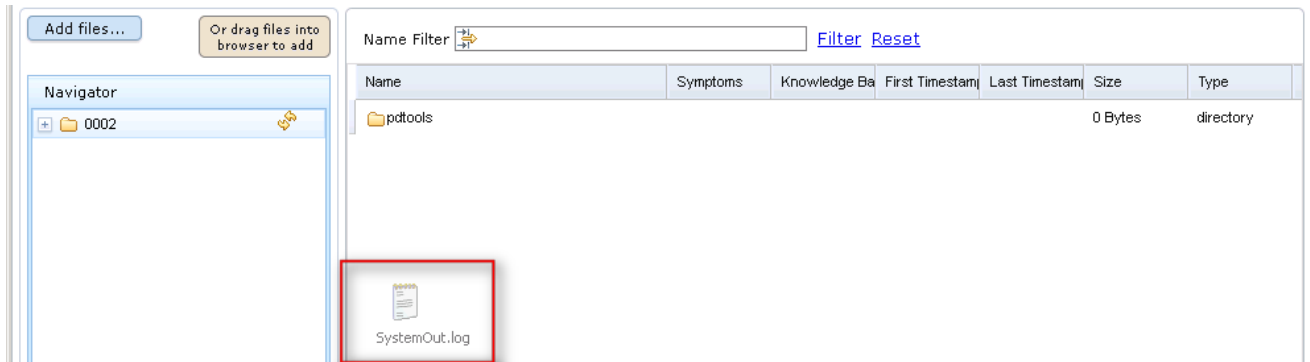


Having made a new case, diagnostic data can be added. This lab provides a sample WebSphere Application Server SystemOut log file that contains some error messages. Open the file on from the desktop, and note that it contains various WebSphere errors and Java stack traces.

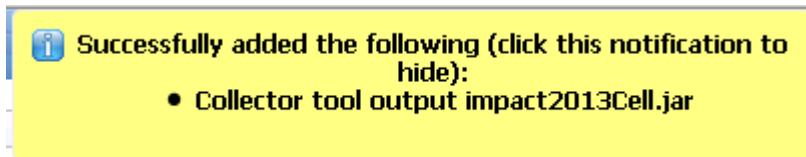


Close the text editor.

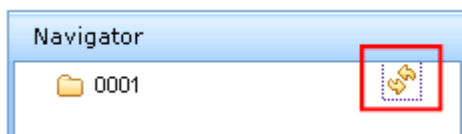
_____ Add this sample SystemOut log to the ISA case by dragging it from the desktop to the ISA file list in the browser.



_____ Click the **yellow box** to dismiss the notification.

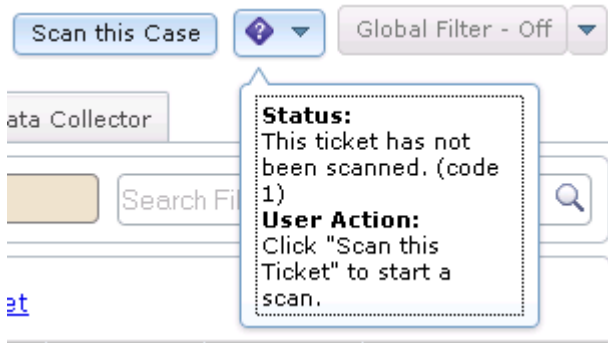


_____ Click the **refresh** icon in the Navigator to see the files in the case.



_____ Click the **Status** button. Notice the "Ticket" (case) has not been scanned.





_____ Click the **Scan this Case** button.

Problem Analysis x

Run Automated Analysis

Input Files and Folders

/ISA5Beta2/ISA5/isa/cases/0002

Parameters

Parameter	Description	Value
force_rescan	This parameter controls the type of scan. Default is an incremental scan, scanning only new files. Check here to force a re-scan of ALL THE FILES within this case.	<input type="checkbox"/>
ignore_failed	This parameter controls handling of previous failures. By default, a scan will not run if a previous scan has failed. Check here to force a scan regardless of the previously failed one. Note: Even with this setting, the scan will be terminated if it runs longer than 1 hour.	<input type="checkbox"/>

Run as background task:

_____ Click **Submit** to start the Scan.

Note:

Scanning the case may take up to 5 minutes. In the meantime, let's investigate the tools that are installed into ISA 5.

_____ Click the **Tools** tab. Note that only the tools that will be used in this lab have already installed, more tools are available.

Note:

There are three key types of tools provided in ISA and each type has its benefits and compromises.

Report generator tools - process input data (e.g. log files or Java dumps) and generate a simple output file, usually in the form of an HTML or .txt report. These tools are not interactive but they are simple to run and have the benefit of consuming no local resources.

Web-based tools - run most of their analysis processing on the ISA server and provide a rich, interactive experience in the browser based user interface. These types of tools are ideal for activities where you want to off-load heavy processing of files to a more powerful server.

Desktop tools - typical desktop client-side applications that are launched via the ISA browser UI. By leveraging Java WebStart, the entire tool will be installed and run locally on your desktop. This type of tool has a few drawbacks - a Java plugin is required for your browser, local system resources are required to run the client tool, and you must have local access to the files you wish to analyze. However, some ISA tools are only available as desktop tools.

The screenshot shows the IBM Impact2013 interface with the following elements:

- Navigation Tabs:** Files, Tools (selected), Reports, Overview, Symptoms, Global Knowledge Base Matches, Data Collector.
- Search Bar:** "Enter keyword" with "Filter" and "Reset" buttons.
- Sort and Filter:** "Sort By: a-z" and "Tag: All Tags".
- Tool List:**
 - Garbage Collection and Memory Visualizer (GCMV) [Desktop] ✓ [J]
 - Garbage Collection and Memory Visualizer (GCMV) [Report] ✓ [Bar]
 - Health Center ✓ [J]
 - HeapAnalyzer [Desktop] [Wrench] [J]
 - Memory Analyzer [Desktop] ✓ [J]
 - Memory Analyzer [Report] ✓ [Bar]
 - Memory Analyzer Web Edition [Web] [Wrench] [Globe]
 - Thread and Monitor Dump Analyzer (TMDA) [Desktop] [Wrench] [J]
 - Thread and Monitor Dump Analyzer (TMDA) [Report] [Wrench] [Bar]
 - WebSphere Application Server Configuration Visualizer [Wrench] [Bar]
- Right Panel:** "Select a tool from the list to display ..." and a link "Read more about additional tools available to in..."

Note:

By now the scanning of the case should be finished, let's check the results.

Click the **Overview** tab. It shows useful system information determined from the log file(s) in the case. (If the scan is not completed nothing will display and you will have to repeat this step until data is ready).

Click the **Symptoms** tab. It shows a list of the errors encountered in the log file(s).

Click the **Global Knowledge Base** tab. This compares the symptoms to a local database (XML file) to suggest possible resolutions including APARs (IBM fix references) and Technotes. Click on a suggestion to see a detailed description below.

The screenshot shows the IBM Support Assistant web application in a Mozilla Firefox browser. The address bar shows the URL: localhost:10911/jsa5/index.html#id=0002. The page title is "IBM Support Assistant".

The interface includes a navigation bar with tabs for Files, Tools, Reports, Overview, Symptoms, Global Knowledge Base Matches, and Data Collector. The "Symptoms" tab is active, and a search filter is applied: "[0002] lab case 2".

The search results table shows 17 of 22 results. The table has columns for Global Score, Type, Knowledge Base Entry, Symptom, Tool, and ID. The results are as follows:

Global Score	Type	Knowledge Base Entry	Symptom	Tool	ID
	APAR	PM15719: THE TRANSACTION MANAGER FAILS TO GET AN XARESOURCE TO ROLLBACK THE TRANSACTION.	Multiple symptoms (2) matched by this entry	LocalKBSe	9
	APAR	PK91826: CSCP0007E, WTRN0005W THE XARESOURCE FOR A TRANSACTION PARTICIPANT COULD NOT BE RECREATED.	WTRN0005W: The XAREsource for a transaction participant could not be recreated and transaction recovery may not be able to complete properly. The resource was J2CXAResourceInfo :	LocalKBSe	16
	APAR	PK81814: MESSAGE "OPEN FOR E-BUSINESS" IS NOT BEING PRINTED IN SYSTEMOUT.LOG	CWZZZ0001W: Possible abnormal startup - did not find 'open for e-business'	LocalKBSe	1
	APAR	PK83560: WLM RETURNING TARGET WITH EMPTY ENDPOINTS CAUSES SIB EXCEPTIONS RESULTING IN CWSIT0019E OR CWSIA0241E ERRORS	CWZZZ0001W: Possible abnormal startup - did not find 'open for e-business'	LocalKBSe	3

Below the table, the "Knowledge Base Matches" section is expanded for the selected match (PK83560). It shows the following details:

- Type:** APAR
- Found by Tool:** LocalKBSearch
- Global Score:** 1846
- Label:** PK83560: WLM RETURNING TARGET WITH EMPTY ENDPOINTS CAUSES SIB EXCEPTIONS RESULTING IN CWSIT0019E OR CWSIA0241E ERRORS
- Match ID:** 3
- Symptom IDs associated with this Match:** 85
- Description:** WLM RETURNING TARGET WITH EMPTY ENDPOINTS CAUSES SIB EXCEPTIONS RESULTING IN CWSIT0019E OR CWSIA0241E ERRORS.

The footer of the page includes the Build ID: 5.0.0.0_Beta2_20121016-1409 and the copyright notice: © Copyright IBM Corp. 2011, 2012. All rights reserved.

_____ To demonstrate one of the report generator tools, first switch to the “PlantsByWebSphere” case that has been prepared for this lab by selecting it from the dropdown.

The screenshot shows the IBM Support Assistant interface with the "Cases" dropdown menu open. The dropdown list contains the following items:

- [0003] PlantsByWebSphere
- [0000] Example Case
- [0002] lab case 2
- [0003] PlantsByWebSphere

The first and last items in the list, "[0003] PlantsByWebSphere", are highlighted with a red rectangular box. The current selection in the dropdown is "[0003] PlantsByWebSphere".

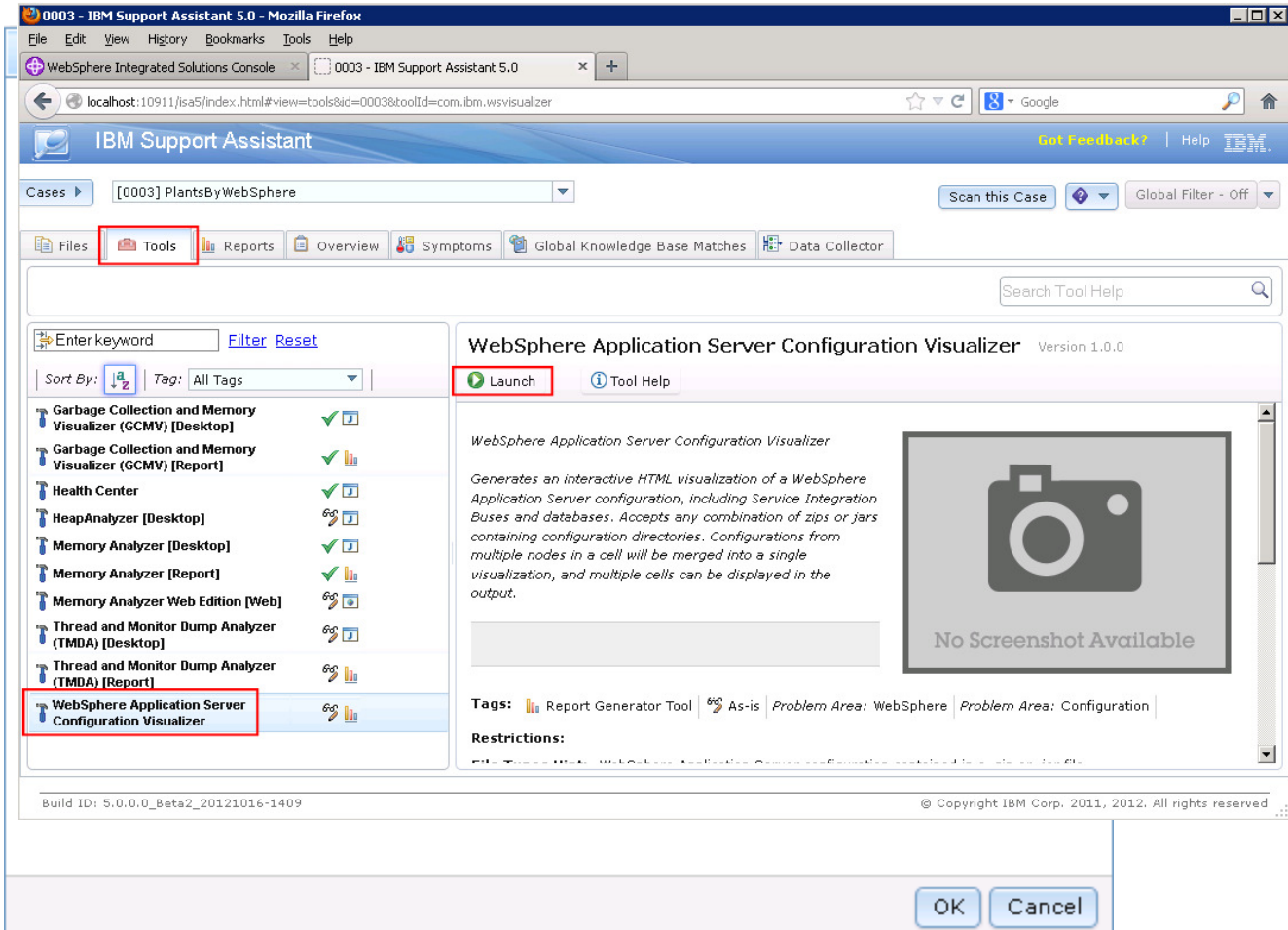
The screenshot shows the IBM Support Assistant interface. At the top, the title bar reads 'IBM Support Assistant'. Below it, a 'Cases' dropdown menu is set to '[0003] PlantsByWebSphere'. A 'Scan this Case' button is visible. The main navigation bar includes 'Files', 'Tools', 'Reports', 'Overview', 'Symptoms', 'Global Knowledge Base Matches', and 'Data Collector'. The 'Tree View' section shows a folder structure with 'CASE:0003/*'. A 'Navigator' pane on the left shows a folder '0003'. A table below the navigator displays the following file:

Name	Symptoms	Knowledge Ba	First Timestam	Last Timestam	Size	Type
Collector tool output impact2013Cell.jar					6 MB	jar

Note:

The case contains file “Collector tool output impact2013Cell.jar”. This is output from the WebSphere collector tool which collects information about the WebSphere Application Server configuration – it is frequently requested by IBM support when investigating PMRs.

_____ To demonstrate one of the report generator tools, return to the Tools tab and select **WebSphere Application Server Visualizer**. Then press the launch button.



