Capturing Virtual Bytes
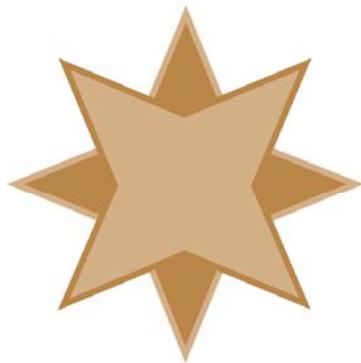# To A PerfMon Log File

A White Paper From

GOLDSTAR
SOFTWARE

www.GoldstarSoftware.com

For more information, see our web site at
http://www.goldstarsoftware.com

Capturing Virtual Bytes to A PerfMon Log File
**Last Updated: 09/07/2010**

The Pervasive database engine, when running as a 32-bit process on a 32-bit server, has a limited addressing space of 2GB. If it is running on a 64-bit server, the available memory space is a bit better (4GB), but it is still something that should be monitored.

In this paper, we are going to investigate a few different ways of monitoring the Virtual Bytes for the database engine process. Prior to reading this, you may wish to also review the two white papers from http://www.goldstarsoftware.com/press.asp on **Understanding Server Memory Counters** and **Interpreting Server Memory Counters**.

Spot-Checking Your Virtual Bytes Manually

The easiest way to monitor the Virtual Bytes is to simply check on it periodically with PerfMon, as described in the papers above. We will repeat the details here for completeness.

First, start up Performance Monitor by selecting Start/Run and entering PerfMon.
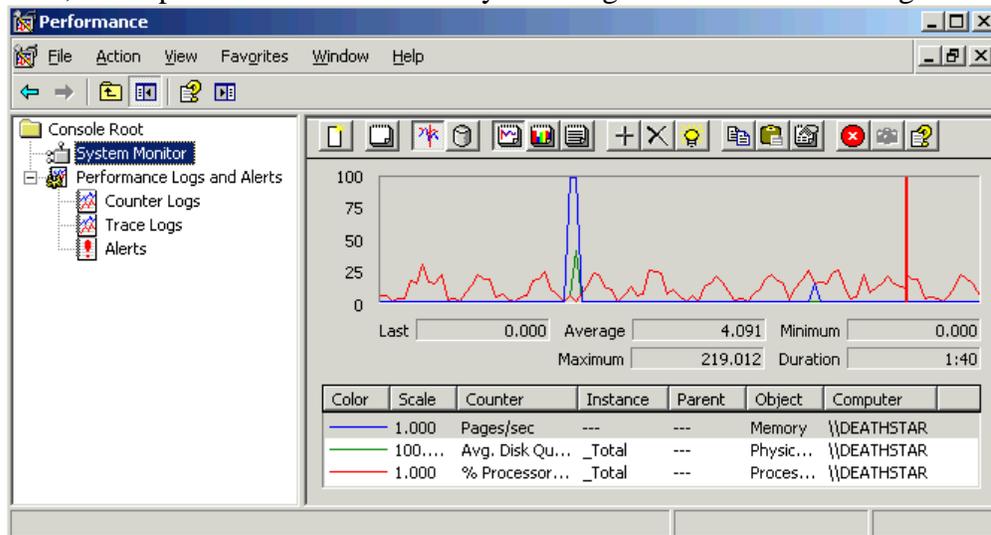
Then, click on the plus sign (+) on the tool bar and add the setting that you want. In this case, we want to add the **Virtual Bytes** value, so select the **Process** object, the **Virtual Bytes** counter, and the **ntdbsmgr** instance.
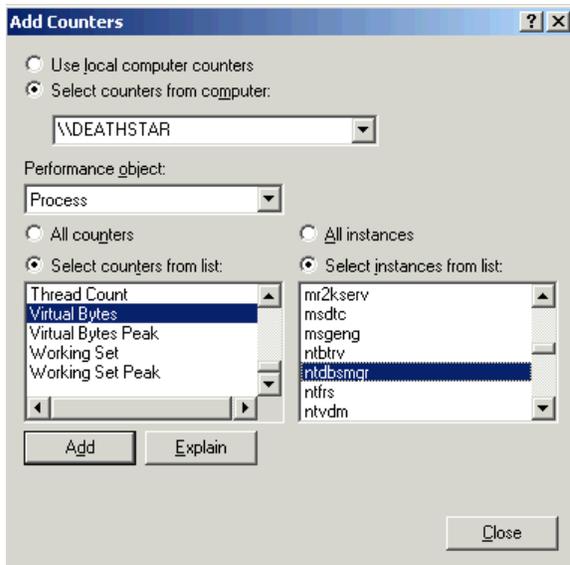
Click **Add**, then **Close**, and you'll have added the counter to the PerfMon screen. Then, just click on it and read the **Last** value to find out where you are right now.
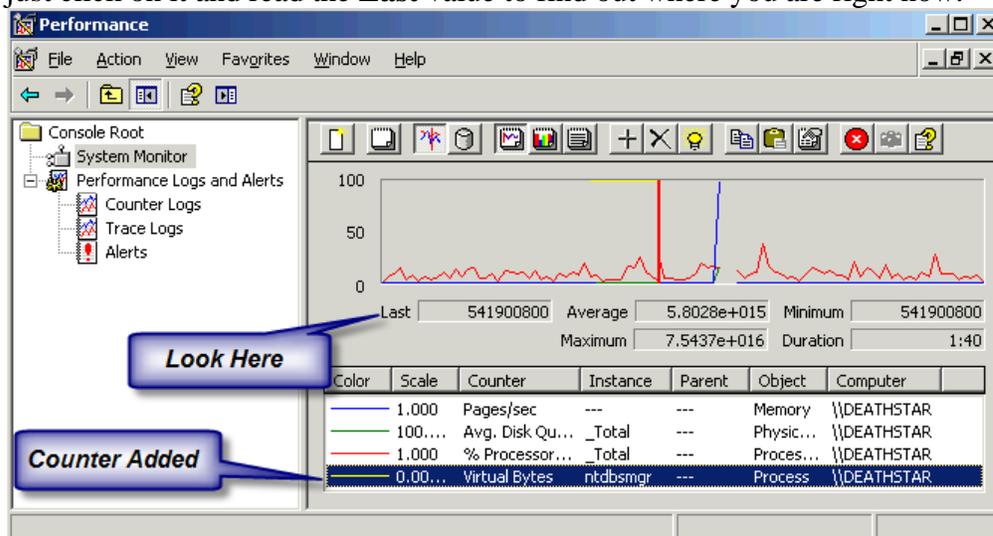

