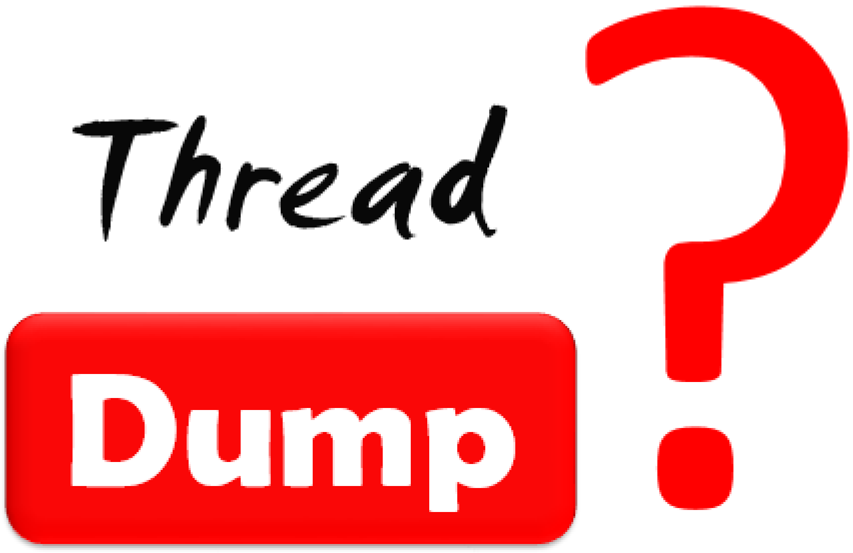
fast

Universal Java Thread Dump Analyzer

Fast thread

Universal Java Thread Dump Analyzer

HOW TO TAKE THREAD DUMPS? – 8 OPTIONS



Thread dumps are vital artifacts to diagnose CPU spikes, deadlocks, poor response times, memory problems, unresponsive applications, and other system problems. There are great online thread dump analysis tools such as http://fastthread.io/, which can analyse and spot problems. But to those tools you need provide proper thread dumps as input. Thus in this article, I have documented 8 different options to capture thread dumps.

1. jstack

'jstack' is an effective command line tool to capture thread dumps. jstack tool is shipped in JDK_HOMEbin folder. Here is the command that you need to issue to capture thread dump:

```
jstack -l  <pid> > <file-path>
```

where

pid: is the Process Id of the application, whose thread dump should be captured

file-path: is the file path where thread dump will be written in to.

Example:

```
jstack -l 37320 > /opt/tmp/threadDump.txt
```

As per the example thread dump of the process would be generated in /opt/tmp/threadDump.txt file.

Jstack tool is included in JDK since Java 5. If you are running in older version of java, consider using other options

2. Kill -3

In major enterprises for security reasons only JREs are installed on production machines. Since jstack and other tools are only part of JDK, you wouldn't be able to use jstack tool. In such circumstances 'kill -3' option can be used.

```
kill -3 <pid>
```

where

pid: is the Process Id of the application, whose thread dump should be captured

Example:

```
Kill -3 37320
```

When 'kill -3' option is used thread dump is sent to the standard error stream. If you are running your application in tomcat, thread dump will be sent into <TOMCAT_HOME>/logs/catalina.out file.

Note: To my knowledge this option is supported in most flavours of *nix operating systems (Unix, Linux, HP-UX operating systems). Not sure about other Operating systems.

3. JVisualVM

Java VisualVM is a graphical user interface tool that provides detailed information about the applications while they are running on a specified Java Virtual Machine (JVM). It's located in JDK_HOMEbinjvisualvm.exe. It's part of Sun's JDK distribution since JDK 6 update 7.s

Launch the jvisualvm. On the left panel, you will notice all the java applications that are running on your machine. You need to select your application from the list (see the red color highlight in the below diagram). This tool also has the capability to capture thread dumps from the java processes that are running on the remote host as well.

