Using JConsole to Monitor Your IBM Notes Client

by Julian Robichaux, panagenda
originally published on socialbizug.org, May 2014

Beginning with Java 5.0, Oracle added a monitoring app called JConsole (http://docs.oracle.com/javase/6/docs/technotes/guides/management/jconsole.html) to their JDK distributions. This app shows you not only how memory, threads, and CPU are being used in real-time by a Java application, it also taps into the JMX MBeans (http://docs.oracle.com/javase/1.5.0/docs/api/javax/management/package-summary.html) that have been registered with the Java VM. This article will discuss how to connect JConsole to your IBM Notes client.

Step 1: Download and Install a Recent Java JDK

You've probably already got one, but just in case, make sure that you have a Java JDK (5.0 or higher) installed on your machine. If you're not sure if you have the JDK or not, just go to the "bin" folder of your current Java install and look for the jconsole.exe program. If it's not there, you probably only have a JRE.

The screenshots in this article are Java 6.0 running on Windows 7.

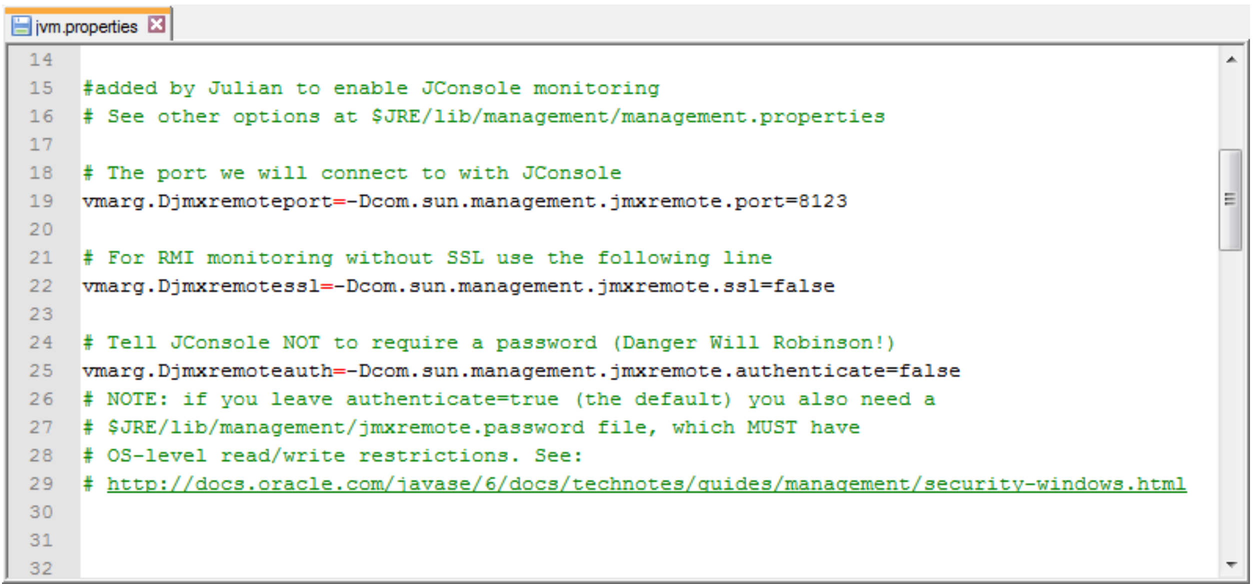
Step 2: Edit your jvm.properties file

Open the {Notes}/framework/rcp/deploy/jvm.properties file in a text editor and add the following lines:

```
vmarg.Djmxremoteport=-Dcom.sun.management.jmxremote.port=8123
vmarg.Djmxremotessl=-Dcom.sun.management.jmxremote.ssl=false
vmarg.Djmxremoteauth=-Dcom.sun.management.jmxremote.authenticate=false
```