## Generating a Log File With Virtual Bytes Counter Values

If you are trying to find out WHEN the virtual bytes are being used up, then watching it manually won't be very helpful, since it will change slowly over time, and it may take weeks for the engine to leak enough memory to reach the maximum addressing space.

A better solution is to write the counter values to a log file, which can be done using these steps.
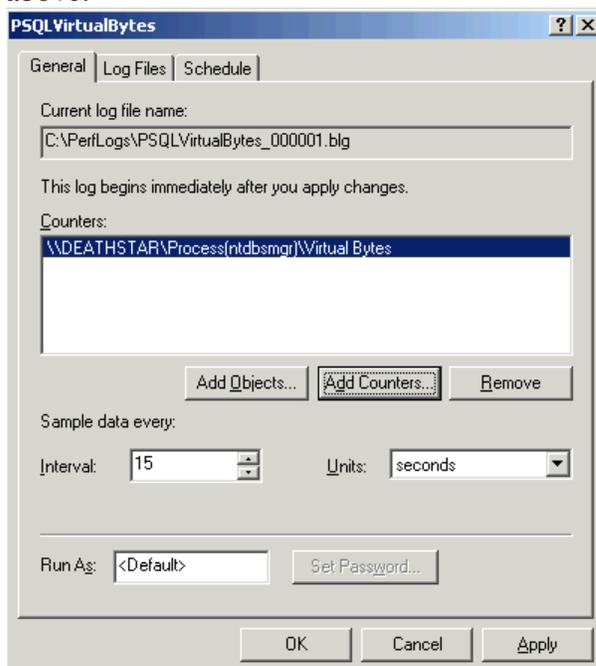
First, start up PerfMon and click on the **Counter Logs** entry in the tree on the left.
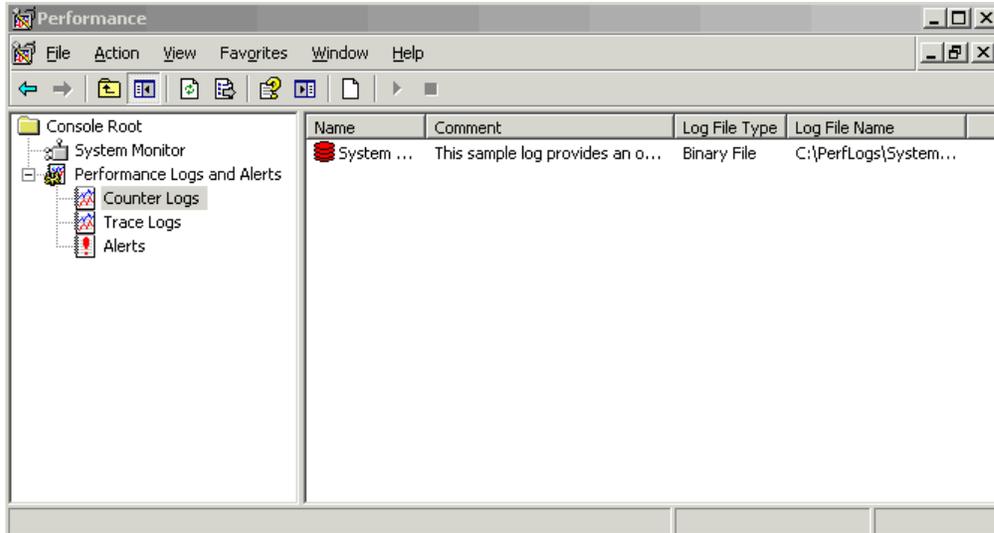
Then, right click in the space and select **New Log Settings...**, where you will get a dialog box asking for a name. Give it a useful name and click **OK**.
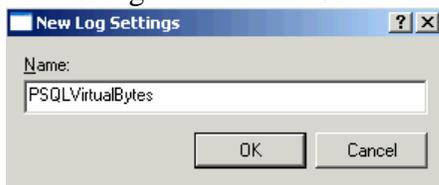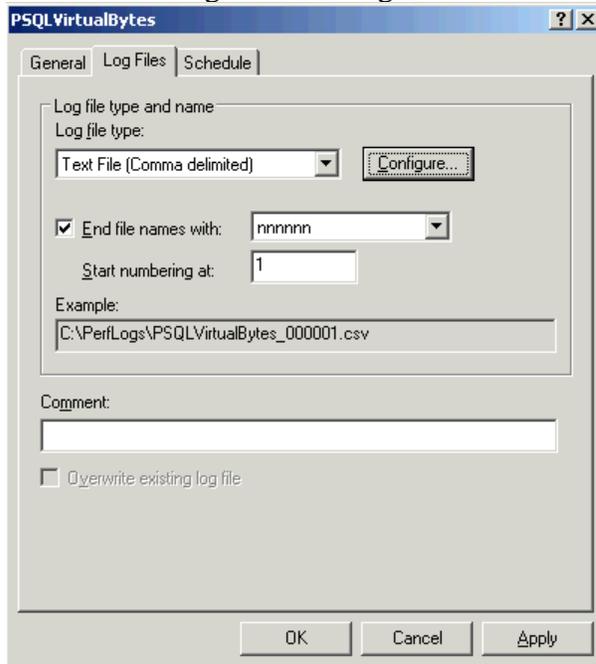

From the next dialog box, click **Add counters...** and add the counter exactly as indicated above.

Note that the default sample rate interval is every 15 seconds. This is probably overkill for this setting (unless you are leaking memory very rapidly). You may wish to change this to every 1 minute or perhaps even higher.

Click on the Log Files tab to get to the next dialog.

As we are going to graph this data in Excel, we want to change the **Log file type...** setting from the default of a **Binary File** to **Text File (Comma Delimited)**. The rest of the settings can remain the same. The Comment is optional, but you can add it if you want. If the default directory is unacceptable for your log files (remember that you don't want to run the C: drive out of disk space), then click on the **Configure...** button and you can change the path and file names.