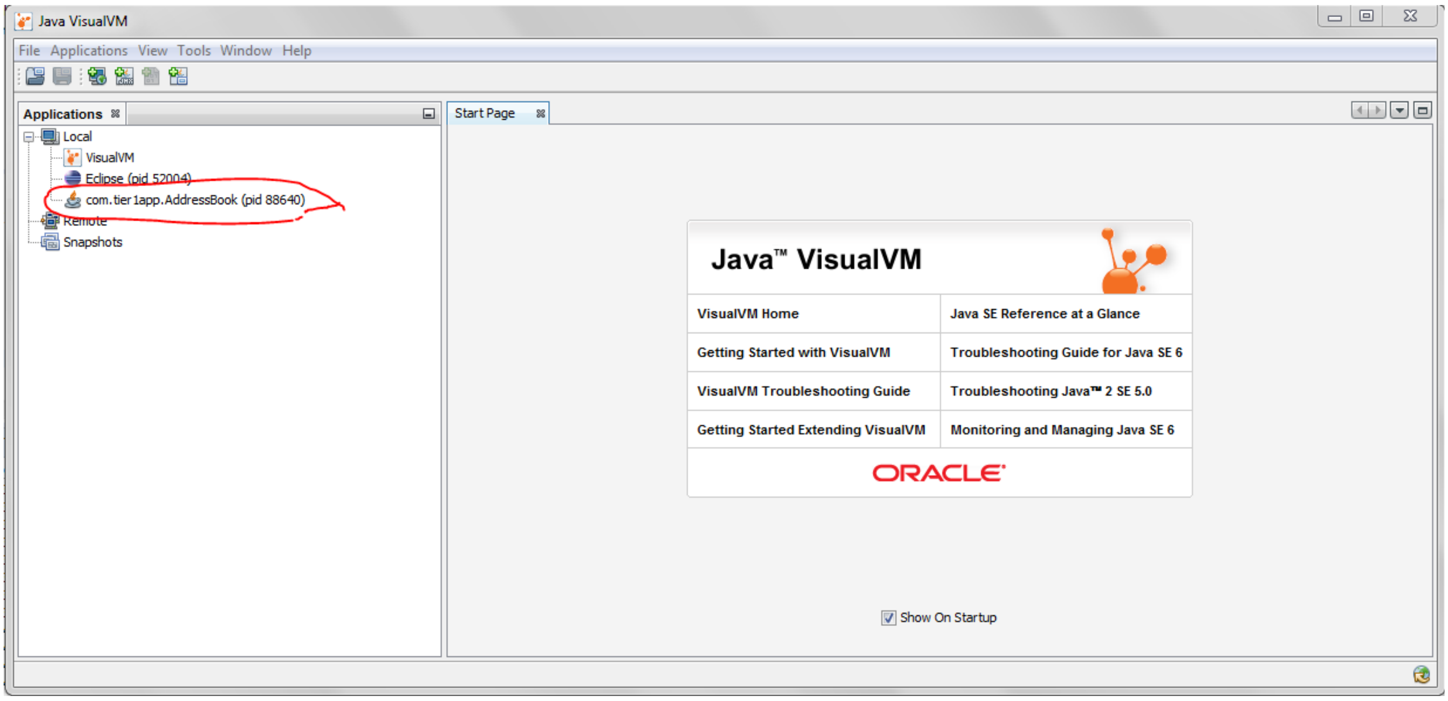

Java VisualVM
File Applications View Tools Window Help
Applications
Local
VisualVM
Eclipse (pid 52004)
com.tier1app.AddressBook (pid 88640)
Remote
Snapshots
Start Page
Java™ VisualVM
VisualVM Home
Java SE Reference at a Glance
Getting Started with VisualVM
Troubleshooting Guide for Java SE 6
VisualVM Troubleshooting Guide
Troubleshooting Java™ 2 SE 5.0
Getting Started Extending VisualVM
Monitoring and Managing Java SE 6
ORACLE
Show On Startup

Fig: Java Visual VM

Now go to the "Threads" tab. Click on the "Thread Dump" button as shown in the below image. Now Thread dumps would be generated.