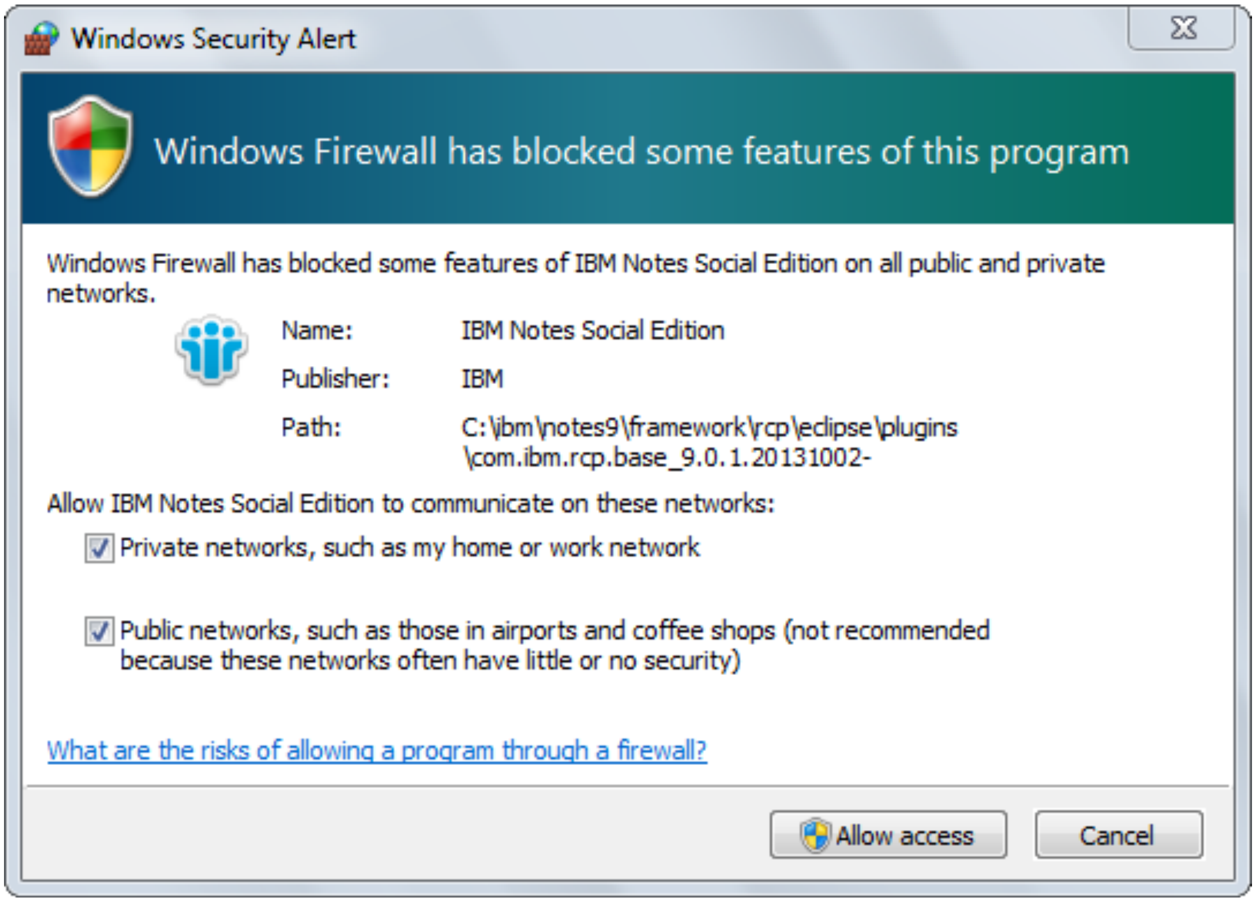
The first line tells the Notes client JVM to listen on port 8123 for JMX clients like JConsole (feel free to use a different port if you'd like). The second line tells it not to use SSL for the connection (which it would do by default). And the third line tells it not to require a username and password for the connection.

You can also leave yourself comments in the jvm.properties file so you can remember what all this stuff means, by prefixing the comment lines with a hash (#) like this:



```

#added by Julian to enable JConsole monitoring
# See other options at $JRE/lib/management/management.properties

# The port we will connect to with JConsole
vmarg.Djmxremoteport=-Dcom.sun.management.jmxremote.port=8123

# For RMI monitoring without SSL use the following line
vmarg.Djmxremotessl=-Dcom.sun.management.jmxremote.ssl=false

# Tell JConsole NOT to require a password (Danger Will Robinson!)
vmarg.Djmxremoteauth=-Dcom.sun.management.jmxremote.authenticate=false
# NOTE: if you leave authenticate=true (the default) you also need a
# $JRE/lib/management/jmxremote.password file, which MUST have
# OS-level read/write restrictions. See:
# http://docs.oracle.com/javase/6/docs/technotes/guides/management/security-windows.html



```

Save and close the file, then start your Notes client. You might receive a firewall warning like this:



Windows Firewall has blocked some features of IBM Notes Social Edition on all public and private networks.