Click **OK** to go back to the main PerfMon screen, and the counter will be started. This is evidenced by the green database icon in the **Counter Logs** listing.
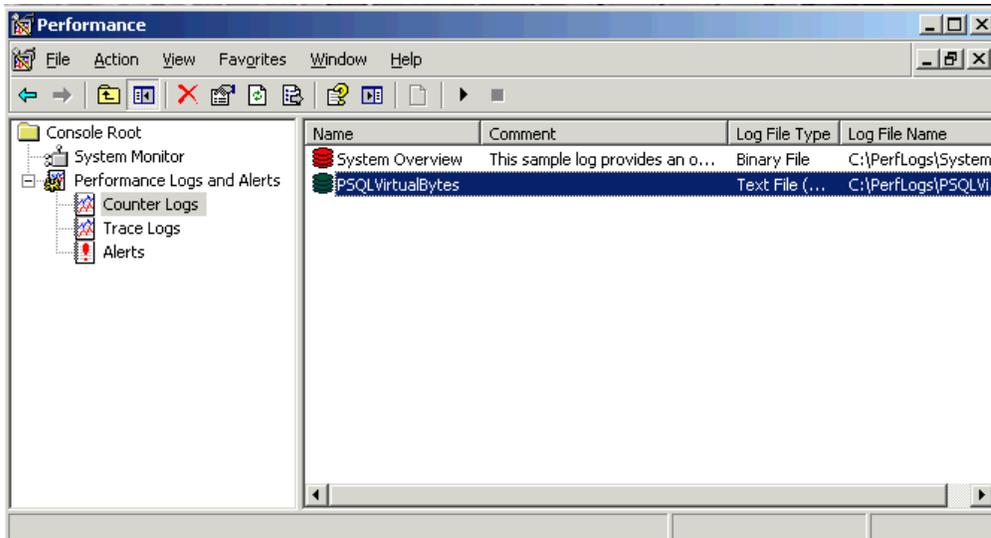
Now, let this run for a while to collect useful data. When you have enough data in the log file, right click on the Counter Log that you just created and select **Stop**.   You can later restart the log by right-clicking and selecting **Start**.

Reviewing the Log File Data

When you are done collecting data, you can directly open the log file that you created. If you open it with NotePad, then you'll see something like this:
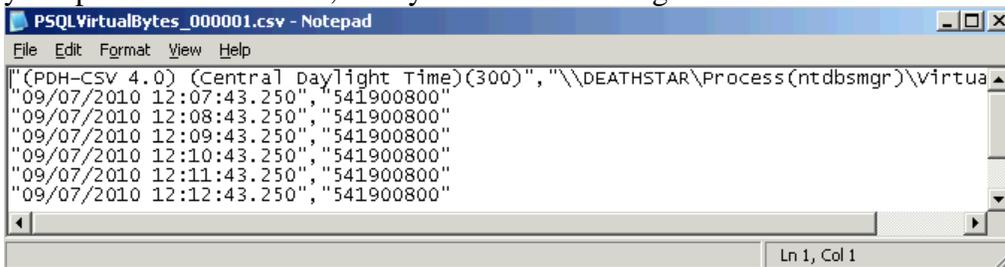

You can scroll through the data file and watch the Virtual Bytes values go up and down with respect to the timestamp field in the first column. This amount of detail can be overwhelming, though, especially for a long log file where the numbers are rapidly changing, so we'll look for a better solution.