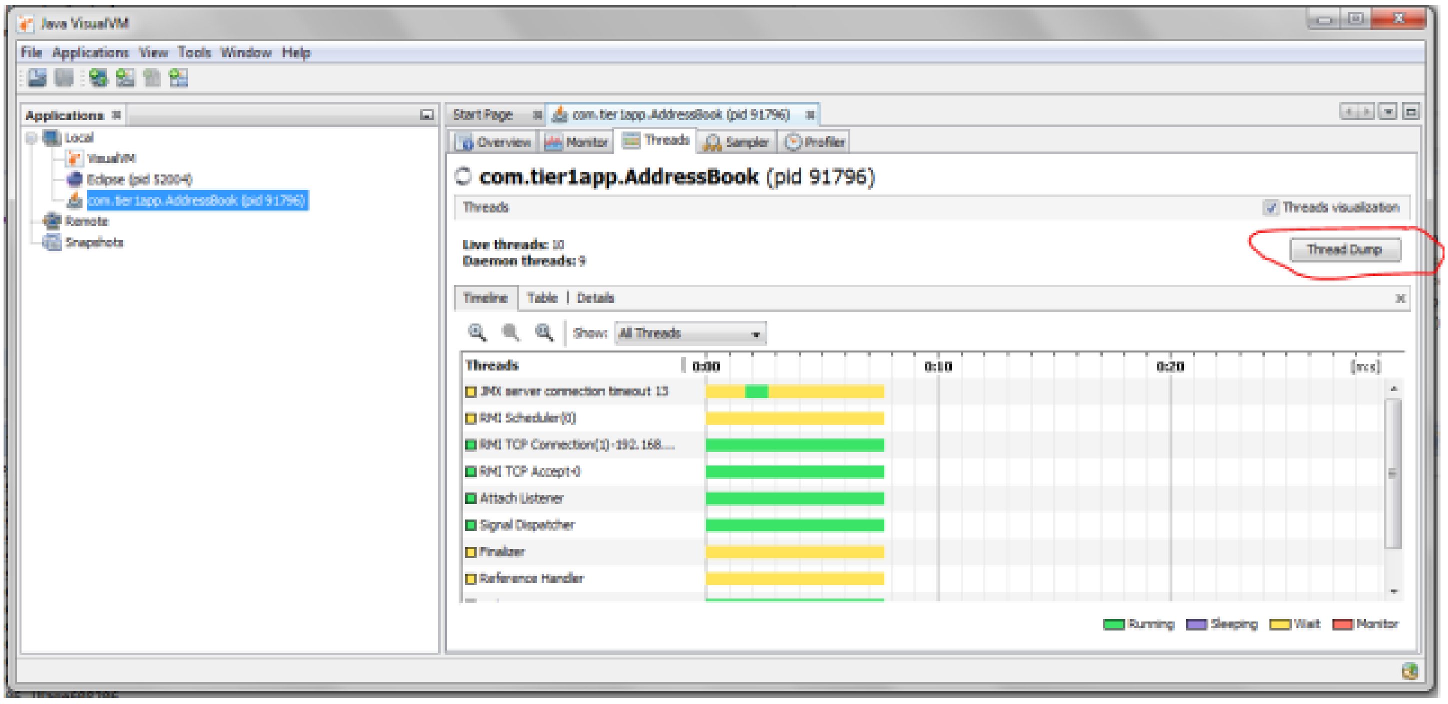