_____ Navigate to the file **Collector tool output impact2013Cell.jar** select it and press OK.

Problem Analysis x

Run WebSphere Application Server Configuration Visualizer (Version 1.0.0)

Input Files and Folders *

|ISA5Beta2/ISA5/isa/cases/0003/Collector tool output impact2013Cell.jar
Browse

Parameters

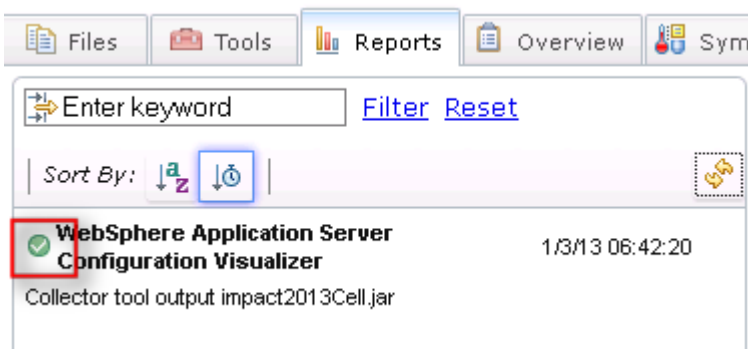
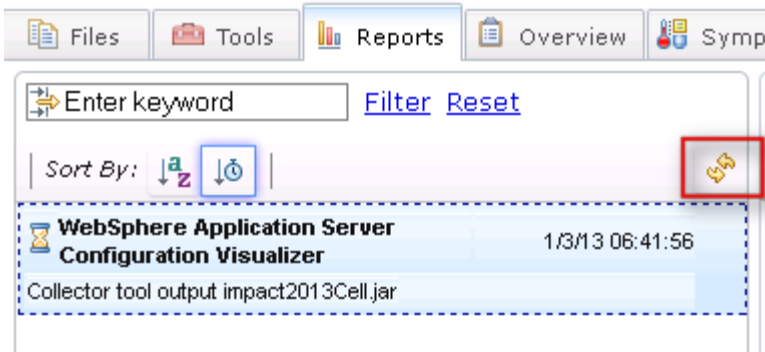
Parameter	Description	Value
ReportName	Report Name	<input style="width: 80%;" type="text" value="topology.html"/>

Run as background task:

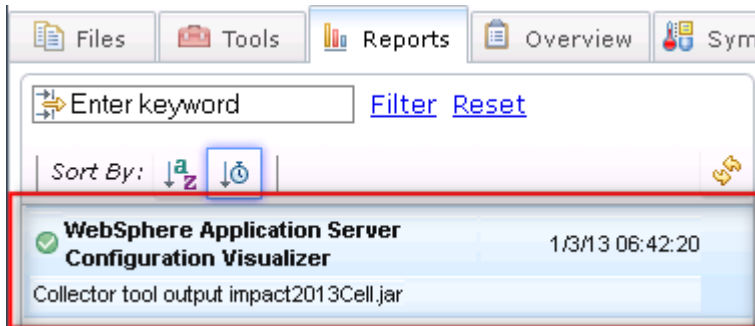
Submit
Cancel

_____ Click **Submit**

_____ Click on the **Reports** tab, then click the **Refresh** icon a few times until the report is complete, as indicated by a green tick icon.



_____ Select the completed report.

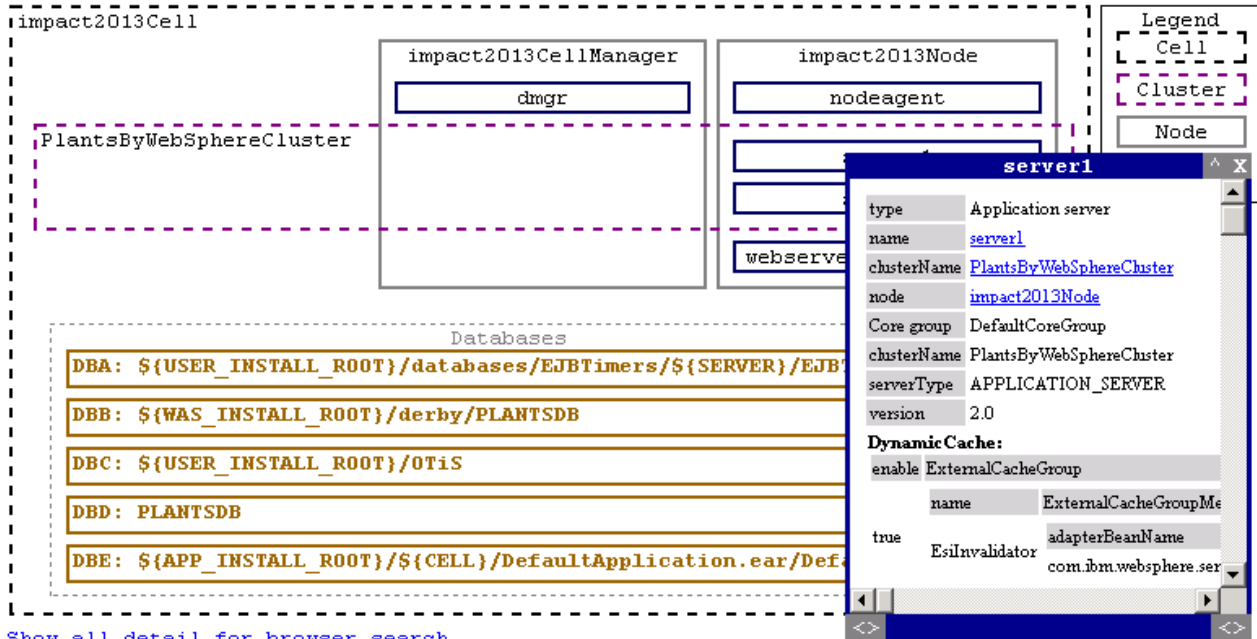


Note:

Take a look at the report in the right hand window. This visualizes the topology of WebSphere Application Server. For example, in this lab the topology consists of a single cell containing two nodes – one containing the deployment manager, and the second containing two application servers. Those servers are part of a cluster called “PlantsByWebSphereCluster”. In addition, there are several datasources defined in the configuration.

_____ Try clicking on any of the topology components, the report is interactive and will pop up dialogues with more detailed information.

WebSphere Application Server Configuration Visualizer



[Show all detail for browser search](#)

Part 3: Advanced Javacore Analysis with IBM Thread and Monitor Dump Analyzer Tool

Note:

Java dumps (ala javacore files or thread dumps) are useful for analyzing hung thread scenarios. In this section, you will use a javacore file help debug a “Possible Hung Thread” warning in the WebSphere log file. The following ISA 5 tools will be used:

IBM Thread and Monitor Dump Analyzer – used to interpret Javacore files

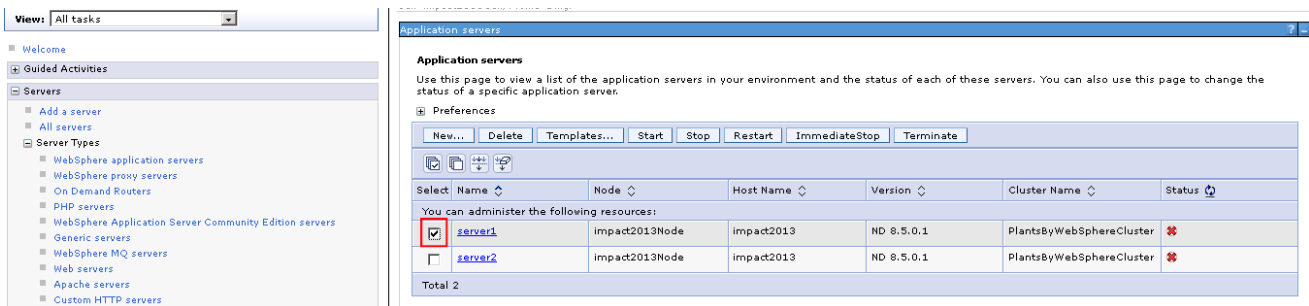
Memory Analyzer – used to inspect the Java objects associated with the hung thread

IBM Extensions to Memory Analyzer– provides addition functions to correlate thread Ids in Javacore files with other WebSphere traces

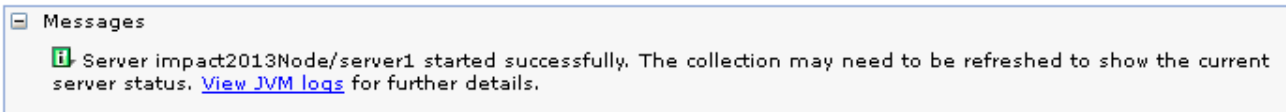
_____ Launch the browser and use the bookmarks to load the WebSphere Integrated Solutions Console (admin console) in a new tab.

_____ Login using a blank user name.

_____ Navigate to **Servers->Server Type->WebSphere Application Servers**. Select **server1**.



_____ Press the Start button and wait for the admin console to report the server has started.



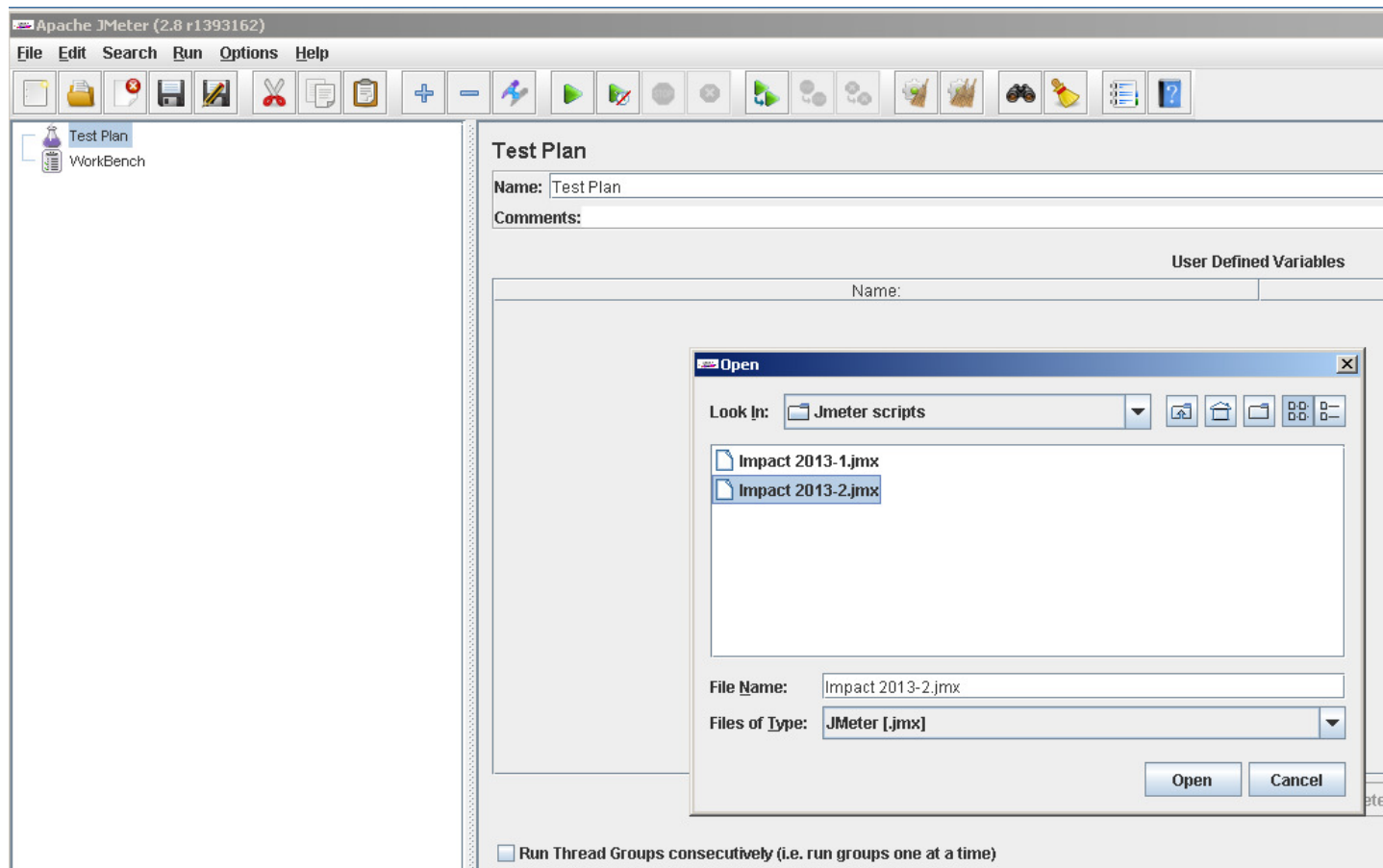
Note:

Next you will use the Jmeter load generating tool to simulate some user requests to the Plants web application. Some of these user requests will trigger a deliberate error which you will diagnose using ISA 5 tools.

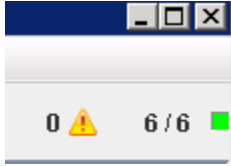
_____ Double click the **Jmeter** shortcut on the desktop.



_____ Click **File->Open** and navigate to **E:\Impact lab files\Jmeter scripts\Impact 2013-2.jmx**. Click **Open**.



_____ Click **Run->Start**. Wait a few moments until the number of threads has reached 6, as indicated in the right hand corner of the Jmeter window.



_____ From the desktop, double click the **Tail** icon.



_____ Close any open files in Tail.

_____ Use Tail to monitor the SystemOut.log of the **server1**. The file is located at:

C:\IBM\WebSphere\AppServer\profiles\AppSrv01\logs\server1

_____ Monitor the file for a hung thread warning message similar to that shown below. It may take up to 3 minutes to appear (you may read the “Note:” section below while you wait). Make a note of the thread title, e.g. “WebContainer : n” – the actual value of “n” may vary with the screenshots in this lab document.

Note: Be sure to watch the tail carefully as the error will scroll off the screen.

```
SystemOut.log
[2/11/13 6:09:55:578 PST] 000000a7 SystemOut      O ==> STARTING SLOW METHOD
[2/11/13 6:09:55:578 PST] 000000a7 SystemOut      O timestamp=1360591795578
[2/11/13 6:09:55:578 PST] 000000a7 SystemOut      O resume at=1360591805578
[2/11/13 6:09:58:015 PST] 00000097 SystemOut      O ==> ENDING SLOW METHOD
[2/11/13 6:09:58:125 PST] 000000aa HtmlGridRende W   PanelGrid shopping:j_id164039580_461abe83 has not enough children. Child count
[2/11/13 6:09:58:156 PST] 00000097 SystemOut      O performProductDetail : itemID=F0018
[2/11/13 6:09:58:156 PST] 00000097 SystemOut      O ==> STARTING DELIBERATE LARGE SESSION
[2/11/13 6:09:58:156 PST] 00000097 SystemOut      O ==> ENDING DELIBERATE LARGE SESSION
[2/11/13 6:10:05:578 PST] 000000a7 SystemOut      O ==> ENDING SLOW METHOD
[2/11/13 6:10:07:218 PST] 00000073 ThreadMonitor W   WSVR0605W: Thread "WebContainer : 0" (00000096) has been active for 147406 mil
    at java.lang.Thread.sleep(Native Method)
    at java.lang.Thread.sleep(Thread.java:896)
    at com.ibm.websphere.samples.pbw.war.ShoppingBean.deliberateHungThread(ShoppingBean.java:415)
    at com.ibm.websphere.samples.pbw.war.ShoppingBean.performProductDetail(ShoppingBean.java:182)
    at sun.reflect.GeneratedMethodAccessor166.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:37)
    at java.lang.reflect.Method.invoke(Method.java:611)
    at org.apache.webbeans.intercept.InterceptorHandler.invoke(InterceptorHandler.java:287)
    at org.apache.webbeans.intercept.NormalScopedBeanInterceptorHandler.invoke(NormalScopedBeanInterceptorHandler.java:98)
    at com.ibm.websphere.samples.pbw.war.ShoppingBean $$javassist 1.performProductDetail(ShoppingBean $$javassist 1.java)
```

Note:

WebSphere Application Server has a built in heuristic for detecting threads that could be potentially hung. To speed up this lab, some custom properties have been set to influence the behavior of the hung thread detection with the goal of making it trigger more quickly than normal. In addition, automatic generation of Javadumps on hung thread detection has been configured. The properties configured for the lab are summarized in the table below. No action is required.

Custom WebSphere Property	Description	Default	Lab Setting
com.ibm.websphere.threadmonitor.interval	The frequency, in seconds, at which managed threads in the selected application server will be interrogated	180 seconds	60 seconds
com.ibm.websphere.threadmonitor.threshold	The length of time, in seconds, in which a thread can be active before it is considered hung. Any thread that is detected as active for longer than this length of time is reported as hung	600 seconds	120 seconds
com.ibm.websphere.threadmonitor.false.alarm.threshold	The number of times that false alarms can occur before automatically increasing the threshold	100	0 (disabled)
com.ibm.websphere.threadmonitor.dump.java	Set to true to cause a javacore to be created when a hung thread is detected and a WSVR0605W message is printed	false	True

_____ After the hung thread warning has been made in the SystemOut log, switch to directory “C:\IBM\WebSphere\AppServer\profiles\AppSrv01”. Verify that a Javacore.txt file has been automatically written by WebSphere Application Server in response to the hung thread notification.