Graphing a Log File In Microsoft Excel

An easier way to analyze the resulting log file data is to quickly graph it via Microsoft Excel. To do this, we'll be using Exccel 2007 as an example, but other versions should work similarly.
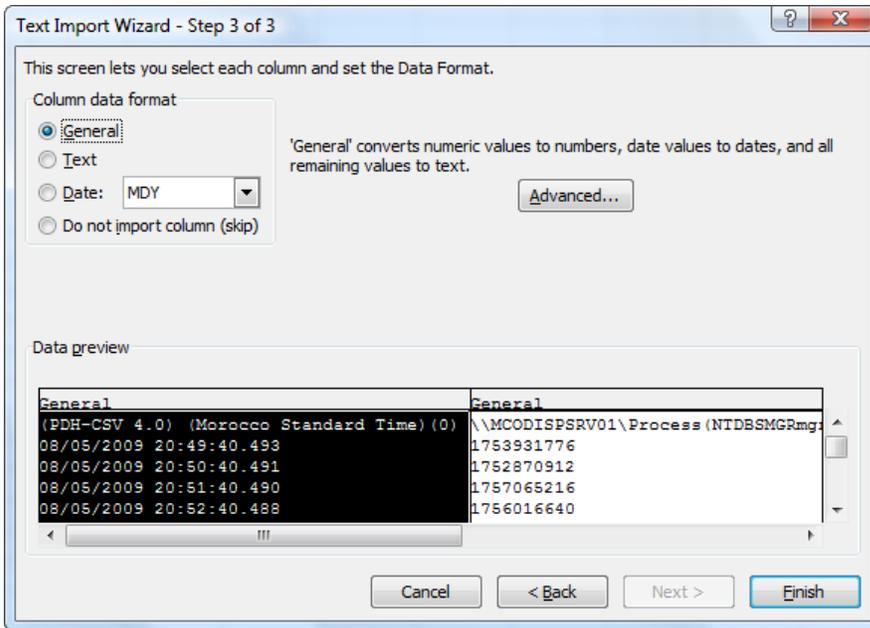
First, start Excel, then go to the Data menu and select From Text. Select your CSV