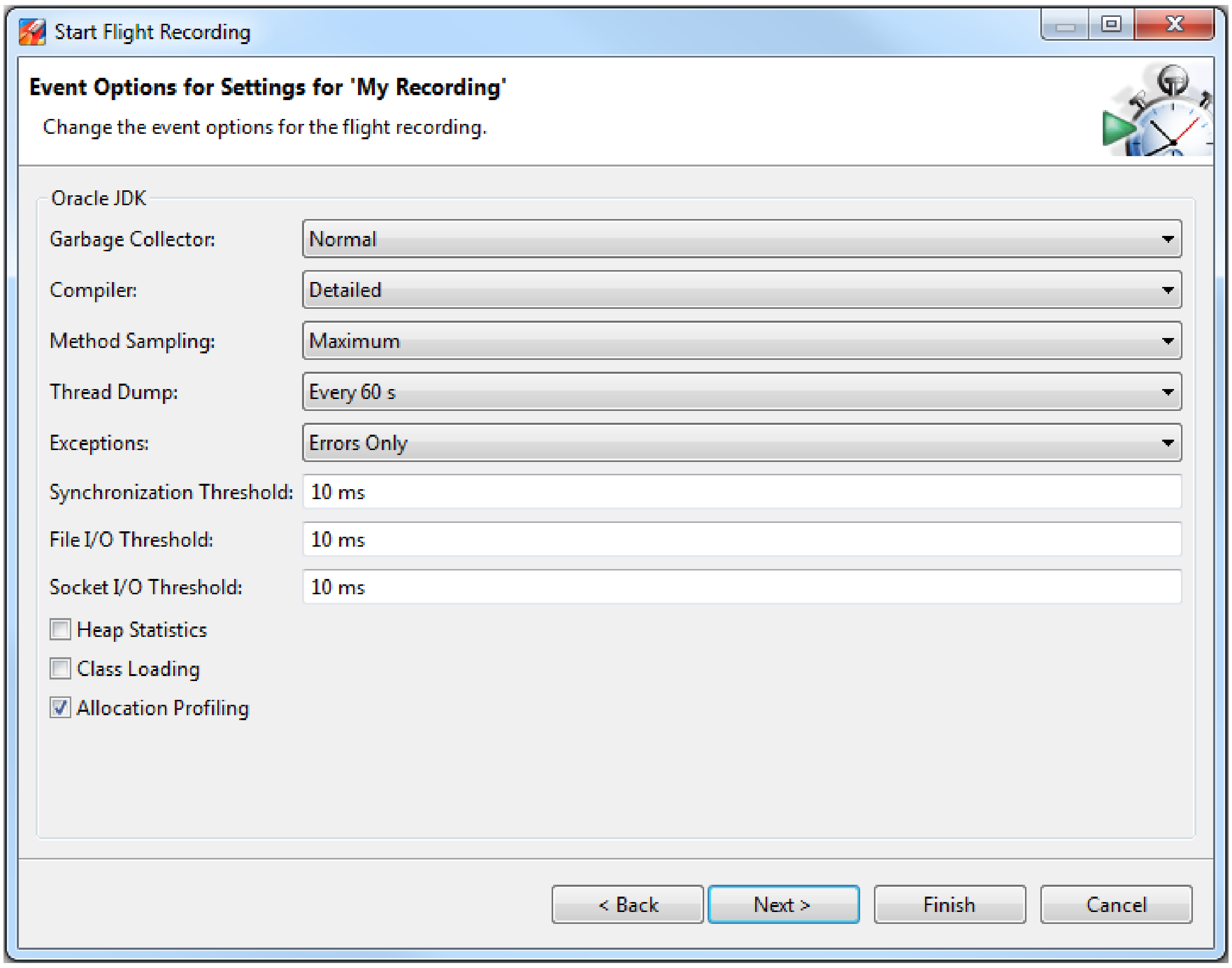