_____ Stop the Jmeter script by clicking **Run->Stop**. Close the Jmeter window.

Note:

When analyzing hung threads or other performance problems, it is often useful to have several Javadumps over a period of a few minutes. This can show that while a thread has been active for a long time, its current stack frame is changing meaning it is still busy processing its work, and actually not hung at all.

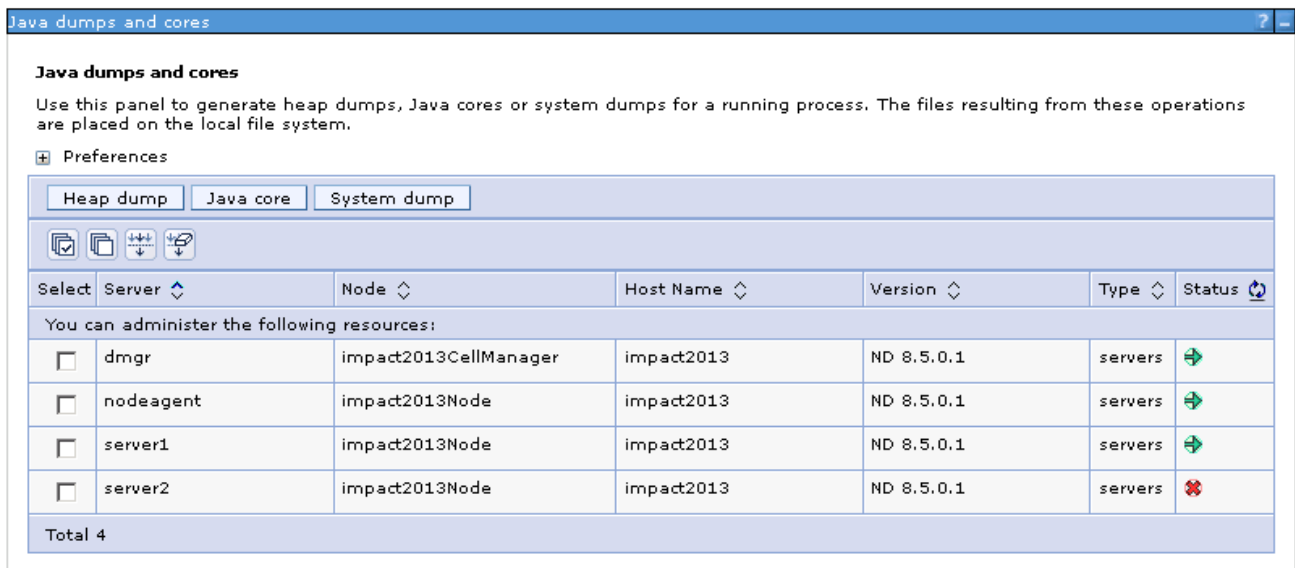
In the next steps you will request the JVM to write a 2nd Javacore, and a system dump. There are several ways to do this including:

1. wsadmin scripting interface
2. Health Center tool
3. WebSphere Administration Console (for WebSphere Application Server 8.0 and above)

In this lab, you will use the WebSphere Administration Console.

_____ Launch the browser and use the bookmarks to load the WebSphere Integrated Solutions Console (admin console).

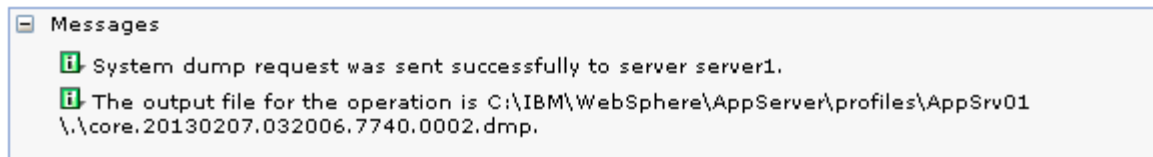
_____ Navigate to **Troubleshooting->Java dumps and cores**.



_____ Select **server1** and press **Java core**. Wait a few minutes for the response.



_____ Select **server1** and press System dump. Wait a few minutes for the response. A system dump takes longer than the Java dump in the previous step.



Note:

In previous versions of the IBM SDK shipped with WebSphere Application Server, it was necessary to post-process a system core file using a command called jextract before the system core could be used by tools such as Memory Analyzer. However, newer SDKs do not require this jextract step.

For reference, the IBM SDKs 1.6.0 SR9 or later, or any version of Java 1.7, no longer require jextract to be run. A technote in the references sections correlates the WebSphere Application Server versions to the SDK shipped.

This shows the SDKs which do not require jextract were shipped with WebSphere Application Server 7.0.0.15 and above.

____ Navigate to **Servers->Server Type->WebSphere application servers**. Select the running server, and press the Stop button.

Application servers

Use this page to view a list of the application servers in your environment and the status of each of these servers. You can also use this page to change the status of a specific application server.

+ Preferences

New... Delete Templates... Start **Stop** Restart ImmediateStop Terminate

Select	Name	Node	Host Name	Version	Cluster Name	Status
<input checked="" type="checkbox"/>	server1	impact2013Node	impact2013	ND 8.5.0.1	PlantsByWebSphereCluster	
<input type="checkbox"/>	server2	impact2013Node	impact2013	ND 8.5.0.1	PlantsByWebSphereCluster	

Total 2

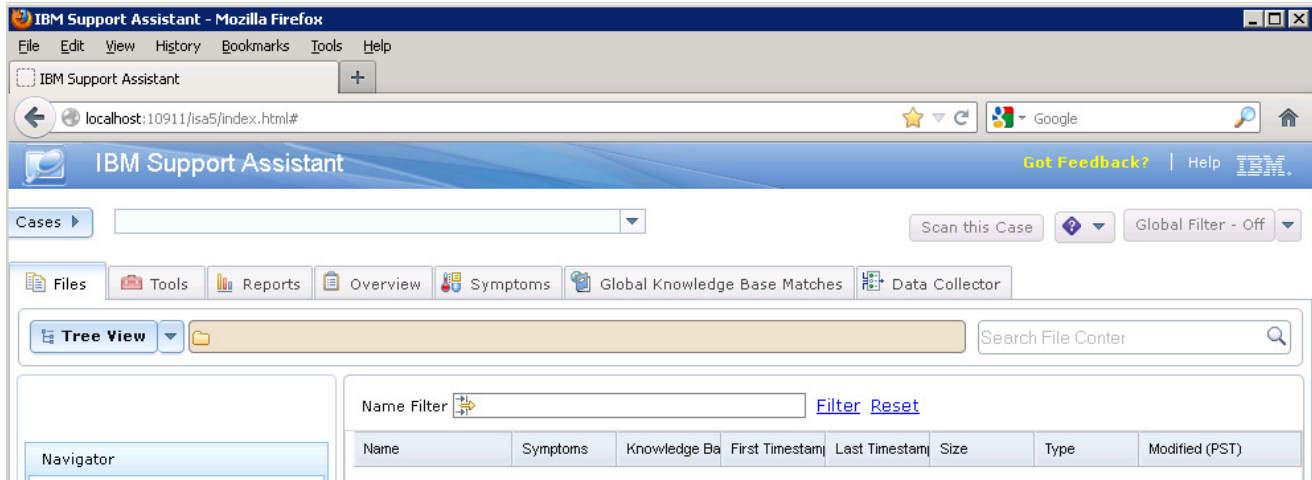
Server status feedback

Server status provides information about events that occur while the server stops.

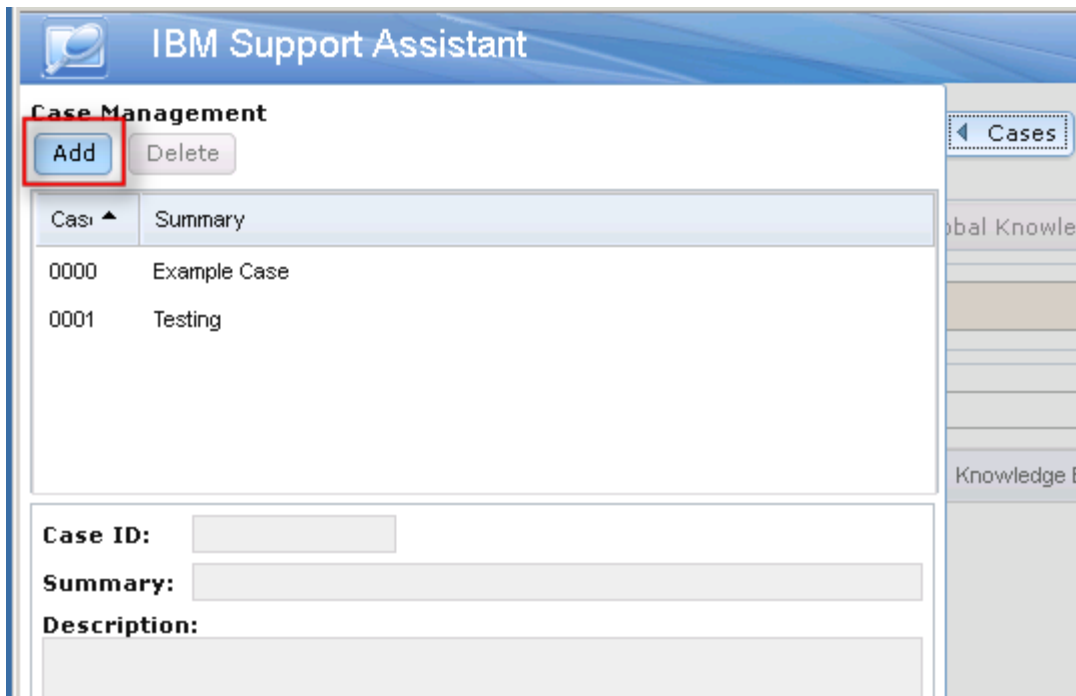
- impact2013Node:server1
 - i** Server impact2013Node/server1 stopped successfully. The collection may need to be refreshed to show the current server status. View JVM logs for further details.

OK

____ Launch the browser and use the bookmarks to load the ISA web interface.



____ Create a new case by clicking the **Cases** button, and then **Add**.

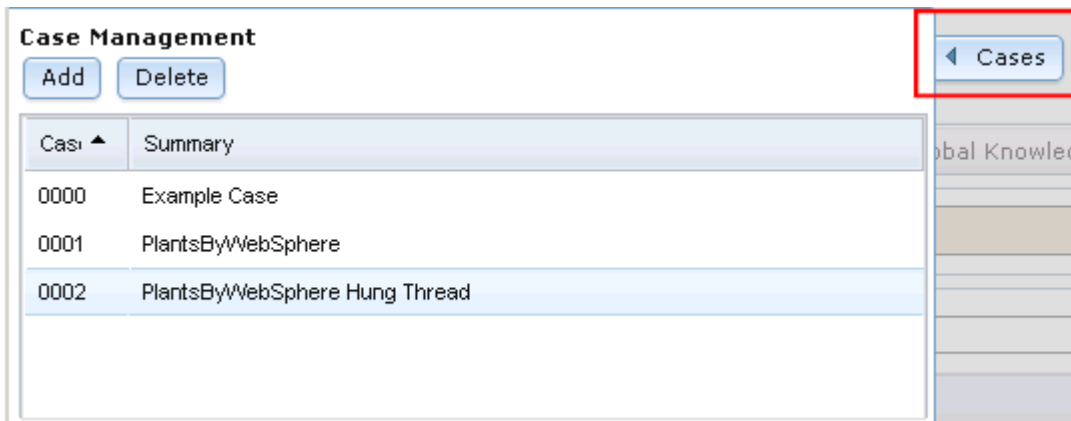


_____ Complete the summary and description, and click the **green tick**.



The screenshot shows a form for creating a new case. It has three main sections: 'Case ID' with a text box containing '[New]' and a green checkmark icon; 'Summary' with a text box containing 'PlantsByWebSphere Hung Thread'; and 'Description' with a larger text area containing 'Hung thread warning seen.'.

_____ Shrink the cases dialog by clicking **Cases**.

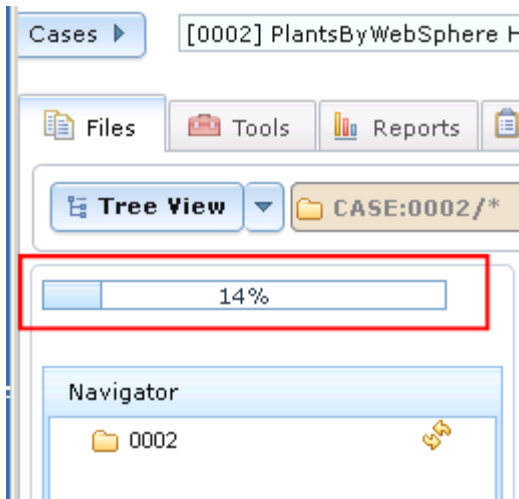
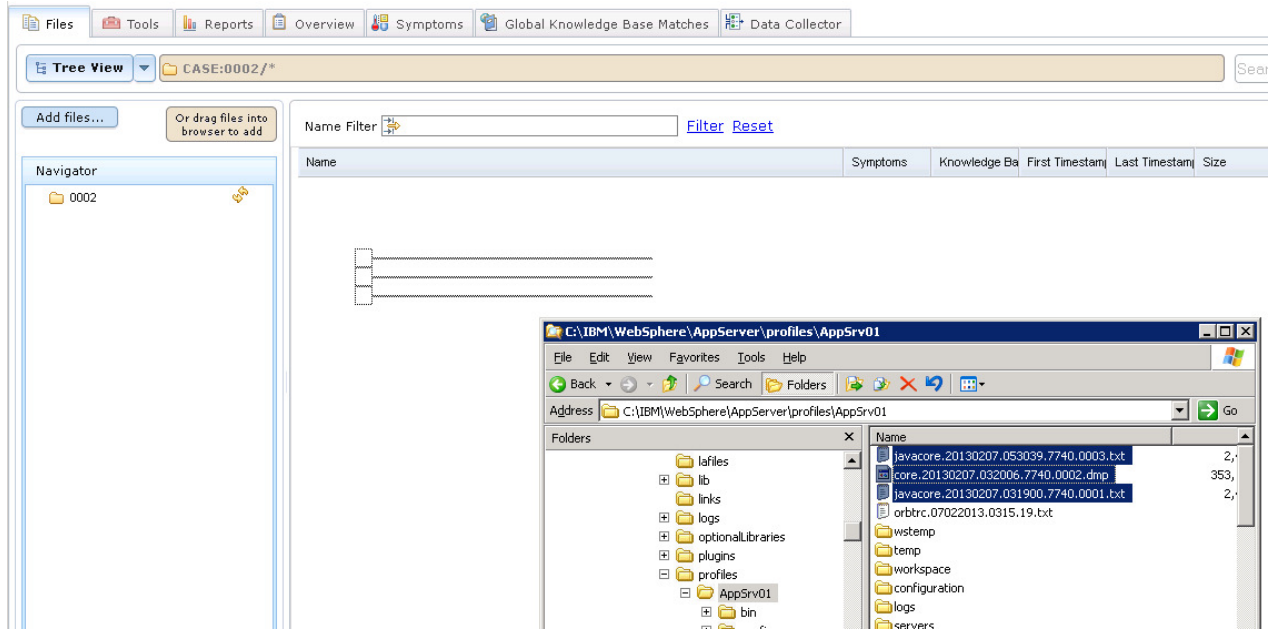


The screenshot shows the 'Case Management' dialog box. It has 'Add' and 'Delete' buttons at the top left. Below them is a table with two columns: 'Case' and 'Summary'. The table contains three rows: '0000 Example Case', '0001 PlantsByWebSphere', and '0002 PlantsByWebSphere Hung Thread'. On the right side of the dialog, there is a 'Cases' button with a left-pointing arrow, which is highlighted with a red box.