Name:	IBM Notes Social Edition
Publisher:	IBM
Path:	C:\ibm\notes9\framework\rcp\eclipse\plugins \com.ibm.rcp.base_9.0.1.20131002-

Allow IBM Notes Social Edition to communicate on these networks:

[✓] Private networks, such as my home or work network

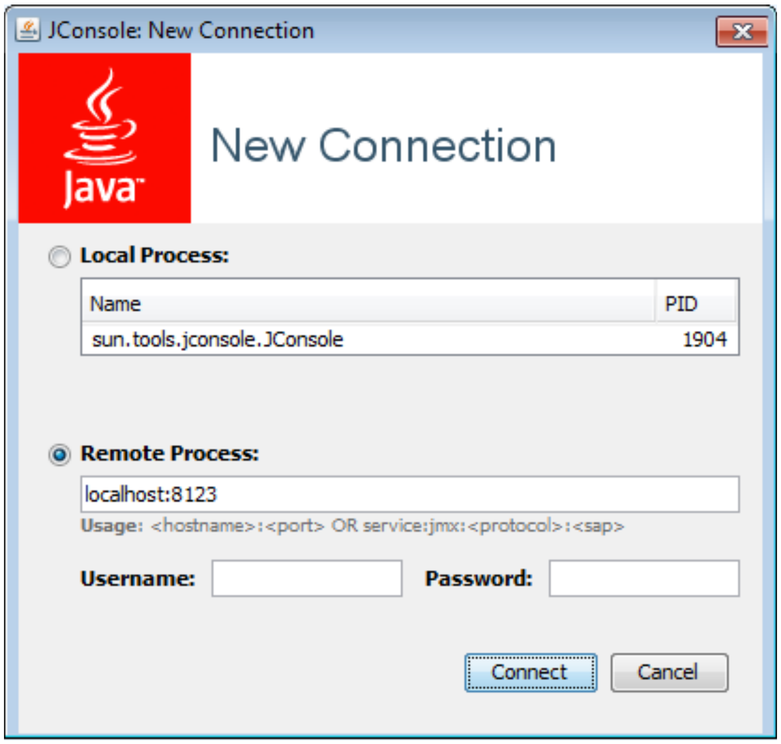
[✓] Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing a program through a firewall?
Allow access
Cancel

If so, just click “Allow access” and continue. This is the Notes JVM attempting to listen for the JConsole connection.

Step 3: Connect with JConsole
Open a command prompt. If your Java JDK “bin” directory is on the path, you can simply type “jconsole” to launch the program. Otherwise you will need to navigate to the JDK “bin” directory first, and then type “jconsole”. You will see a window like this:

To connect to your Notes client, choose the “Remote Process” option and use an address of “localhost:8123” in the remote process field (8123 is the port we used in the jvm.properties file earlier). Because we set authenticate=false, no username or password is required. Just click the “Connect” button.