Fig: Highlighting "Thread Dump" button in the "Threads" tab

4. JMC

Java Mission Control (JMC) is a tool that collects and analyze data from Java applications running locally or deployed in production environments. This tool has been packaged into JDK since Oracle JDK 7 Update 40. This tool also provides an option to take thread dumps from the JVM. JMC tool is present in *JDK_HOMEbinjmc.exe*

Once you launch the tool, you will see all the Java processes that are running on your local host. Note: JMC also can connect with java processes running on the remote host. Now on the left panel click on the "Flight Recorder" option that is listed below the Java process for which you want to take thread dumps. Now you will see the "Start Flight Recording" wizard, as shown in the below figure.


Start Flight Recording

Event Options for Settings for 'My Recording'

Change the event options for the flight recording.

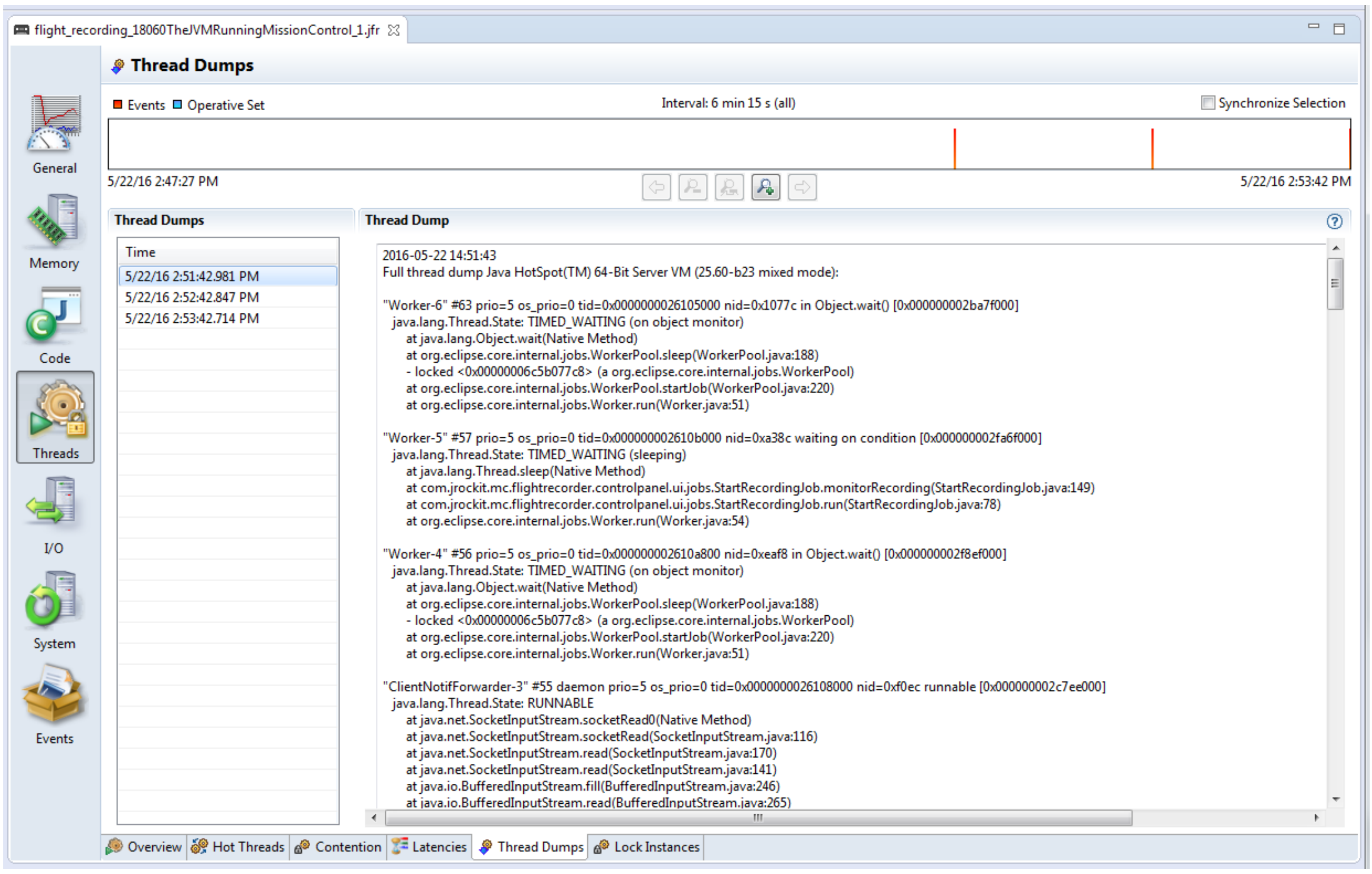
Oracle JDK

Garbage Collector:	Normal
Compiler:	Detailed
Method Sampling:	Maximum
Thread Dump:	Every 60 s
Exceptions:	Errors Only
Synchronization Threshold:	10 ms
File I/O Threshold:	10 ms
Socket I/O Threshold:	10 ms

- [ ] Heap Statistics
- [ ] Class Loading
- [✓] Allocation Profiling

Fig: Flight Recorder wizard showing 'Thread Dump' capture option.

Here in the "Thread Dump" field, you can select the interval in which you want to capture thread dump. As per the above example, every 60 seconds thread dump will be captured. After the selection is complete start the Flight recorder. Once the recording is complete, you will see the thread dumps in the "Threads" panel, as shown in the figure below.