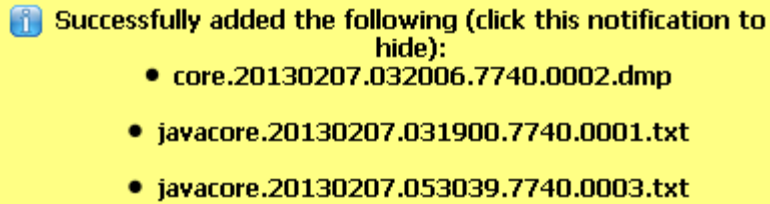
Case	Summary
0000	Example Case
0001	PlantsByWebSphere
0002	PlantsByWebSphere Hung Thread

_____ Add the two javacore.txt files, and one system dump file to the case by dragging the files from Windows Explorer into the ISA files list. The location of the files is:

C:\IBM\WebSphere\AppServer\profiles\AppSrv01



_____ Wait for the file add progress bar to read 100%, and for the yellow message box to appear.



Successfully added the following (click this notification to hide):

- core.20130207.032006.7740.0002.dmp
- javacore.20130207.031900.7740.0001.txt
- javacore.20130207.053039.7740.0003.txt

_____ Click the yellow box to dismiss the message

Note:

The files are now safely stored in the ISA folder where they can be accessed by a team of people.

The information in a Javadump is human readable, but the TDMA tool helps interpret the data. The primary benefits this tool provides are:

1. Summary of the Javadump, including any warnings, thread histogram by type and state, memory segment information, etc.
2. User interface for navigating thread stacks and monitors.
3. User interface for comparing multiple Javadumps and their respective thread stacks and monitors.

To make an initial assessment of the files, it can be convenient to run a TMDA report on the ISA server. This avoids the need to download the files & TMDA tool to a local workstation. More detailed analysis can be performed later using the desktop version of TMDA if required.

In the next steps you will generate a basic TMDA report to see high level details contained in the Javadump file.

_____ Ensure the relevant case is selected, then choose Thread and Monitor Dump Analyzer (TMDA) Report from the tools tab. Click **Launch**.

tool request has been submitted
[Go to output folder](#)

Cases ▶ [0002] PlantsByWebSphere Hung Thread

Files Tools Reports Overview Symptoms Global Knowledge Base Matches Data Collector

Enter keyword [Filter](#) [Reset](#)

Sort By: [a-z] Tag: All Tags

Garbage Collection and Memory Visualizer (GCMV) [Desktop]	✓ [i]
Garbage Collection and Memory Visualizer (GCMV) [Report]	✓ [b]
Health Center	✓ [i]
HeapAnalyzer [Desktop]	[g] [i]
Memory Analyzer [Desktop]	✓ [i]
Memory Analyzer [Report]	✓ [b]
Memory Analyzer Web Edition [Web]	[g] [i]
Thread and Monitor Dump Analyzer (TMDA) [Desktop]	[g] [i]
Thread and Monitor Dump Analyzer (TMDA) [Report]	[g] [b]
WebSphere Application Server Configuration Visualizer	[g] [b]

Thread and Monitor Dump Analyzer (TMDA) [Report]

Launch [Tool Help](#)

IBM Thread and Monitor Dump Analyzer for Java (TMDA)

TMDA compares each thread dump and monitor dump and automatically detects hair graph structure, and deadlocks.

This tool is provided in two versions:

- as a report generating version that reads javacore files and generates HTML
- as an interactive GUI version running on the desktop

... [\(more\)](#)

Tags: [b] Report Generator Tool [g] As-is | Problem Area: Java

Restrictions:

File Types Hint: Java thread dump or javacore files

Problem Analysis

Run Thread and Monitor Dump Analyzer (TMDA) [Report] (Version 4.3.4)

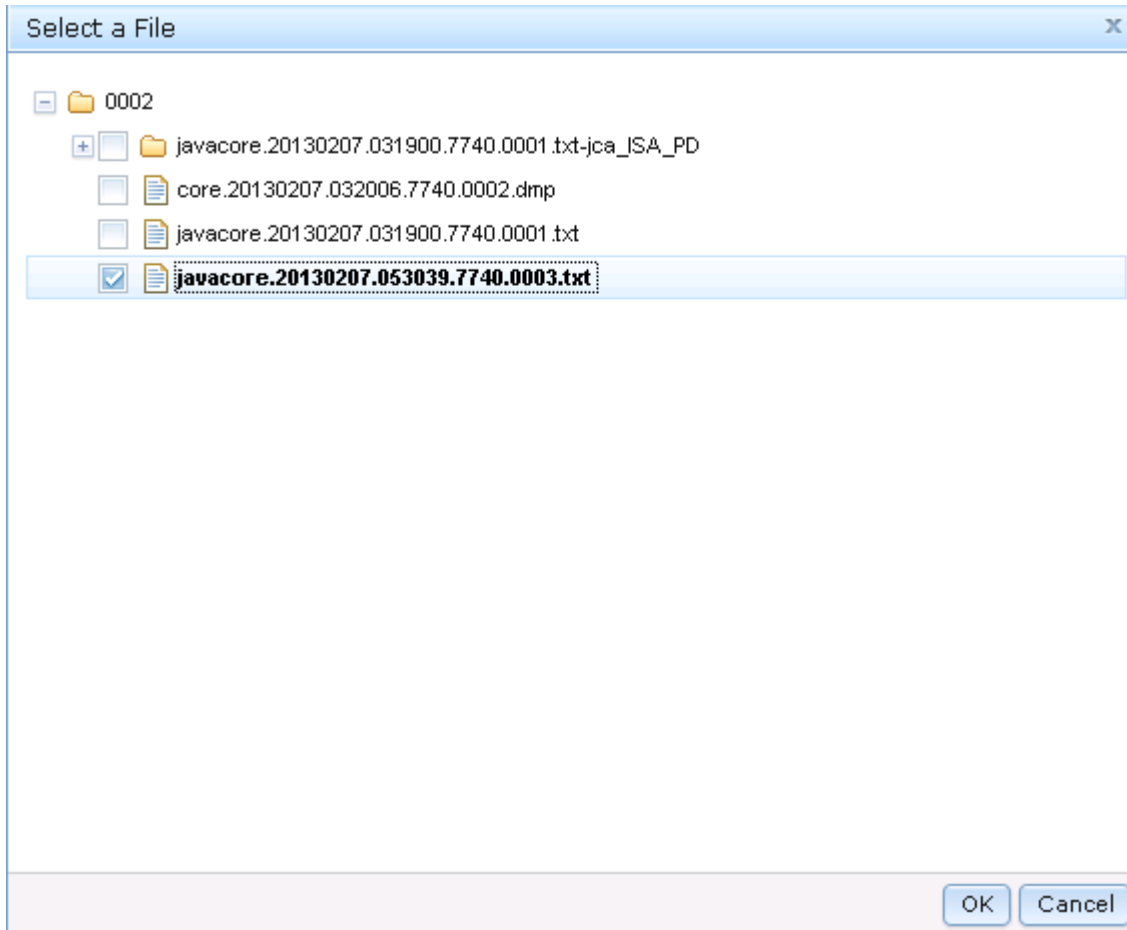
Input Files and Folders *

[Browse](#)

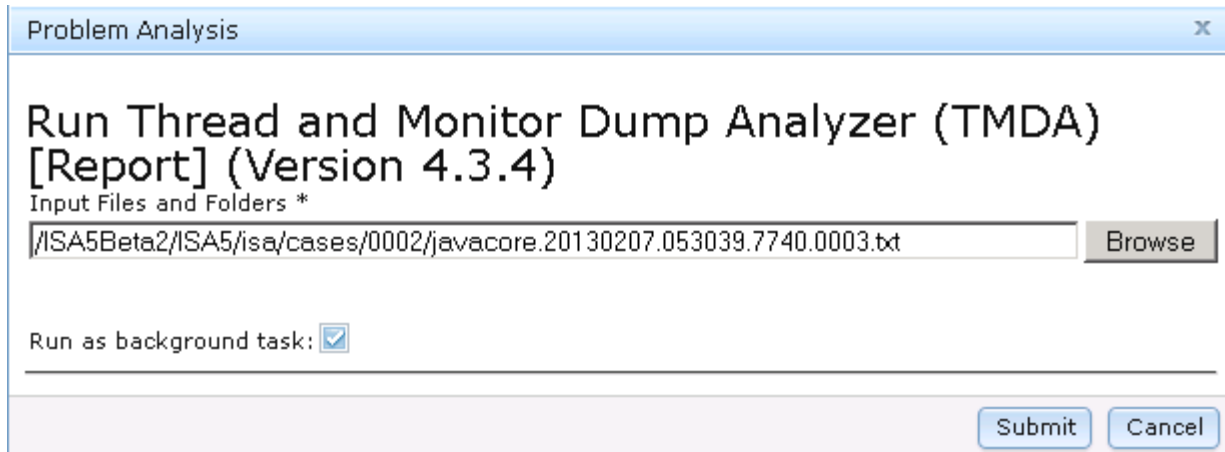
Run as background task:

[Submit](#) [Cancel](#)

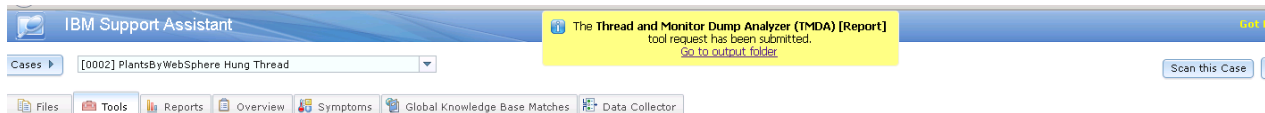
_____ Click **Browse** and select one of the **Javacore,txt** files.



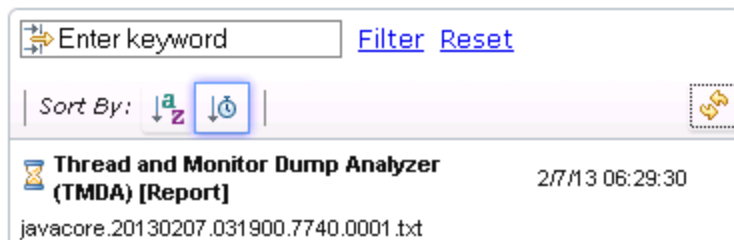
_____ Click **OK**



_____ Click **Submit**.



_____ Click the **yellow box** (not the link) to dismiss it, then click the **reports** tab.



_____ Click the **refresh** icon until the report's status shows a green tick. Then select the **report**.



_____ Take a brief look at the report on the right hand side. You should be able to find the following general information:

Cause of thread dump – This helps determine if the JVM experienced a crash or out of memory. In this case, the dump was triggered by a user signal, i.e. WebSphere's hung thread detection and/or the Health Center tool.

Java Version – The exact java version is displayed making finding IBM fixes easier.

System Classpath & User Arguments – Gives information about the Java environment and its configuration.

Free & Allocated Java Heap Size – Learn if the heap is fully expanded, and what percentage occupied it is.

Thread Status Analysis – Shows the status and number of running threads. In TDMA, the different states are color coded. The meaning of each state is explained later in this lab.

Thread Method Analysis – Shows what code is being executed by each thread at the time of the Javacore

Note:

To diagnose the hung thread problem presented in this lab, the more advanced and interactive features of the desktop TDMA tool are required.

_____ Choose Thread and Monitor Dump Analyzer (TMDA) Desktop from the tools tab. Click **Launch**.

Cases ▾
[0002] PlantsByWebSphere Hung Thread ▾
Go to output folder

Files
Tools
Reports
Overview
Symptoms
Global Knowledge Base Matches
Data Collector

Filter Reset

Sort By: a-z
Tag: All Tags ▾

- Garbage Collection and Memory Visualizer (GCMV) [Desktop]** ✓
- Garbage Collection and Memory Visualizer (GCMV) [Report]** ✓
- Health Center** ✓
- HeapAnalyzer [Desktop]**
- Memory Analyzer [Desktop]** ✓
- Memory Analyzer [Report]** ✓
- Memory Analyzer Web Edition [Web]**
- Thread and Monitor Dump Analyzer (TMDA) [Desktop]**
- Thread and Monitor Dump Analyzer (TMDA) [Report]**
- WebSphere Application Server Configuration Visualizer**

Thread and Monitor Dump Analyzer (TMDA) [Desktop]

Launch
 Tool Help

IBM Thread and Monitor Dump Analyzer for Java (TMDA)

TMDA compares each thread dump and monitor dump and automatically detects hair graph structure, and deadlocks.

This tool is provided in two versions:

- as a report generating version that reads javacore files and generates HTML*
- as an interactive GUI version running on the desktop*

... [\(more\)](#)

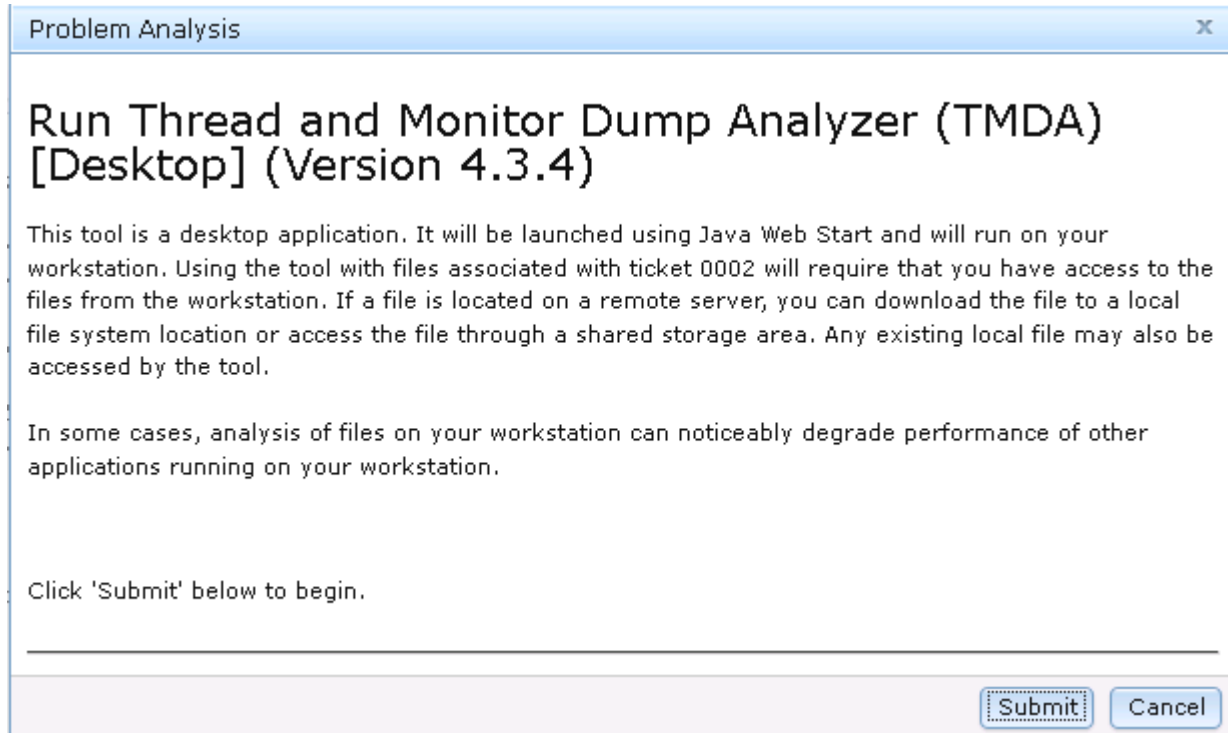
Tags: Desktop Tool | As-is | Problem Area: Java

Restrictions:

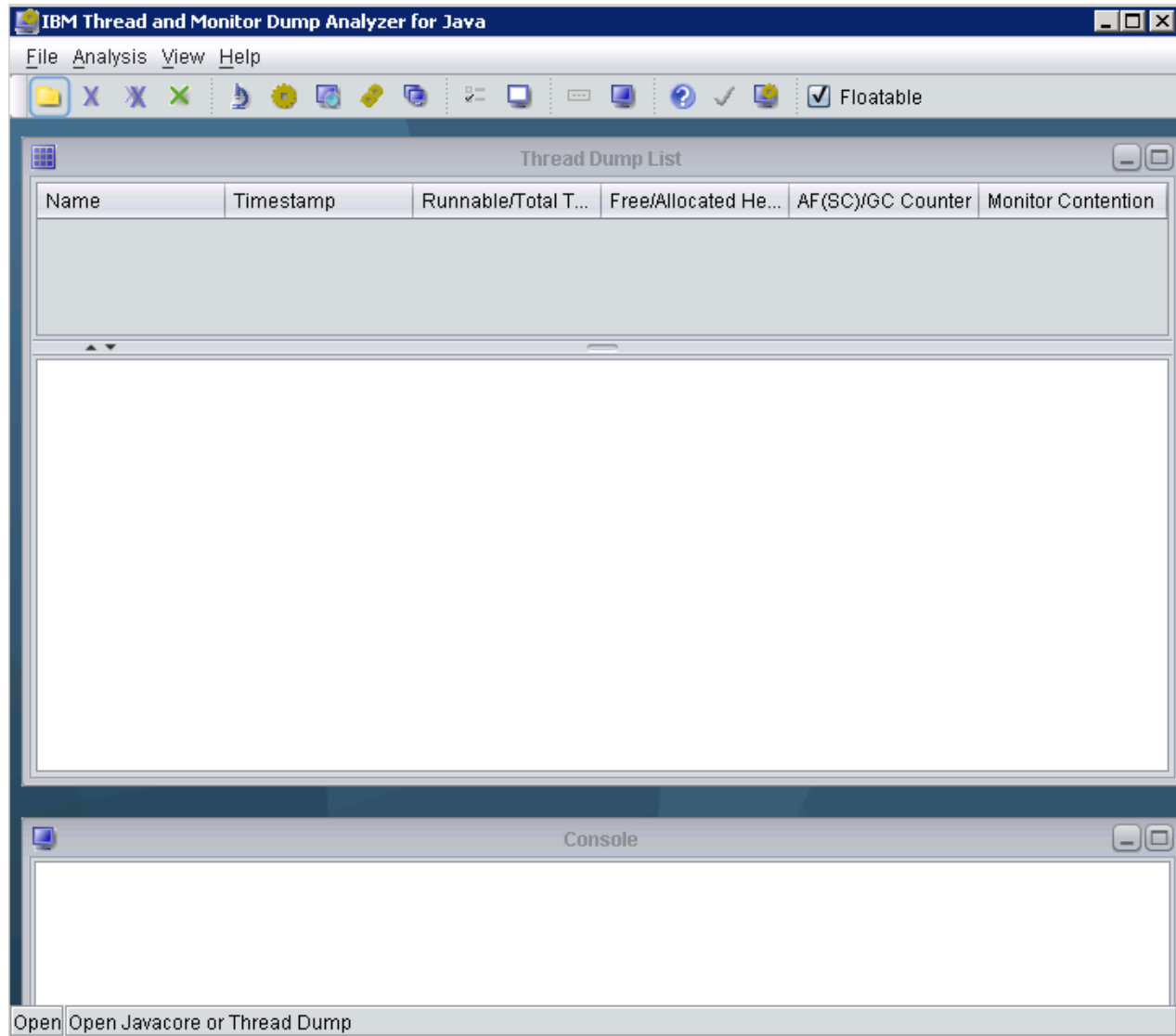
File Types Hint: Java thread dump or javacore files

Execution History/Status

Status	Start Time	Input Files



_____ Click **Submit**. The TMDA desktop tool will be downloaded and launched using Java Web Start – this will take a few moments. In the meantime, read the note below in preparation for using the TMDA desktop tool.

**Note:**

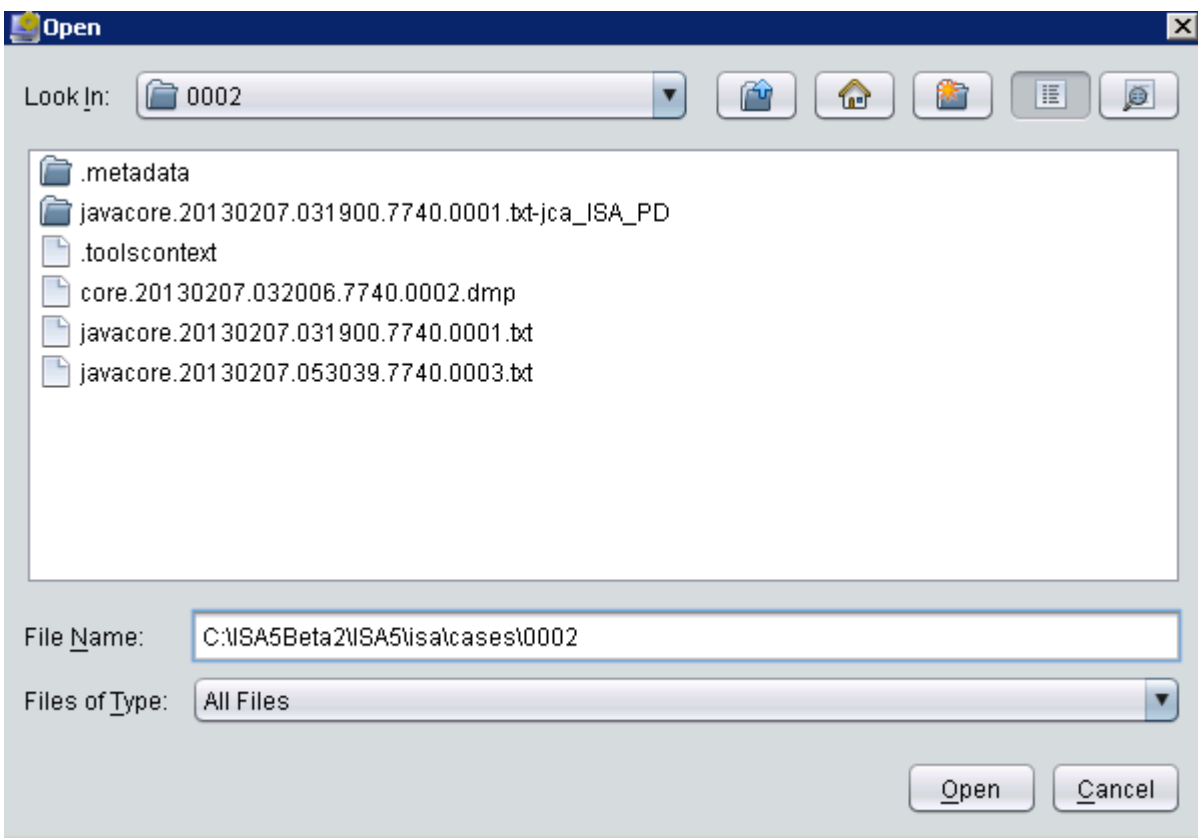
The SystemOut log file reported a “possible hung thread” warning, i.e. a thread that has been active for a period of time longer than considered normal by the hung thread detection heuristic. The log file also showed the name of the thread (something like “WebContainer : 5”) and a stack trace of the currently executing code.

You can learn the naming convention of common WebSphere and JVM threads by referring to a technote in the the references section. Threads starting with “WebContainer” are WebSphere threads dealing with incoming HTTP requests that come from the client.

The Java stacktrace showed that a ShoppingBean object in the PlantsByWebSphere application had executed a Thread.sleep, meaning the thread did not end in a timely manner. However, it is likely that the WebSphere Application Server will have many “WebContainer” threads that are handling incoming HTTP requests, and many would be executing the ShoppingBean code. The remaining steps of this lab will investigate why that particular thread encountered a (deliberate) problem.

_____ In the TMDA tool. Select **File->Open** thread dumps

_____ Navigate to **C:\ISA5Beta2\ISA5\isa\cases\0002**

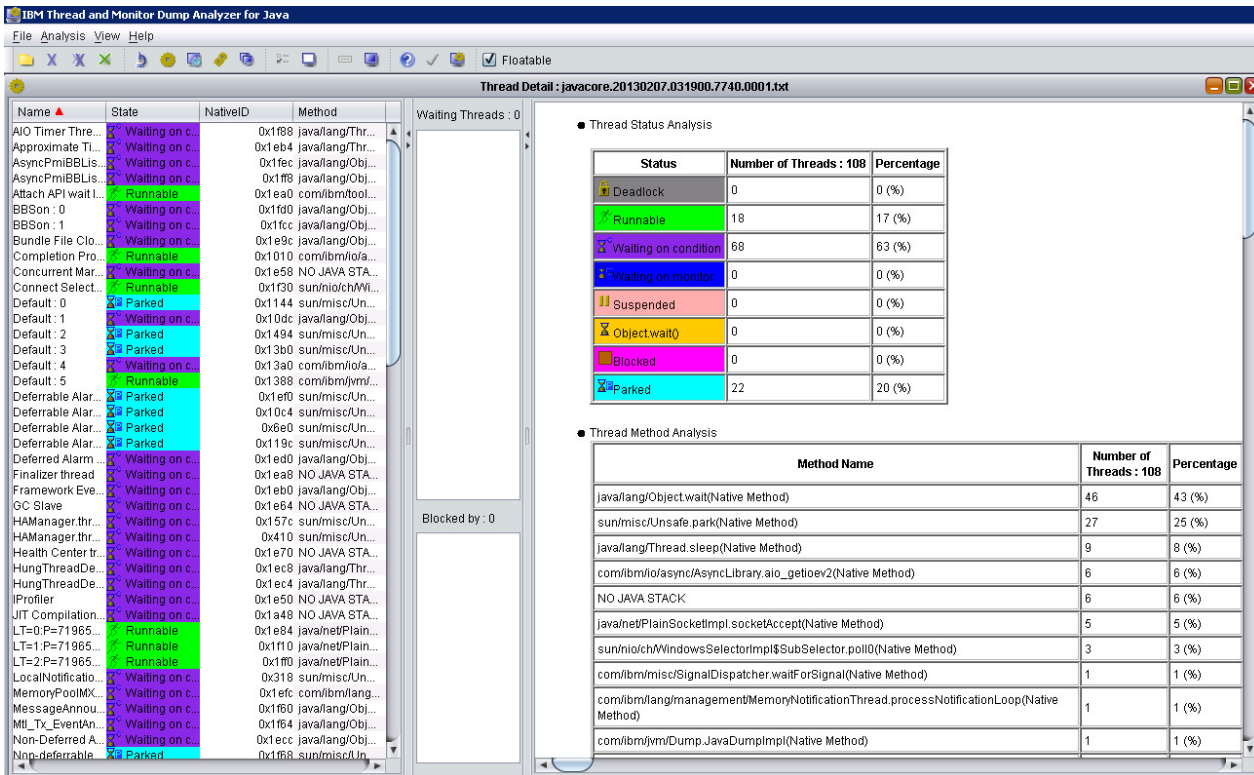
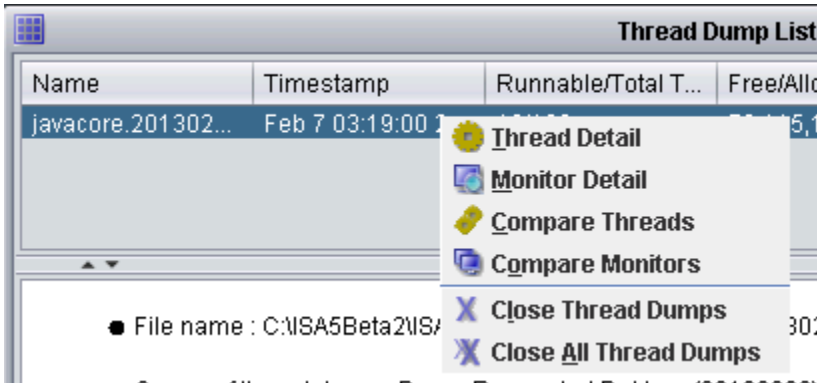


_____ Select the first **Javacore.txt** file, and click **Open** (look at the last few characters of the filename to determine which was created first).

IBM Software Accelerated Value Program

_____ Select the **Javacore.txt** file in the Thread Dump List. Below you will see the general overview information that you saw in the previous TMDA report.

_____ Right click the **file** and choose **Thread Detail** from the popup menu.



_____ Maximize the tool's windows and expand the columns in the TDMA "Thread Details" window to see more information.

Click the **Name** column heading to sort the threads by name. As we already know there is a problem with a WebSphere web application, it seems reasonable to only investigate WebSphere threads, specifically the “WebContainer” threads.

The screenshot shows the 'Thread Detail' window for a specific JVM instance. The table below represents the data shown in the screenshot:

Name ▲	State	NativeID	Method
ThreadManager.JobsPro...	Waiting on condition		0x24e4 java/lang/Object.wait(N...
TrLogger	Waiting on condition		0x2490 java/lang/Object.wait(N...
UpstreamPing	Waiting on condition		0x25fc java/lang/Thread.sleep...
VEUtilWorkDispatcher : 0	Parked		0x27dc sun/misc/Unsafe.park(...
WLMMSonitorSleeper : 0	Waiting on condition		0x2474 java/lang/Object.wait(N...
WMQJCAResourceAdapt...	Waiting on condition		0x1a44 java/lang/Object.wait(N...
WMQJCAResourceAdapt...	Waiting on condition		0x21e4 java/lang/Object.wait(N...
WMQJCAResourceAdapt...	Waiting on condition		0x256c java/lang/Object.wait(N...
WMQJCAResourceAdapt...	Waiting on condition		0x1cbc java/lang/Object.wait(N...
WMQJCAResourceAdapt...	Waiting on condition		0x25ac java/lang/Object.wait(N...
WT=0	Waiting on condition		0x1a30 java/lang/Object.wait(N...
WebContainer : 0	Parked		0x26b8 sun/misc/Unsafe.park(...
WebContainer : 1	Waiting on condition		0x2768 sun/misc/Unsafe.park(...
WebContainer : 10	Waiting on condition		0x27a4 com/ibm/io/async/Asyn...
WebContainer : 11	Parked		0x275c sun/misc/Unsafe.park(...
WebContainer : 2	Parked		0x2758 sun/misc/Unsafe.park(...
WebContainer : 3	Waiting on condition		0x2750 java/lang/Object.wait(N...
WebContainer : 4	Parked		0x25a0 sun/misc/Unsafe.park(...
WebContainer : 5	Waiting on condition		0x2784 java/lang/Thread.sleep...
WebContainer : 6	Runnable		0x260c com/ibm/io/async/Asyn...
WebContainer : 7	Waiting on condition		0x25dc com/ibm/websphere/s...
WebContainer : 8	Waiting on condition		0x2690 com/ibm/websphere/s...
WebContainer : 9	Waiting on condition		0x25e8 com/ibm/io/async/Asyn...
Worker-0	Waiting on condition		0x15ec java/lang/Object.wait(N...
XDTimer	Waiting on condition		0x25e4 java/lang/Object.wait(N...
com.ibm.son.mesh.Peer...	Runnable		0x1a5c sun/nio/ch/WindowsSe...
java.net.MulticastSocket...	Runnable		0x25f0 java/net/PlainDatagra...
pool-2-thread-1	Parked		0x1504 sun/misc/Unsafe.park(...
server.startup : 0	Parked		0x2330 sun/misc/Unsafe.park(...
sonInThreadPool : 0	Runnable		0x2388 com/ibm/io/async/Asyn...

The exact states of the “WebContainer” threads in your Javacore may vary. In the screenshot above, the majority of thread states are purple in color and one is green. Purple threads are threads in state “Condition Wait” and green threads are “Runnable”. An explanation of these states follows in the note below:

Note: Definition of Thread States

State	Name	TDMA Default Colour	Description	Tips and Tricks
R	Runnable	Green	The thread is able to run when given the chance	Many threads that you may expect to be “runnable” are actually shown as “condition wait” in Javacores from JDK 5 and above. This is actually by design as the JVM sets many threads to “Condition Wait” as it writes the Javacore. <i>See “Unexpected Conditional Wait thread states in IBM Javacore” in the references for more details.</i>
CW	Condition Wait	Purple	The thread is waiting. For example, because: A sleep() call is made The thread has been blocked for I/O A wait() method is called to wait on a monitor being notified The thread is synchronizing with another thread with a join() call	Common methods at the top of the stack trace for threads in this state include: Object.wait: Waiting for a notify from another thread Thread.sleep: Operating system call to sleep for X ms AsyncLibrary.aio_getioev2: Async I/O waiting on new work SocketInputStream.socketRead0: Socket established, processing or waiting for data
S	Suspended	Salmon	The thread has been suspended by another thread	
Z	Zombie		The thread has been killed	
P	Parked	Cyan	The thread has been parked by the Java concurrency API (java.util.concurrent).	