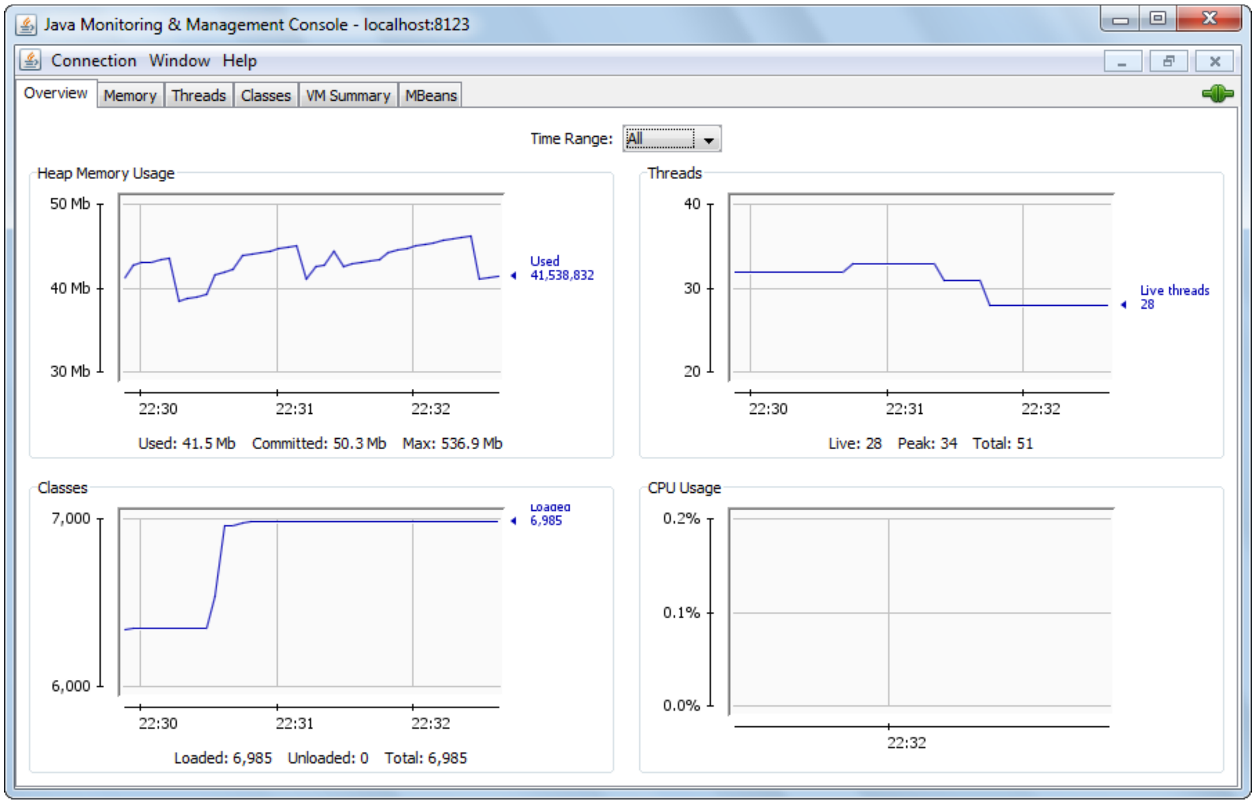


Name	PID
sun.tools.jconsole.JConsole	1904

Remote Process: localhost:8123

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

You should now see the full JConsole interface. It will look something like this:



Time Range: All

Used 41,538,832

Used: 41.5 Mb Committed: 50.3 Mb Max: 536.9 Mb

Live threads 28

Live: 28 Peak: 34 Total: 51

Loaded 6,985

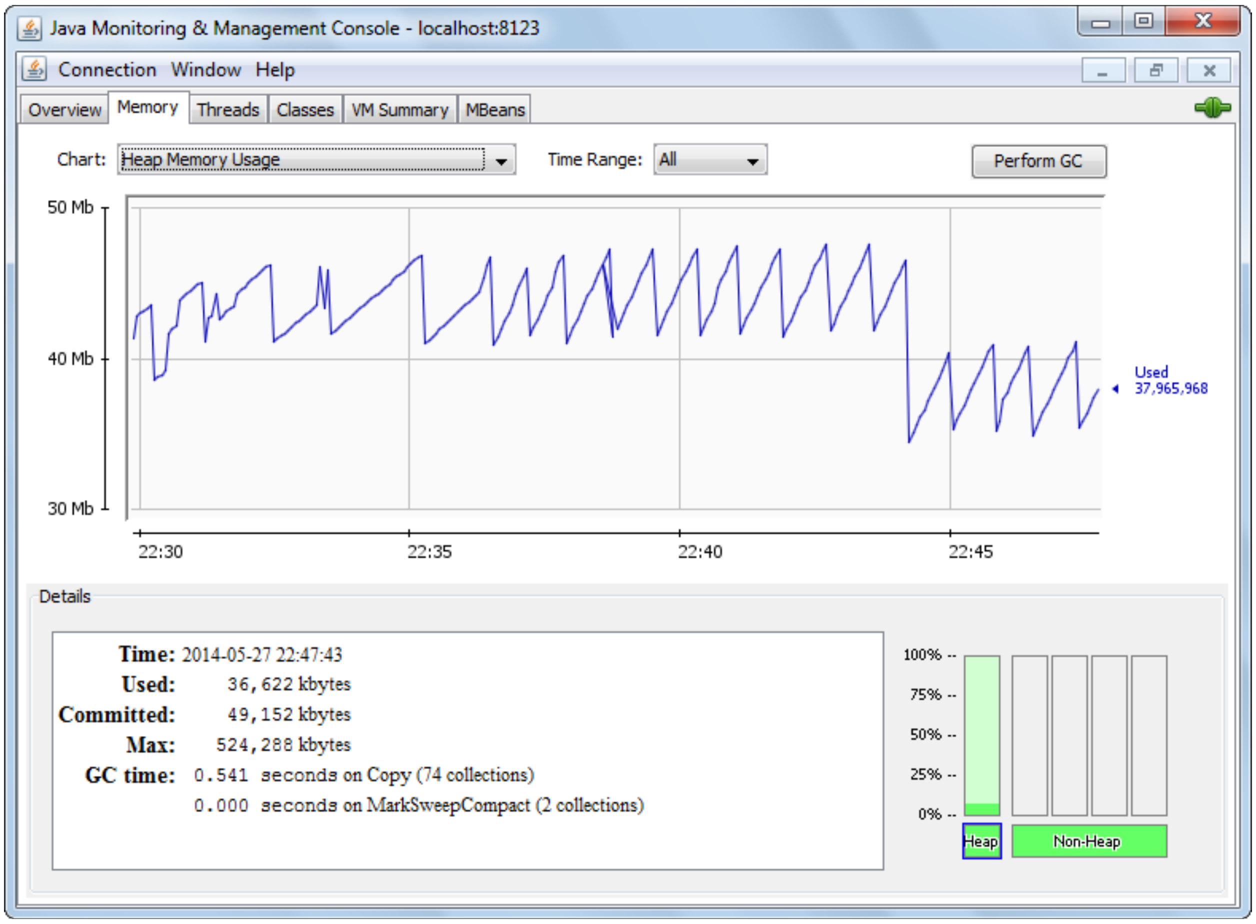
Loaded: 6,985 Unloaded: 0 Total: 6,985

The graphs on the Overview tab will tick along at a regular interval (default is every 4 seconds, although you can change it with a command line argument http://docs.oracle.com/javase/6/docs/technotes/tools/share/jconsole.html), showing changes in heap memory, total threads, and total classes loaded.

You will notice that the CPU graph remains at zero. Mine does anyway. I suspect that the problem is related to APAR IV19214 (http://www-01.ibm.com/support/docview.wss?uid=swg1IV19214) and will be (or has been) fixed in a newer version of IBM Java. If the problem is something else or you know a way to fix it, please leave a note in the comments.

Checking Memory Usage
Okay, great, we're using JConsole. What can we use it for?

First of all, keep in mind that this is only tapping in to the Eclipse part(s) of the IBM Notes client.



Time: 2014-05-27 22:47:43
Used: 36,622 kbytes
Committed: 49,152 kbytes
Max: 524,288 kbytes
GC time: 0.541 seconds on Copy (74 collections)
0.000 seconds on MarkSweepCompact (2 collections)

Not the "classic" Notes operations, not LotusScript, and not Java agents. Just the Eclipse/Expeditor wrapper and any plugins you have installed.