Time
5/22/16 2:51:42.981 PM
5/22/16 2:52:42.847 PM
5/22/16 2:53:42.714 PM

Thread Dump

```
2016-05-22 14:51:43
Full thread dump Java HotSpot(TM) 64-Bit Server VM (25.60-b23 mixed mode):

"Worker-6" #63 prio=5 os_prio=0 tid=0x0000000026105000 nid=0x1077c in Object.wait() [0x000000002ba7f000]
  java.lang.Thread.State: TIMED_WAITING (on object monitor)
      at java.lang.Object.wait(Native Method)
      at org.eclipse.core.internal.jobs.WorkerPool.sleep(WorkerPool.java:188)
      - locked <0x00000006c5b077c8> (a org.eclipse.core.internal.jobs.WorkerPool)
      at org.eclipse.core.internal.jobs.WorkerPool.startJob(WorkerPool.java:220)
      at org.eclipse.core.internal.jobs.Worker.run(Worker.java:51)

"Worker-5" #57 prio=5 os_prio=0 tid=0x000000002610b000 nid=0xa38c waiting on condition [0x000000002fa6f000]
  java.lang.Thread.State: TIMED_WAITING (sleeping)
      at java.lang.Thread.sleep(Native Method)
      at com.jrockit.mc.flightrecorder.controlpanel.ui.jobs.StartRecordingJob.monitorRecording(StartRecordingJob.java:149)
      at com.jrockit.mc.flightrecorder.controlpanel.ui.jobs.StartRecordingJob.run(StartRecordingJob.java:78)
      at org.eclipse.core.internal.jobs.Worker.run(Worker.java:54)

"Worker-4" #56 prio=5 os_prio=0 tid=0x000000002610a800 nid=0xeaf8 in Object.wait() [0x000000002f8ef000]
  java.lang.Thread.State: TIMED_WAITING (on object monitor)
      at java.lang.Object.wait(Native Method)
      at org.eclipse.core.internal.jobs.WorkerPool.sleep(WorkerPool.java:188)
      - locked <0x00000006c5b077c8> (a org.eclipse.core.internal.jobs.WorkerPool)
      at org.eclipse.core.internal.jobs.WorkerPool.startJob(WorkerPool.java:220)
      at org.eclipse.core.internal.jobs.Worker.run(Worker.java:51)

"ClientNotifForwarder-3" #55 daemon prio=5 os_prio=0 tid=0x0000000026108000 nid=0xf0ec runnable [0x000000002c7ee000]
  java.lang.Thread.State: RUNNABLE
      at java.net.SocketInputStream.socketRead0(Native Method)
      at java.net.SocketInputStream.socketRead(SocketInputStream.java:116)
      at java.net.SocketInputStream.read(SocketInputStream.java:170)
      at java.net.SocketInputStream.read(SocketInputStream.java:141)
      at java.io.BufferedInputStream.fill(BufferedInputStream.java:246)
      at java.io.BufferedInputStream.read(BufferedInputStream.java:265)
```

Fig: Showing captured 'Thread Dump' in JMC.

5. Windows (Ctrl + Break)

This option will work only in Windows Operating system.

- Select command line console window in which you have launched application.
- Now on the console window issue the "**Ctrl + Break**" command.

This will generate thread dump. A thread dump will be printed on the console window itself.

Note 1: in several laptops (like my Lenovo T series) "Break" key is removedJ. In such circumstance, you have to google to find the equivalent keys for the "Break." In my case, it turned out that "Function key + B" is the equivalent of "Break" key. Thus I had to use "Ctrl + Fn + B" to generate thread dump.s

Note 2: But one disadvantage with the approach is thread dump will be printed on the windows console itself. Without getting the thread dump in a file format, it's hard to use the thread dump analysis tools such as http://fastthread.io. Thus when you launch the application from the command line, redirect the output a text file i.e. Example if you are launching the application "SampleThreadProgram", you would issue the command:

```
java -classpath . SampleThreadProgram
```

instead, launch the SampleThreadProgram like this

```
  java -classpath . SampleThreadProgram > C:workspacethreadDump.txt 2>&1
```

Thus when you issue "Ctrl + Break" thread dump will be sent to **C:workspacethreadDump.txt**file.