_____ Select the **web container thread** that caused the hung thread warning in the WebSphere log file (refer to your SystemOut.log to confirm the exact title which may vary from the one shown in this lab document).

WebContainer : 5

Waiting on condition

0x2784 java/lang/Thread.sleep...

_____ Verify the state of the thread and confirm it is not deadlocked. Deadlocking would be a possible cause of a hung thread and the JVM has automatic detection of this which would be reported in both the Jvadmop, and by TDMA. If the thread was in deadlock it would be colored grey in TDMA, and be entitled “deadlocked”. In this lab, the hung thread is actually in state “Condition Wait” (purple). In addition, note that the first line of the Java stack shows “java/lang/Thread.sleep” which matches the stack trace shown in the SystemOut log file.

_____ Note the information in the panels on the right hand side. This shows “Waiting Threads”, i.e. if the currently highlighted thread held a Java monitor and other threads were waiting for it, they would be displayed in this section. In this lab, no threads are blocked by the hung thread under investigation. In addition, the full Java and native stack traces are shown in the rightmost panel.

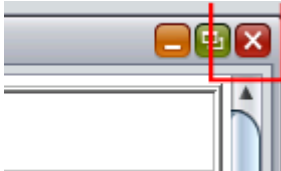
Thread Name	WebContainer : 5
State	Waiting on condition
Stack	<pre> at java/lang/Thread.sleep(Native Method) at java/lang/Thread.sleep(Thread.java:896) at com/ibm/websphere/samples/pbw/war/ShoppingBean.deliberateHungThread(ShoppingBean.java:415) at com/ibm/websphere/samples/pbw/war/ShoppingBean.performProductDetail(ShoppingBean.java:182) at sun/reflect/NativeMethodAccessorImpl.invoke0(Native Method) at sun/reflect/NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:60(Compiled Code)) at sun/reflect/DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:37(Compiled Code)) at java/lang/reflect/Method.invoke(Method.java:611(Compiled Code)) at org/apache/webbeans/intercept/InterceptorHandler.invoke(InterceptorHandler.java:287) at org/apache/webbeans/intercept/NormalScopedBeanInterceptorHandler.invoke(NormalScopedBeanInterco at com/ibm/websphere/samples/pbw/war/ShoppingBean_\$\$javassist_1.performProductDetail(ShoppingBea at sun/reflect/NativeMethodAccessorImpl.invoke0(Native Method) at sun/reflect/NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:60(Compiled Code)) at sun/reflect/DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:37(Compiled Code)) at java/lang/reflect/Method.invoke(Method.java:611(Compiled Code)) at org/apache/el/parser/ASTValue.invoke(ASTValue.java:266) at org/apache/el/MethodExpressionImpl.invoke(MethodExpressionImpl.java:278) at org/apache/myfaces/view/facelets/el/TagMethodExpression.invoke(TagMethodExpression.java:83) at javax/faces/component/_MethodExpressionToMethodBinding.invoke(_MethodExpressionToMethodBinding at org/apache/myfaces/application/ActionListenerImpl.processAction(ActionListenerImpl.java:100) at javax/faces/component/UICommand.broadcast(UICommand.java:120) at javax/faces/component/UIViewRoot._broadcastAll(UIViewRoot.java:973) at javax/faces/component/UIViewRoot.broadcastEvents(UIViewRoot.java:275) at javax/faces/component/UIViewRoot._process(UIViewRoot.java:1285) at javax/faces/component/UIViewRoot.processApplication(UIViewRoot.java:711) at org/apache/myfaces/facelets/el/TagMethodExpression.invoke(TagMethodExpression.java:83) </pre>

Note:

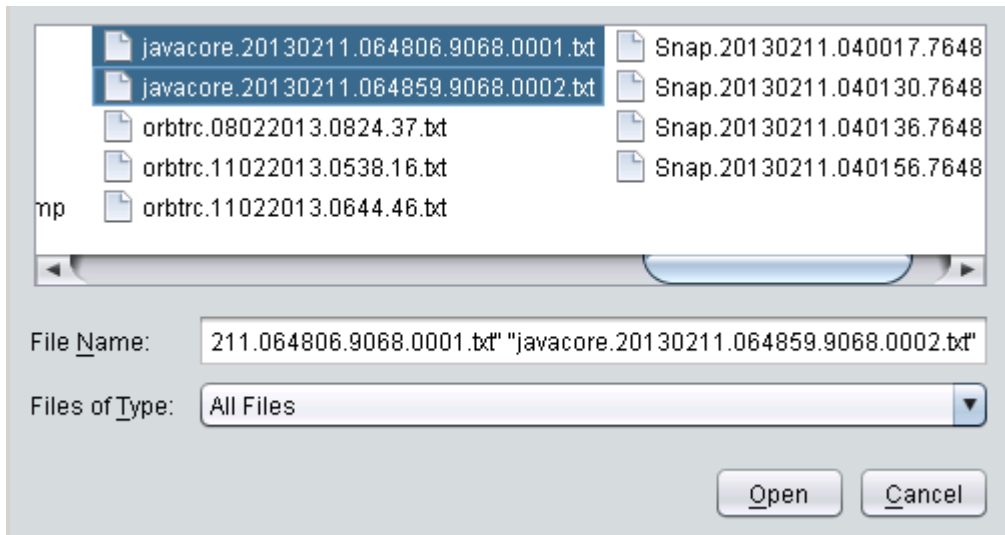
It is clear that the when the Jvadmop was generated, the suspected hung thread was performing a Thread.sleep. However, this might be part of its design and perhaps after a short Thread.sleep it would

continue and finish its work, ending normally. A good way to see if a thread is actually progressing (albeit slowly) is to use multiple Javacores over a period of time. If the stack trace for this thread never shows a Java stack associated with incoming HTTP requests and only ever shows Thread.sleep, it will be increasing likely that the thread will not complete in a timely manner, meaning an application defect has been found.

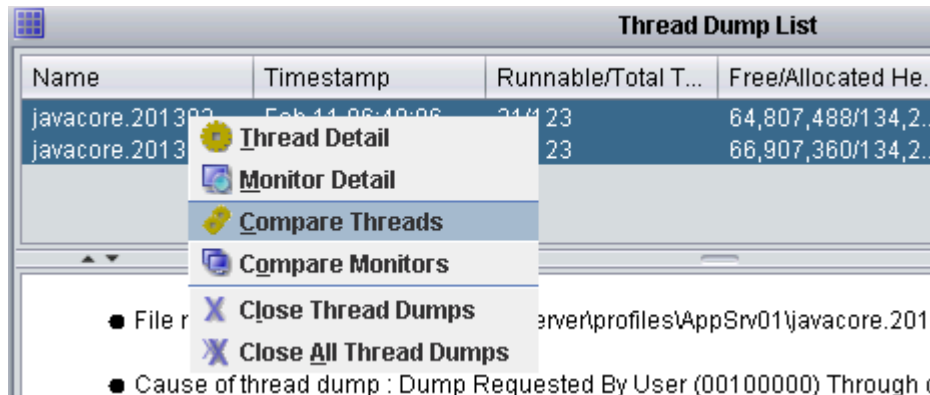
_____ Click **File->Close all thread dumps**, and close the "Thread Detail" window.



_____ "File->Open thread dumps" and select both **Javacore.txt** files and click **Open**. TDMA will compare the Javacore file written by WebSphere's hung thread detection, and the other triggered manually a few minutes later via the WebSphere administration console.



_____ Highlight both Javacores and right click. Select **Compare Threads** from the popup menu.



As before, maximize the windows and expand out the columns. Order the first column by name to locate the “WebContainer” threads.

The two columns show the state of the threads a few minutes apart. You will probably find that some or all of the “WebContainer” threads show a different stack in each column indicating the threads were executing different code at each point in time. However, the thread that has been identified as a hang suspect will definitely be highlighted in red meaning that TDMA also considers it a hang suspect. This is because it is showing exactly the same Java stack in both Javadumps. Therefore the thread is still paused in a Thread.sleep, confirming the suspicion that it really is hung and not just proceeding slowly.

WebContainer : 6	com/ibm/lo/async/Async...	sun/misc/Unsafe.park(...)
WebContainer : 5	java/lang/Thread.sleep(...)	java/lang/Thread.sleep(...)
WebContainer : 4	sun/misc/Unsafe.park(...)	java/lang/Object.wait(N...

Note:

To find more detail about this hung thread, it can often be useful to relate the thread to a system dump. In addition this provides a correlation between the thread name seen in the Javadump, and the thread Ids that would be traced by WebSphere's logging and tracing mechanism. The technique to achieve this mapping will be demonstrated later in this lab.

A system dump contains information about the Java threads whereas an IBM heap dump (.phd) does not. The Memory Analyzer can present the same thread information when inspecting an IBM heap dump if both the heap dump and a Javacore from the same point in time are loaded into the tool simultaneously.

However a system dump is the only format that includes the full details about Java field names and values. Therefore this type of dump is the most useful for debugging Java issues, especially hung threads or other performance issues as demonstrated in this lab.

_____ Close the TDMA windows, pressing the Yes button to exit.

_____ Return to the ISA 5 tab in the browser.

_____ From the Tools tab, select **Memory Analyzer (Desktop)** and click **Launch**.

The screenshot shows the IBM Tools catalog interface. On the left, a list of tools is displayed with search filters. The 'Memory Analyzer [Desktop]' tool is highlighted with a red box. On the right, the details panel for 'Memory Analyzer [Desktop]' is shown, featuring a 'Launch' button and descriptive text about the tool's capabilities and versions.

Tool Name	Status	Icon
Garbage Collection and Memory Visualizer (GCMV) [Desktop]	✓	J
Garbage Collection and Memory Visualizer (GCMV) [Report]	✓	Bar Chart
Health Center	✓	J
HeapAnalyzer [Desktop]	60	J
Memory Analyzer [Desktop]	✓	J
Memory Analyzer [Report]	✓	Bar Chart
Memory Analyzer Web Edition [Web]	60	J
Thread and Monitor Dump Analyzer (TMDA) [Desktop]	60	J

Memory Analyzer [Desktop]

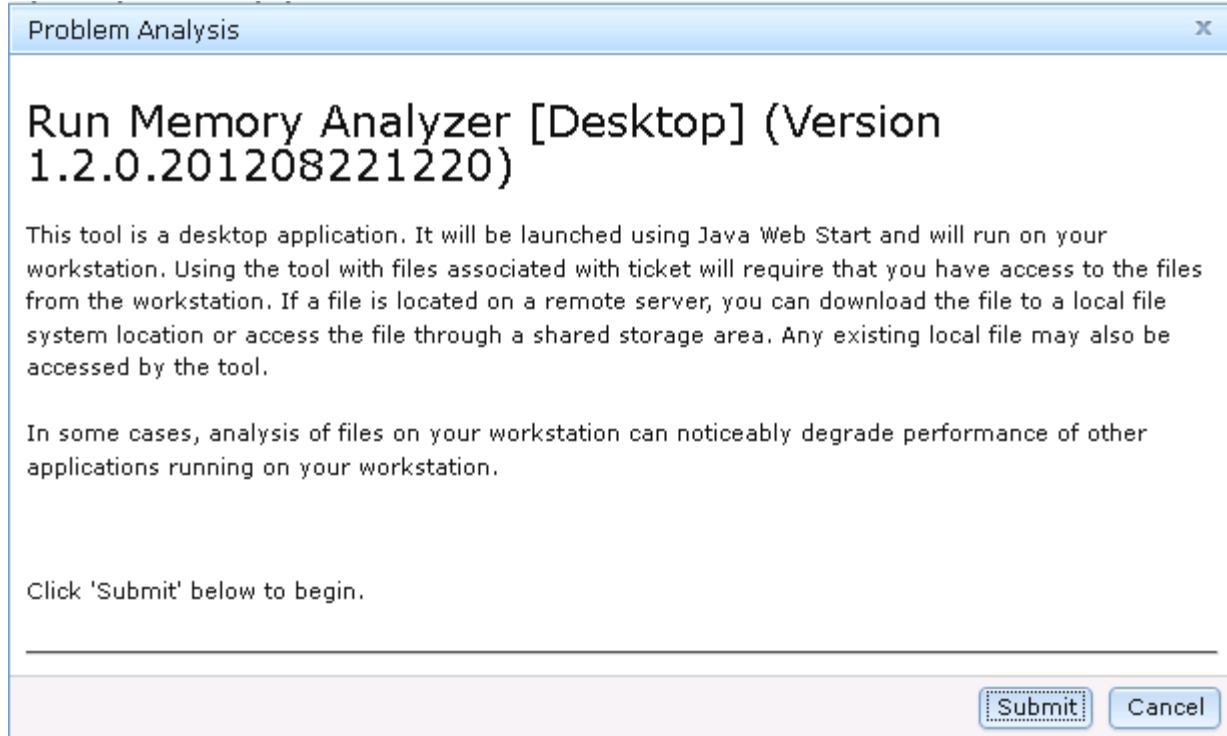
▶ Launch ℹ Tool Help

IBM Monitoring and Diagnostic Tools for Java

Memory Analyzer is a feature-rich Java heap leaks and reduce memory consumption.

This tool is provided in three versions:

- as a report generating version that re... and generates some zipped HTML rep...
- as an interactive GUI version running...
- an interac...



_____ Click **Submit** and wait for the Memory Analyzer tool to download and launch. Note that the window does not take focus, so click the program on the taskbar.



_____ Click **File->Open Heap Dump**.

_____ Navigate to path below and open the system dump you previously triggered via the WebSphere admin console. The system dump file will end ".dmp".

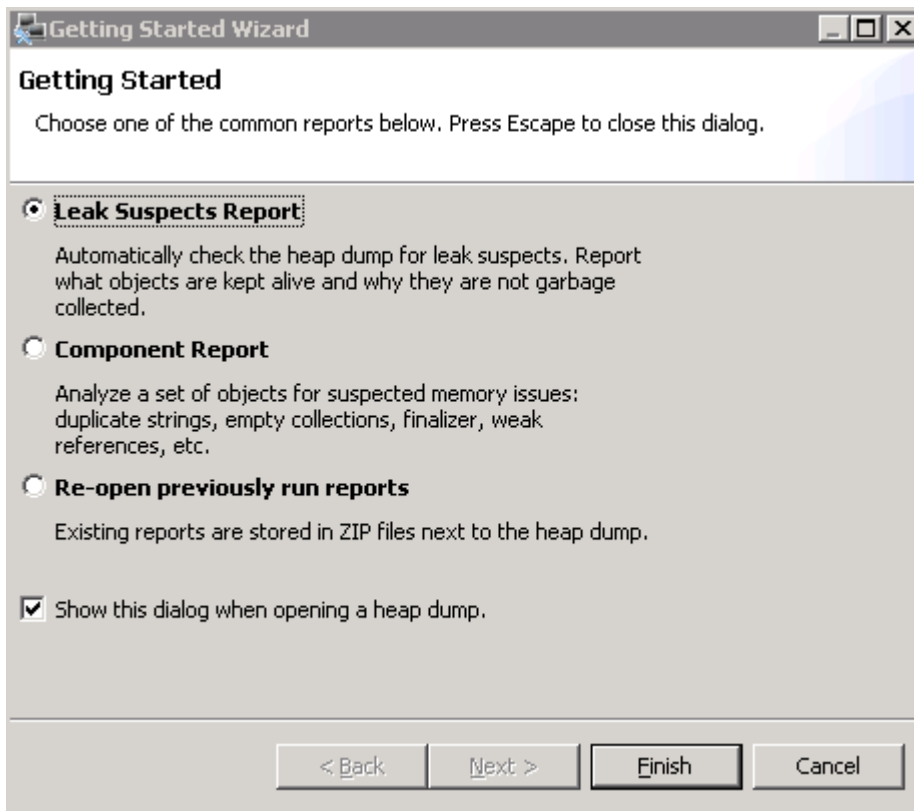
C:\ISA5Beta2\ISA5\isa\cases\0002


Note:

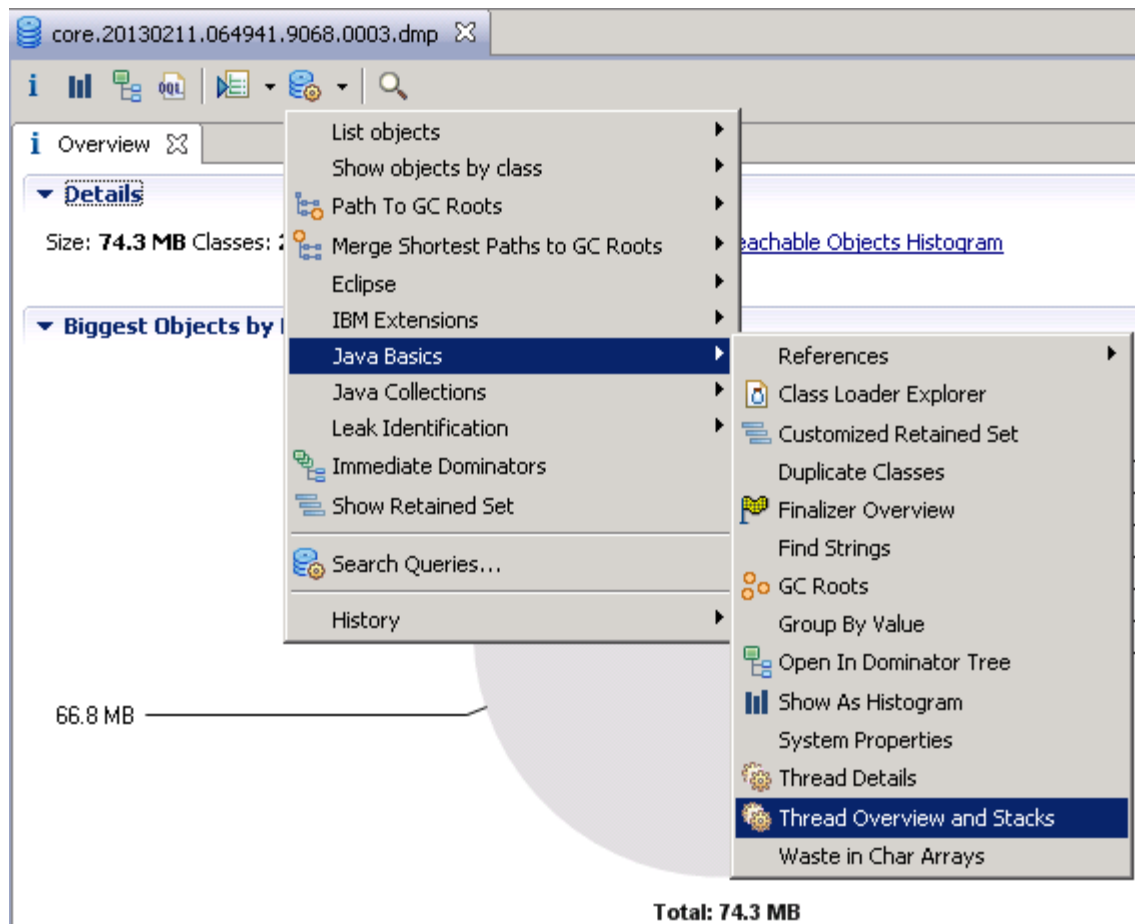
Please be patient while Memory Analyzer parses the dump, approximately 3-5 minutes may pass with little nothing reported by the progress bar.

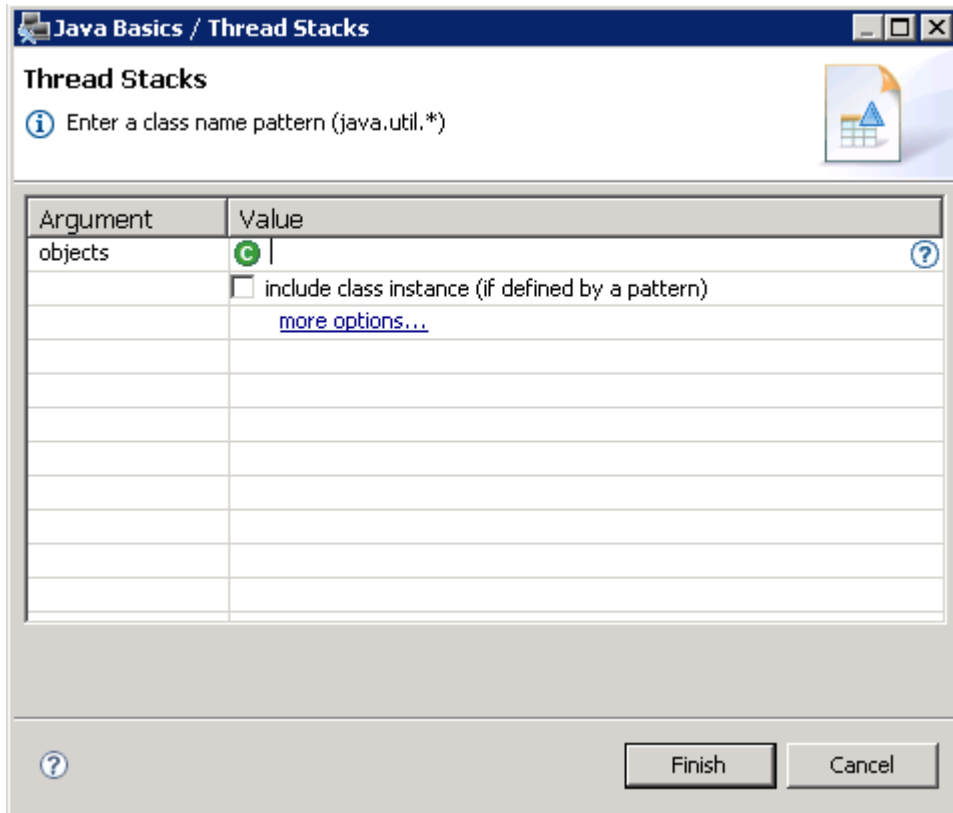
_____ A warning may be displayed about a missing snapshot. Dismiss this warning by pressing OK

_____ Click **Cancel** on the **Getting Started Wizard**.



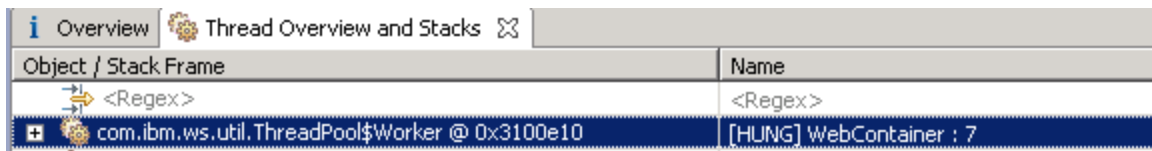
Click the **Open Query Browser** icon  and select **Java Basics->Thread Overview and Stacks**



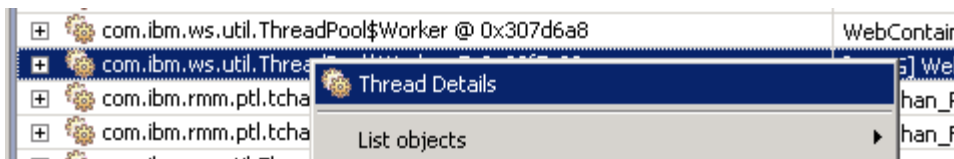


_____ Click **Finish** to show all Threads

_____ In the table, locate the ThreadPoolWorker thread with name “WebContainer” that was reported as hung. Remember, the exact Id may differ from the screenshots in this lab.



_____ Highlight the thread and right click it. Try launching **Java Basics->Thread Details** to inspect all details about the thread.



Thread Details

Thread Details

Thread [HUNG] WebContainer : 7

Thread Properties

Object / Stack Frame	com.ibm.ws.util.ThreadPool\$Worker @ 0x3100e10
Name	[HUNG] WebContainer : 7
Shallow Heap	144
Retained Heap	193,784
Context Class Loader	com.ibm.ws.classloader.CompoundClassLoader @ 0x267e0b8
Is Daemon	true
DTFJ Name	WebContainer : 7
JNIEnv	0x8d45800
Priority	5
State	[alive, sleeping, waiting]
State value	0xc1
Native id	7576
Σ Total: 12 entries	

Thread Stack

```
[HUNG] WebContainer : 7
  at java.lang.Thread.sleep(J)V (Native Method)
  at java.lang.Thread.sleep(J)V (Thread.java:896)
  at com.ibm.websphere.samples.pbw.war.ShoppingBean.deliberateHungThread()V (ShoppingBean.java:415)
  at com.ibm.websphere.samples.pbw.war.ShoppingBean.performProductDetail()Ljava/lang/String; (Shoppi
```

Note:

Key point to note here is that the thread stack that is hung as shown above points to the method “ShoppingBean.performProductDetail” which in turn calls the method “ShoppingBean.deliberateHungThread” as being responsible for calling Thread.sleep.

Next, return to the “thread_stacks” window or tab and ensure the hang suspect thread is still highlighted as shown below. Look at the information in the “Inspector” panel on the left hand side and note that the thread has an attribute “isHung” which is set to true. The IBM Extensions for Memory Analyzer have also annotated the Thread Stacks report with the text **[Hung]** to further emphasize this. This confirms that the thread was still considered hung by WebSphere Application Server at the time the system dump was triggered.

Object / Stack Frame	Name
<Regex>	<Regex>
com.ibm.ws.util.ThreadPool\$Worker @ 0x3100e10	[HUNG] WebContainer : 7
com.ibm.rmm.ptl.tchan.receiver.PacketProcessor @ 0x1d36590	Ptl_Tchan_PacketProcessor
com.ibm.rmm.ptl.tchan.transmitter.PacketFireout @ 0x1d36138	Ptl_Tchan_Fireout
com.ibm.ws.util.ThreadPool\$Worker @ 0x26b9880	server.startup : 1
com.ibm.rmm.mtl.receiver.MessageAnnouncer @ 0x1d36448	MessageAnnouncer for com.ibm.rmm.mtl.receiver.MStreamSetR@b2d4b17d
com.ibm.ws.util.ThreadPool\$Worker @ 0x2fd58b8	WebContainer : 3
com.ibm.ws.util.ThreadPool\$Worker @ 0x2fd5a68	WebContainer : 0

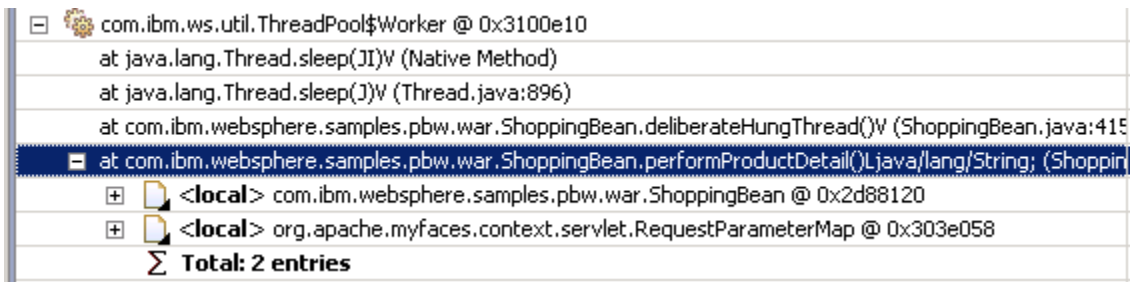
Inspector

@ 0x3100e10

- ThreadPool\$Worker
- com.ibm.ws.util
- class com.ibm.ws.util.ThreadPool\$Worker @ 0xe966e8
- java.lang.Thread
- org.eclipse.osgi.internal.baseadaptor.DefaultClassLoader @...
- 144 (shallow size)
- 193,784 (retained size)
- GC root: Thread

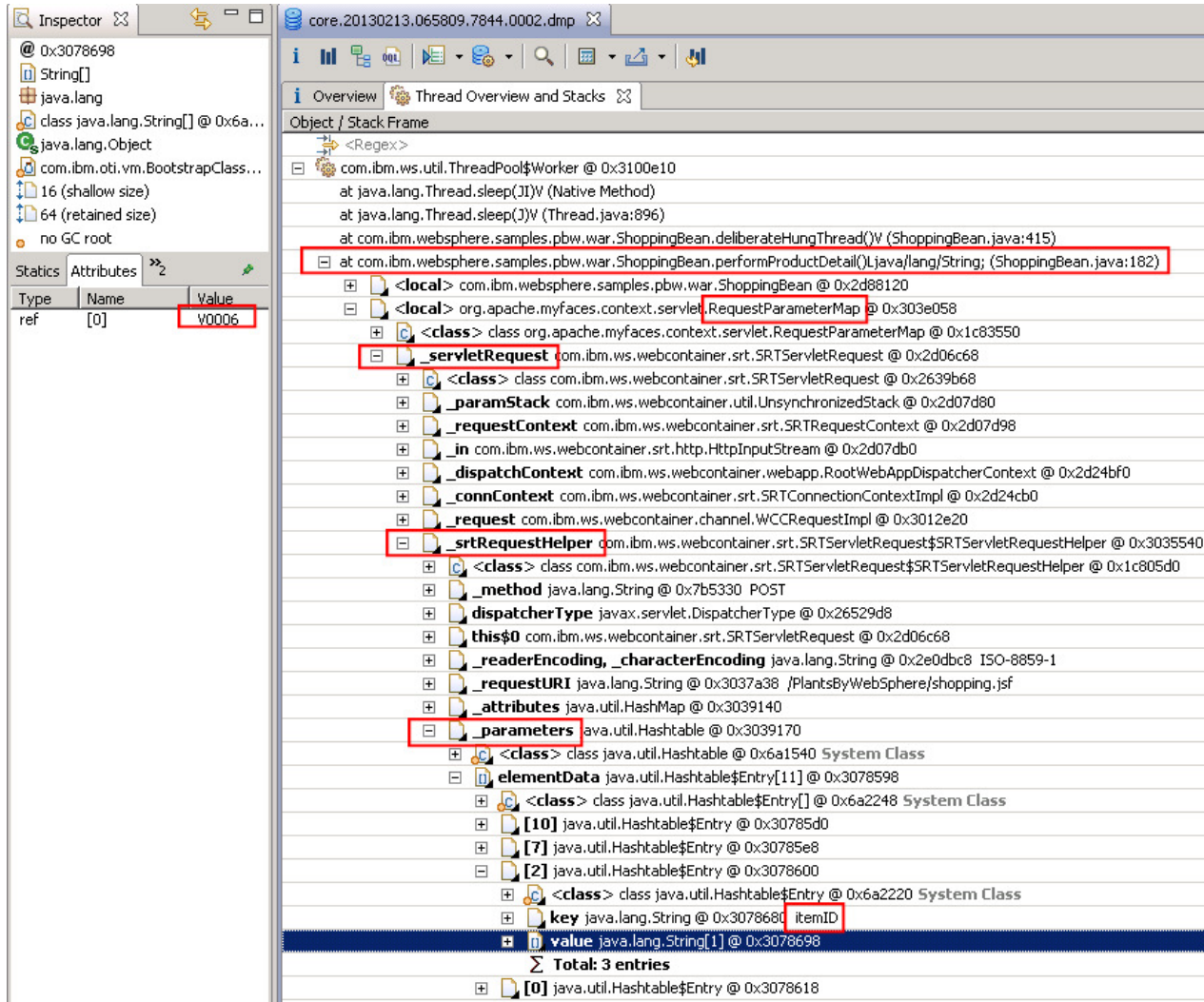
Type	Name	Value
ref	firstTask_	null
ref	wsThreadL...	java.lang.Object[44] @ 0x3035da8
long	startTime	1360767250781
long	createdTime	1360767231406
bool...	isHung	true
ref	threadNum...	000000a2
ref	activeWasT...	com.ibm.ws.util.ThreadPool\$WasThread
int	numberOfA	0

In the “thread_stacks” window, expand the thread by clicking on the “+” next to it and look for the method “ShoppingBean.performProductDetail” as seen in the stack earlier since it was responsible for calling Thread.sleep. Inspect the local variables and their values – in this case an instance of ShoppingBean and a RequestParameterMap as shown below.