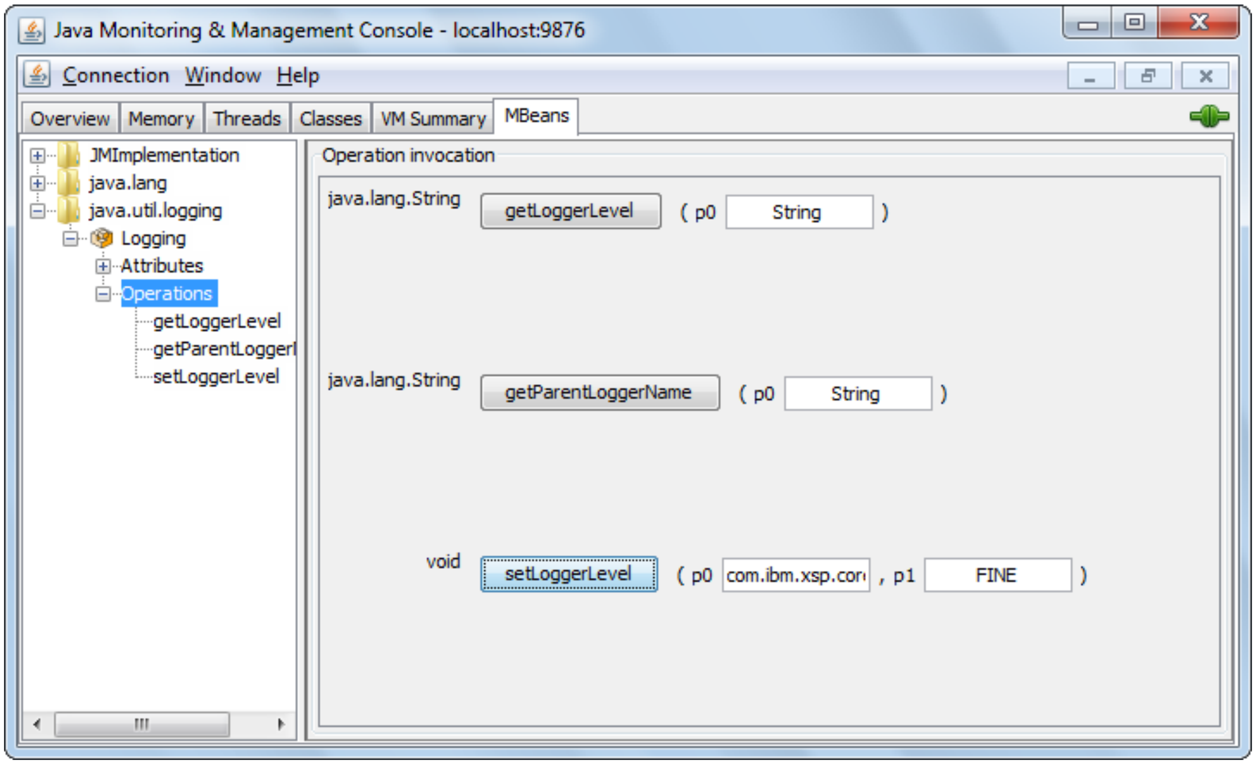
That being said, the most obvious use of JConsole is for monitoring the memory use of your Notes client, and the Domino Designer (DDE) client. Ideally you want a kind of sawtooth graph of memory, like this:

This will show memory being used and released on a regular basis, and the “Memory” tab will also keep track of how close you are to the maximum amount of memory allotted to your client. The Memory tab also has a button to force garbage collection, although that’s not something you’ll ever really need to do.

If you want to see some of the interesting memory tricks you can play with JConsole, like turning on verbose garbage collection or checking object finalization information, please read the excellent article “Using JConsole to Monitor Applications” (http://www.oracle.com/technetwork/articles/java/jconsole-1564139.html).

Adjusting Log Levels
Another thing that JConsole allows you to do easily is change the logging levels of plugins that are currently running in the client (any levels you set will be reset when the client is restarted). You can then see the log messages in the trace log of the Notes client from the menu option Help > Support > View Trace.

To set a logger level, go to the “MBeans” tab and navigate to the java.util.logging Logging Operations item:



java.lang.String getLoggerLevel (p0 String)

java.lang.String getParentLoggerName (p0 String)

void setLoggerLevel (p0 com.ibm.xsp.cor , p1 FINE)

Next to the “setLoggerLevel” button, enter the name of your logger in the first field and the level (like “FINE”) in the second field, then click the setLoggerLevel button. If you choose a logger that doesn’t exist — either because you typed the name wrong or because the corresponding plugin hasn’t been activated yet — you will get an error message.

Securing Access

When we set this up, we explicitly turned authentication off, which might have set off a few alarm bells. The JMX listener will only accept connections from localhost by default, which gives you some measure of security. If you decide that you want to require authentication, you also need a {Notes}/jvm/lib/management/jmxremote.password file, which MUST have OS-level read/write restrictions. For information on how to set this up, please read the information in these two links:

http://docs.oracle.com/javase/6/docs/technotes/guides/management/agent.html#gdeup
http://docs.oracle.com/javase/6/docs/technotes/guides/management/security-windows.html

There are a lot of other options (and comments on what the options mean) in the {Notes}//lib/management/management.properties file. The Oracle documentation also contains a lot of details on configuration and use at: http://docs.oracle.com/javase/6/docs/technotes/guides/management/jconsole.html