file containing the log data, and click OK. You'll see the Text Import Wizard page.
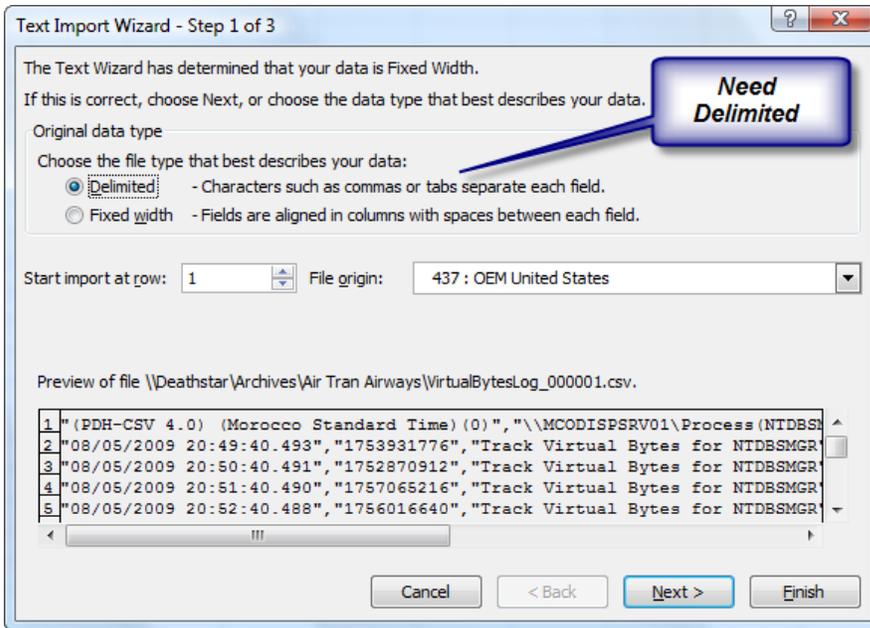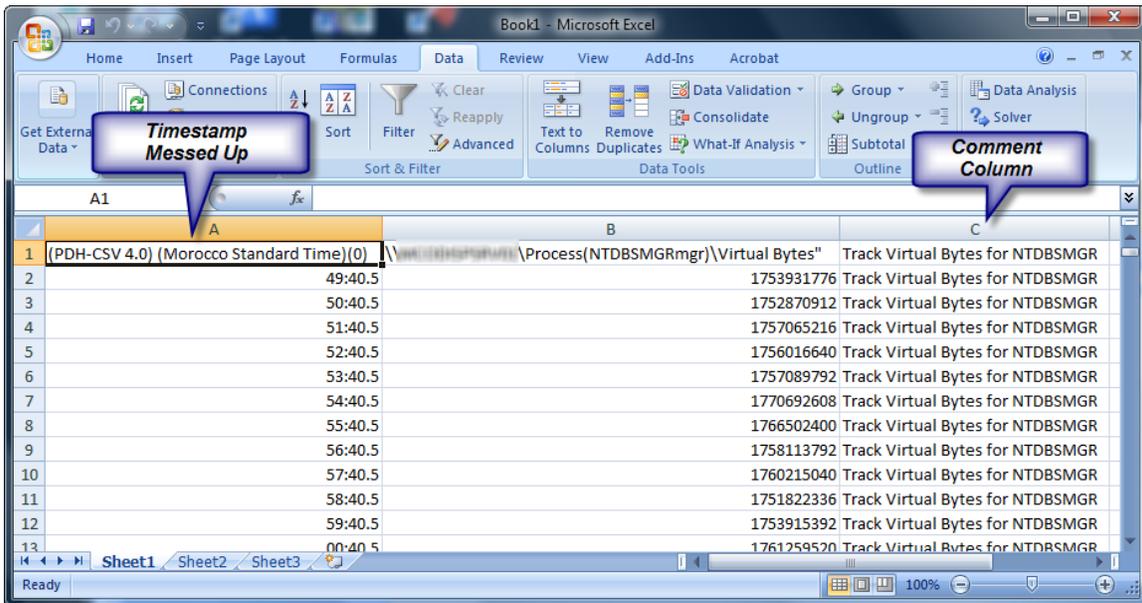
Be sure that you have selected "Delimited" and click **Next**.
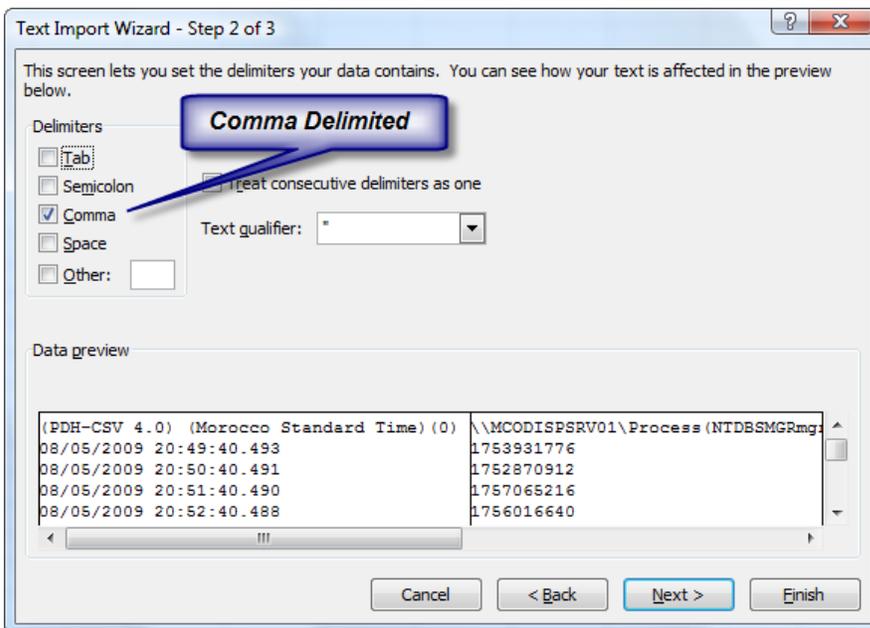
Be sure that you have selected **Comma** as the delimiter and click **Next** again.