This technique normally gives a lot of information about what the thread was processing. For example if it has opened a socket, the values will show the destination IP. Similarly, if the thread is performing a database query, the internal structures will reveal the SQL. This is the reason a system dump is recommended for debugging performance related issues such as hung threads.

_____ In this scenario, looking for the HTTP request values could be of interest. Take a look at the screenshot and expand your stack frame to find the request parameters, specifically the “key” itemID. Select “value” below to see the value in the Inspector window on the left hand side.

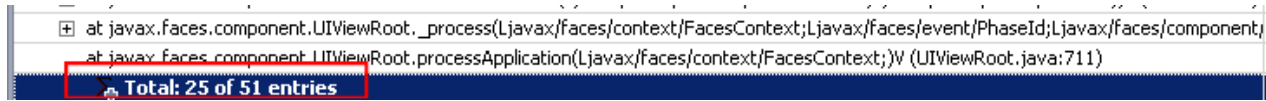


The sequence of references may look quite complicated, but to someone familiar with the JSF framework they would probably be quite familiar. You can see that by looking at JSF references relating to the request, eventually you find a “_parameters” Hashtable that has an “itemID” with value “V0006”.

This may relate to a product Id in the ShoppingBean, and could determine the exact HTTP request that caused this particular thread to become hung while other “WebContainer” threads did not.

_____ Collapse the “ShoppingBean.performProductDetail” stack frame you were inspecting.

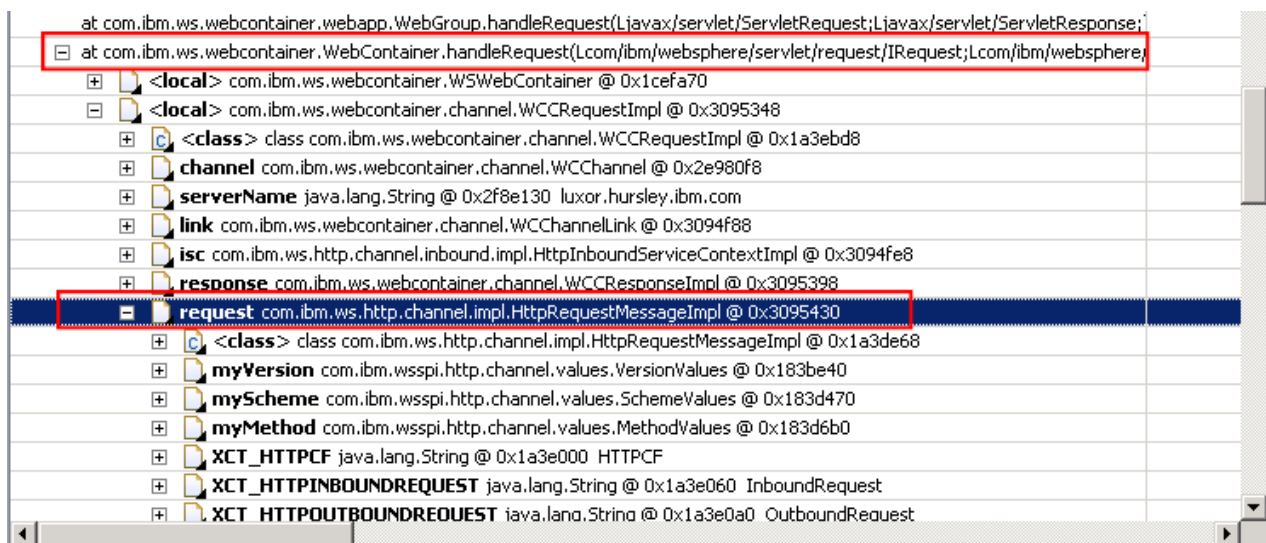
_____ In the stack frame of the same thread, scroll down and double click the **25 out of 51 entries** (your values may vary)



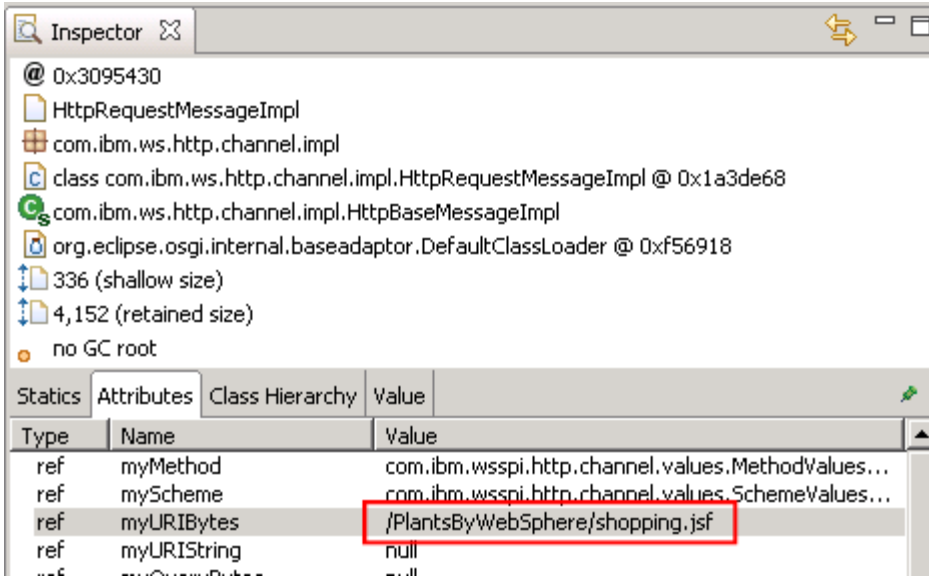
_____ Scroll down to a method earlier in the servlet lifecycle. Highlight the method **“com.ibm.ws.WebContainer.handleRequest”**. Note that there are similar methods above and below, so be sure to locate the correct method. In this method we should be able to find the HTTP request object, which can help to determine the URI that was being processed by the Web Container thread.



_____ Expand the references until the variable “request” of type “com.ibm.ws.channel.imp.HttpRequestMessageImpl” is located. Of course it takes some experience to locate the right data structures.



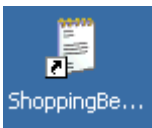
_____ Refer to the Inspector window on the left hand side. The request parameter includes the URI “/PlantsByWebSphere/shopping.jsf”



_____ Optionally, inspect some of the other “WebContainer” threads. They may have completely different thread stacks to the hung thread. If you find one that has a similar stack, i.e. with ShoppingBean.performProduct detail() close to the top, repeat the previous steps to locate the HTTP request information again. You will find that no other threads are dealing with requests where the item id is “V0006”. It is becoming increasingly clear that the HTTP request that caused the hung thread was for a PlantsByWebSphere item with an id of “V0006”.

Optional Steps:

_____ Double click the desktop shortcut to **ShoppingBean.java** to inspect the programming error.



_____ Click **Edit->Find** and search for “V0006”.

The deliberate error is indeed related to a product with Id "V0006". When a request for this product is received, the method "deliberateHungThread" is called which puts the thread to sleep for an hour.

```
    } else if ("V0006".equals(invID)) {  
  
        // -----  
        // User clicked on the Strawberries, let's pause  
        // a webcontainer thread for a berry long time  
        // -----  
  
        // Comment out the method call below to remove this deliberate mistake  
        this.deliberateHungThread();  
    }  
  
private void deliberateHungThread() {  
  
    // -----  
    // User clicked on the Strawberries, let's pause  
    // a webcontainer thread for a berry long time  
    // -----  
  
    try {  
  
        System.out.println("==> STARTING LONG SLEEP from Thread Id=" +  
            Thread.currentThread().getId() + " Thread name=" +  
            Thread.currentThread().getName());  
  
        // Sleep for an hour  
        Thread.sleep(3600000);  
  
        System.out.println("==> ENDING LONG SLEEP from Thread Id=" +  
            Thread.currentThread().getId() +  
            " Thread name=" + Thread.currentThread().getName());  
    } catch (InterruptedException e) {  
        e.printStackTrace();  
    }  
}
```

Part 4: (Optional) Mapping a Thread Id in a Javdump to the output of WebSphere logging or trace (10 minutes)

Note:

If you need to correlate a thread Id from a Javdump to the output of WebSphere logging or trace file, there are different options according to the version of WebSphere Application Server being used. In a default configuration, WAS logging and tracing both print out a "thread identifier" with each log entry. This identifier has no relationship to the internal or native thread identifiers for that Java thread. Therefore, this identifier by itself is only useful to correlate messages on the same thread within logs and trace, but not to Javadumps. This behavior is by design, but thread identifiers may be aligned in future versions.

Correlation between Javadumps and WebSphere logging or trace can be achieved using one of these methods:

Method #1

Thread identifiers may be correlated at any one point in time using a system dump, like the one used in the previous section of this lab. The IBM Extensions for Memory Analyzer can be used to display an additional column, the "WAS Thread ID". You will try out this method in this part of the lab.

Method #2

You can also configure a server's trace output with the "Advanced" file formatting option to see more detailed trace information for use in troubleshooting and problem determination. Below is an example of the advanced format which prints the thread name with each trace entry:

```
[9/21/11 12:22:44:507 PDT] 00000022 I UOW=null source=com.ibm.ws.webcontainer.servlet
class=com.ibm.ws.webcontainer.servlet.ServletWrapper method=init org=null prod=null component=null
thread=[WebContainer : 0]
```

Method #3

Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log, SystemErr.log, trace.log, and activity.log files. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory.

With HPEL enabled on a server, the logViewer command can be used to print both log and trace entries in the advanced format. In method #2, we had to explicitly change to advanced format. With HPEL, that information is always recorded, but we use different formatting options with the logViewer command to display the advanced format. For example:

```
<WAS_PROFILE_ROOT>/bin/logViewer -format advanced
```

Prints:

```
[9/26/11 14:37:06:493 PDT] 00000026 I UOW= source=com.ibm.ws.webcontainer.servlet
class=com.ibm.ws.webcontainer.servlet.ServletWrapper method=init org= prod= component=
thread=[WebContainer : 0]
```

```
SRVE0242I: [WasSwat] [/swat] [/index.jsp]: Initialization successful.
```

Note: Methods 2 and 3 assume that thread names are unique and unchanging, which is not always the case. Method 3 is only available for WebSphere Application Server 8.0 and above.

For more information on these methods, see the references.

_____ In the Memory Analyzer window, return to the “Thread Overview and Stacks” window. (If you have closed this, open the report again with Query Browser (button) -> Java Basics -> Thread Overview and Stacks, then click **Finish**)

_____ Locate the ThreadPoolWorker thread with the thread name that was previously reported as hung in the WebSphere logs, for example “WebContainer : 7”.

Object / Stack Frame	Name
<Regex>	<Regex>
com.ibm.ws.util.ThreadPool\$Worker @ 0x3100e10	[HUNG] WebContainer : 7
com.ibm.rmm.ptl.tchan.receiver.PacketProcessor @ 0x1d36590	Ptl_Tchan_PacketProcessor
com.ibm.rmm.ptl.tchan.transmitter.PacketFireout @ 0x1d36138	Ptl_Tchan_Fireout
com.ibm.ws.util.ThreadPool\$Worker @ 0x26b9880	server.startup : 1
com.ibm.rmm.mtl.receiver.MessageAnnouncer @ 0x1d36448	MessageAnnouncer for com.ibm.rmm.mtl.receiver.MStreamSetR@b2d4b17
com.ibm.ws.util.ThreadPool\$Worker @ 0x2fd58b8	WebContainer : 3
com.ibm.ws.util.ThreadPool\$Worker @ 0x2fd5a68	WebContainer : 0
com.ibm.ws.util.ThreadPool\$Worker @ 0x2fd59d8	WebContainer : 1
com.ibm.ws.util.ThreadPool\$Worker @ 0x2fd5708	WebContainer : 6
com.ibm.ws.util.ThreadPool\$Worker @ 0x2fd5ca8	ProcessDiscovery : 0
com.ibm.ws.util.ThreadPool\$Worker @ 0x2fd5798	WebContainer : 5
com.ibm.ws.util.ThreadPool\$Worker @ 0x2e86190	WebContainer : 13
com.ibm.ws.util.ThreadPool\$Worker @ 0x308ed60	WebContainer : 10
com.ibm.ws.util.ThreadPool\$Worker @ 0x2e89ef8	WebContainer : 14
com.ibm.ws.util.ThreadPool\$Worker @ 0x3100d80	WebContainer : 8
com.ibm.ws.util.ThreadPool\$Worker @ 0x3100cf0	WebContainer : 9
com.ibm.ws.util.ThreadPool\$Worker @ 0x2e86220	WebContainer : 12
com.ibm.ws.util.ThreadPool\$Worker @ 0x2e862b0	WebContainer : 11
com.ibm.ws.util.ThreadPool\$Worker @ 0x2e727e0	WebContainer : 15
com.ibm.ws.util.ThreadPool\$Worker @ 0x2fd5828	WebContainer : 4
com.ibm.ws.util.ThreadPool\$Worker @ 0x2fd5948	WebContainer : 2
java.lang.Thread @ 0x81e9d8	P=83234;O=1;CT
com.ibm.rmm.intrn.util.TaskManager @ 0x1d360a0	Rmm_TaskManager

_____ Scroll to the right to reveal the “WAS Thread ID” column. This additional column has been added to this default report by the IBM Extensions for Memory Analyzer. (Note the column order has been rearranged in the screenshot below)

Name	WAS Thread ID	Native id	Shallow Heap
WebContainer : 11	0x0000009b	7716	144
WebContainer : 2	0x00000091	7872	144
WebContainer : 3	0x00000092	7868	144
WebContainer : 4	0x00000093	6852	144
WebContainer : 5	0x00000094	7880	144
WebContainer : 6	0x00000095	4656	144
WebContainer : 8	0x00000098	7900	144
WebContainer : 9	0x00000099	6808	144
Worker-0	0x0000001d	4524	104
XDTimer	0x0000007c	4860	104
[HUNG] WebContainer : 7	0x00000097	7892	144
com.ibm.son.mesh.Peer-tcp-port-11004	0x00000052	4836	248
java.net.MulticastSocket@f2a9de7c	0x0000007e	5156	96
pool-2-thread-1	0x00000077	6724	96
server startup : 1	0x0000006c	5216	144

The system dump shows the hung thread uses WAS Thread ID **0x00000097**.

This can be correlated to any log statements in the SystemOut.log, e.g.:

```
[2/13/13 6:54:11:343 PST] 00000097 SystemOut  O ==> STARTING LONG SLEEP
```

The same id would be present in a WebSphere trace file.

Finally, to link the WebSphere logs to a thread in the Jvadium, again you need the system dump to tie this together. Either use the thread name, "WebContainer : 7" or the Native Id. Note that the thread name is not necessarily constant and could be changed by the code. To use the Native Id, convert the Native Id column from Memory Analyzer to Hex, and locate this in the Jvadium. In this case, the value **7892** is **1ED4** in hex, and can be located in the Jvadium:

```
3XMTHREADINFO    "WebContainer : 7" J9VMThread:0x15358400, j9thread_t:0x08EE1D40,  
java/lang/Thread:0x03131050, state:CW, prio=5  
  
3XMTHREADINFO1    (native thread ID:0x1ED4, native priority:0x5, native policy:UNKNOWN)  
  
3XMTHREADINFO3    Java callstack:  
  
4XESTACKTRACE     at java/lang/Thread.sleep(Native Method)  
  
4XESTACKTRACE     at java/lang/Thread.sleep(Thread.java:896)  
  
4XESTACKTRACE     at  
com/ibm/websphere/samples/pbw/war/ShoppingBean.deliberateHungThread(ShoppingBean.java:415)
```

_____ Optionally try to link together output from your hung thread in SystemOut.log with the Jvadium, using the system dump as described above.

Reference Links

- IBM Support Assistant Information and Downloads:
<http://www-01.ibm.com/software/support/isa/>
- Verify Java SDK version shipped with IBM WebSphere Application Server fix packs
<http://www-01.ibm.com/support/docview.wss?uid=swg27005002>
- Problem determination for javacore files from WebSphere Application Server
<http://www-01.ibm.com/support/docview.wss?uid=swg21181068>
- Mapping Underlying Java Thread Identifiers to those in Logging and Trace
<http://www-01.ibm.com/support/docview.wss?uid=swg21418557>