6. ThreadMXBean

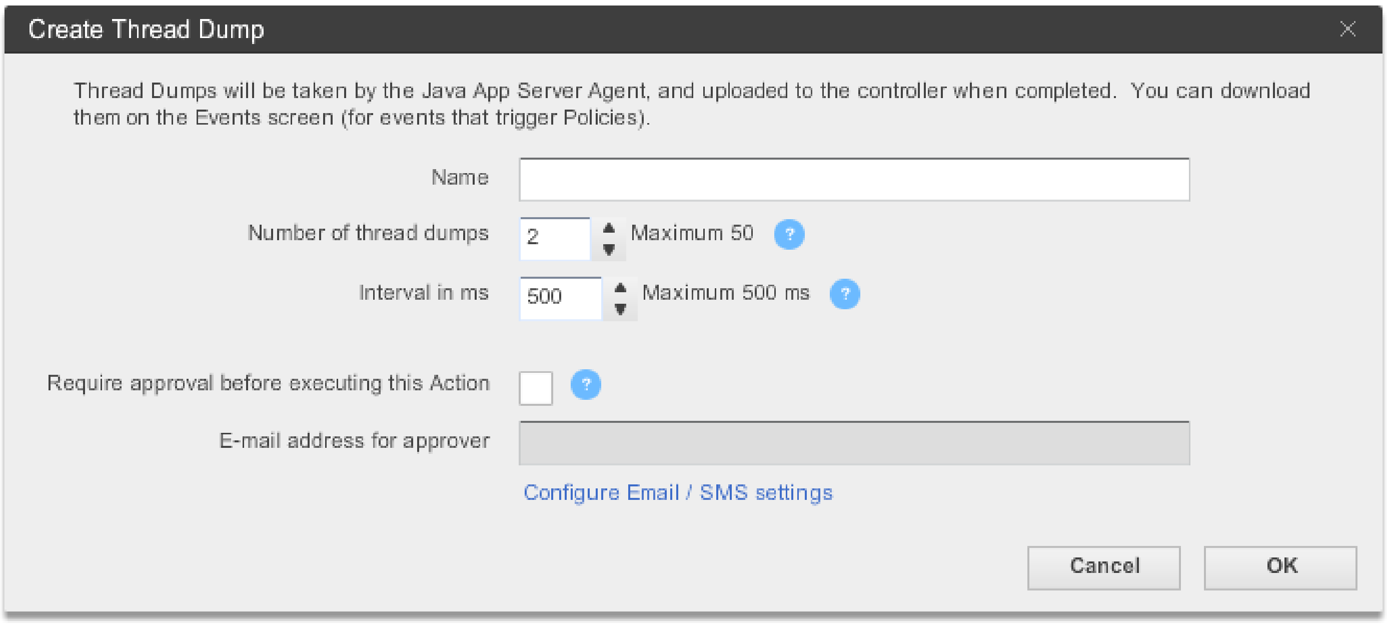
Since JDK 1.5 ThreadMXBean has been introduced. This is the management interface for the thread system in the Java Virtual Machine. Using this interface also you can generate thread dumps. You only have to write few lines of code to generate thread dumps programmatically. Below is a skeleton implementation on ThreadMXBean implementation, which generates Thread dump from the application.

```
      public void  dumpThreadDump() {
          ThreadMXBean threadMxBean = 
  ManagementFactory.getThreadMXBean();
          for (ThreadInfo ti : threadMxBean.dumpAllThreads(true, true)) {
              System.out.print(ti.toString());
          }
      }
```

7. APM Tool – App Dynamics

Few Application Performance Monitoring tools provide options to generate thread dumps. If you are monitoring your application through App Dynamics (APM tool), below are the instructions to capture thread dump:

1. Create an action, selecting **Diagnostics->Take a thread dump** in the Create Action window.
2. Enter a name for the action, the number of samples to take, and the interval between the thread dumps in milliseconds.
3. If you want to require approval before the thread dump action can be started, check the Require approval before this Action checkbox and enter the email address of the individual or group that is authorized to approve the action. See Actions Requiring Approval for more information.
4. Click **OK**.



Create Thread Dump

Thread Dumps will be taken by the Java App Server Agent, and uploaded to the controller when completed. You can download them on the Events screen (for events that trigger Policies).

Name:

Number of thread dumps: 2 Maximum 50

Interval in ms: 500 Maximum 500 ms

Require approval before executing this Action: ☐

E-mail address for approver:

Fig: App dynamics thread dump capturing wizard

8. JCMD

The jcmd tool was introduced with Oracle's Java 7. It's useful in troubleshooting issues with JVM applications. It has various capabilities such as identifying java process Ids, acquiring heap dumps, acquiring thread dumps, acquiring garbage collection statistics, ….

Using the below JCMD command you can generate thread dump:

```
jcmd <pid> Thread.print > <file-path>
```

where

pid: is the Process Id of the application, whose thread dump should be captured

file-path: is the file path where thread dump will be written in to.

Example:

```
jcmd 37320 Thread.print > /opt/tmp/threadDump.txt
```

As per the example thread dump of the process would be generated in /opt/tmp/threadDump.txt file.

Conclusion