On the last dialog, just click **Finish** to leave all of the defaults and load the data. You'll now see that Excel has loaded the data for you. However, there are some problems with it, as you see here:

The first thing we want to do is remove the third column, since this is just a comment field. So, click on the Column C and delete the column completely. Next, replace the two header elements with shorter names to make it easier to work with, and you should have something more resembling this page:
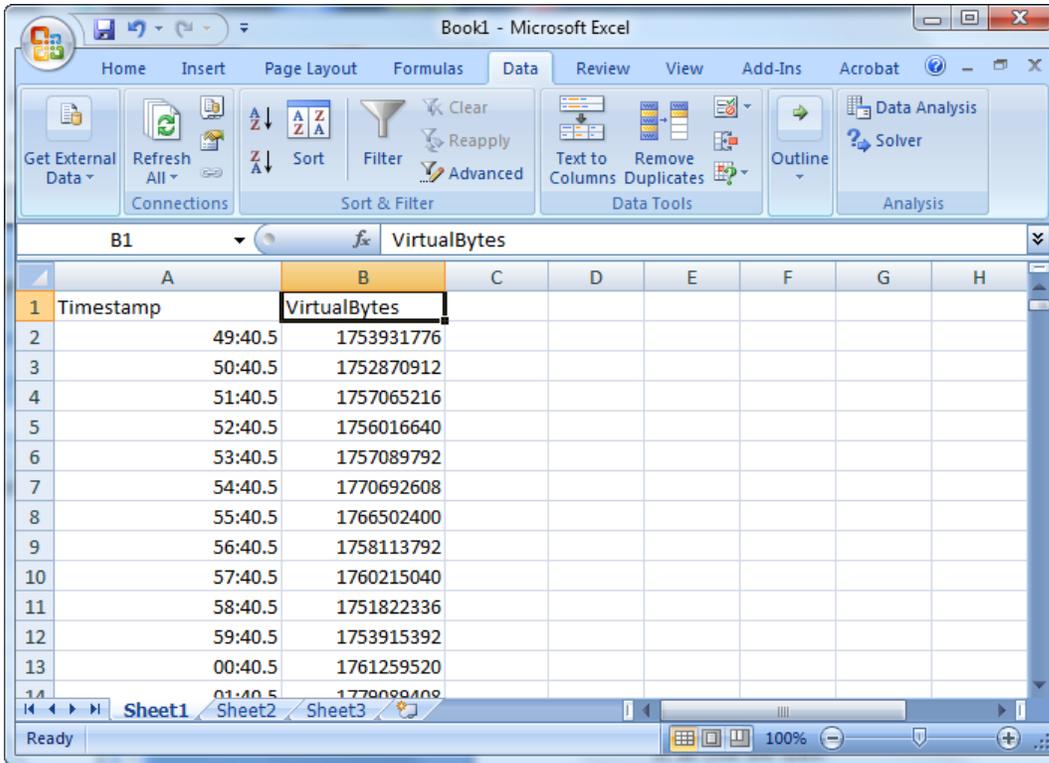
Next, we need to fix the formatting of the first column. Select the entire column, then right-click and select **Format Cells...** to get to this dialog box.
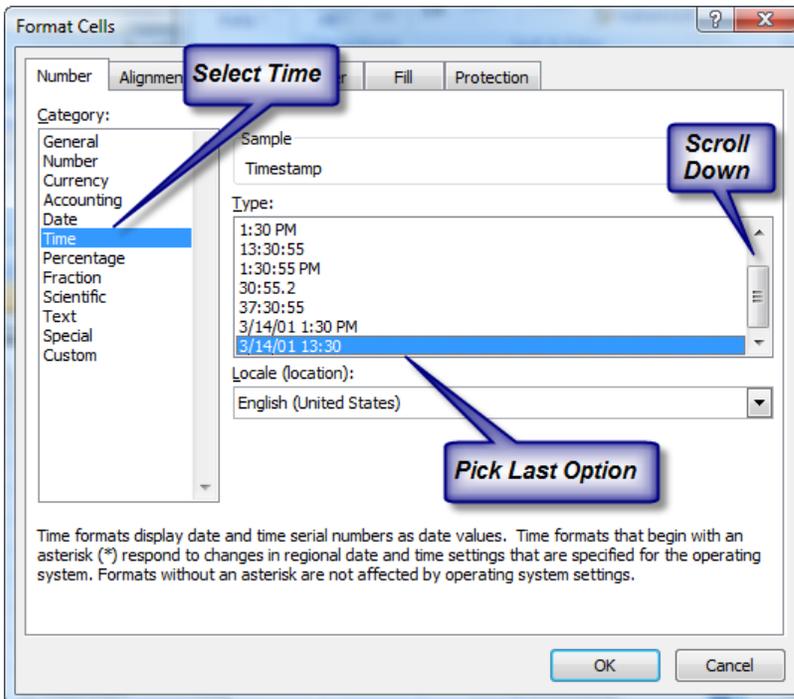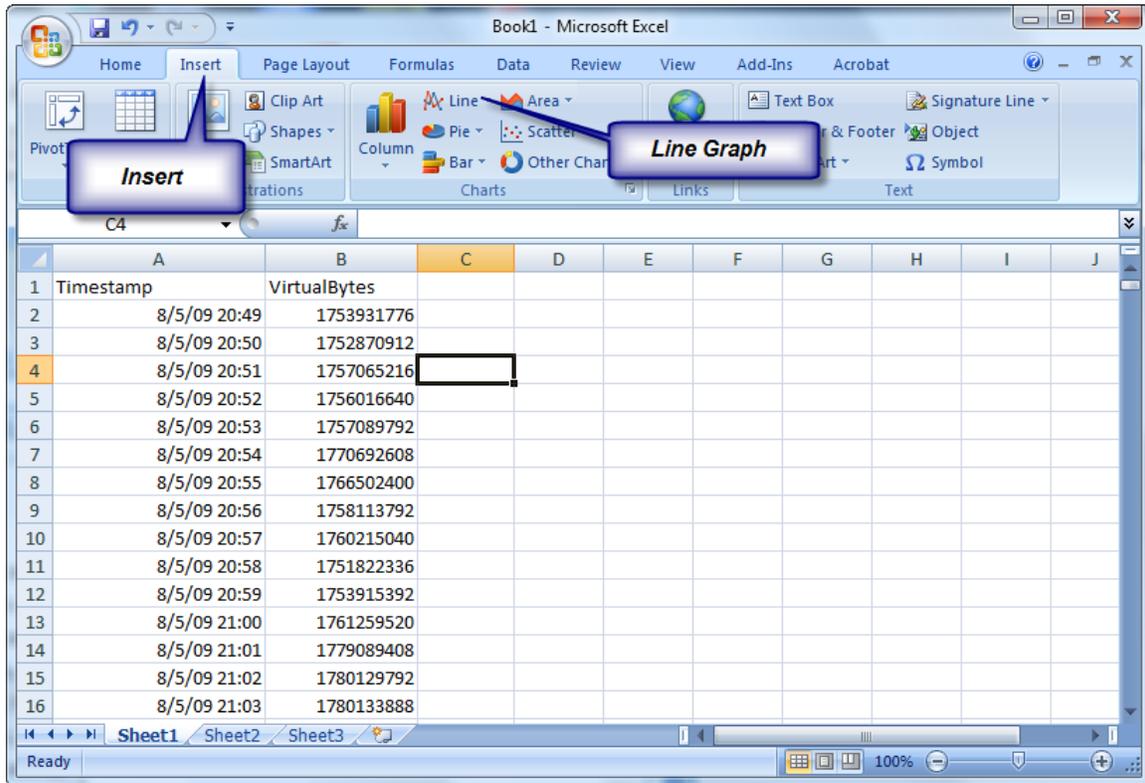

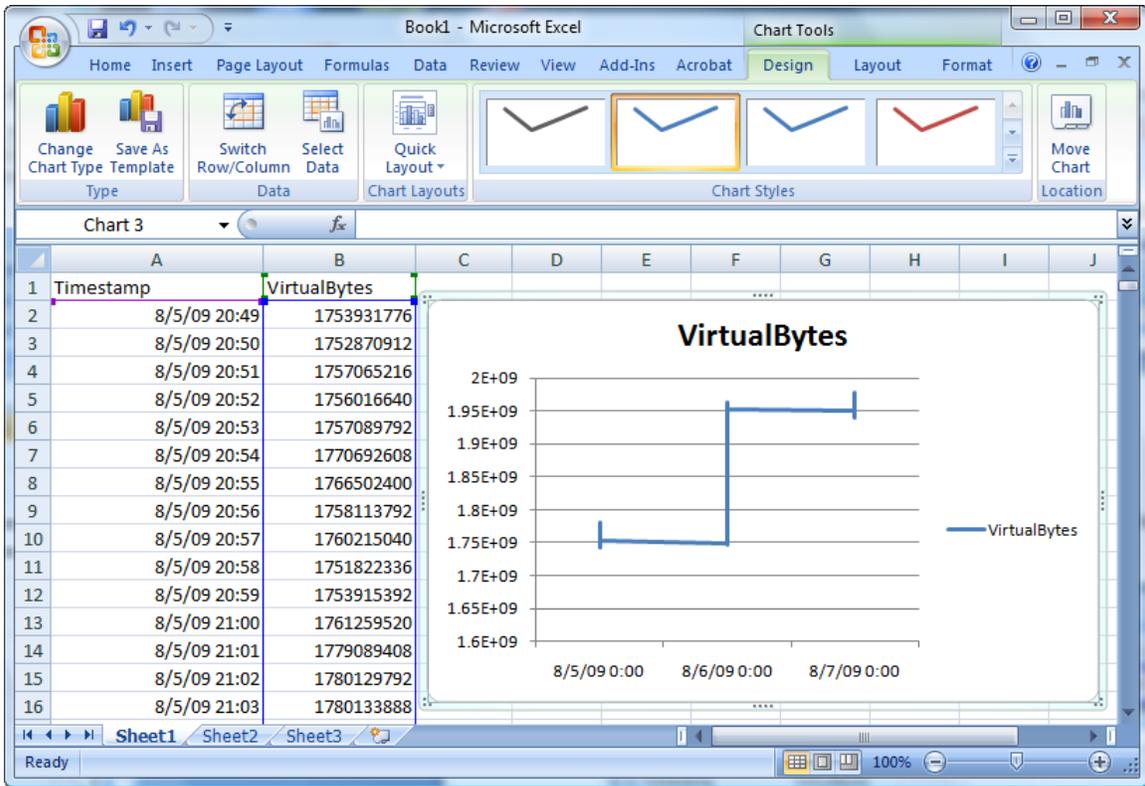Reformat the data as a time field, then click OK.

Now, we just need to add a graph to make it easier to see the changes when they occur. Select the Insert menu, then insert a new Line graph using the first 2D graph option:


Move the graph over so that you can see it better, and you'll get something like this:

Another problem that we have is the two axes do not allow for good analysis of the data. In fact, if we just look at this, it looks like the memory just jumps all at once. This is a problem with the automated graphing tool, so let's fix it.

First, right-click on the vertical axis and select Format Axis... to get to the next dialog box.
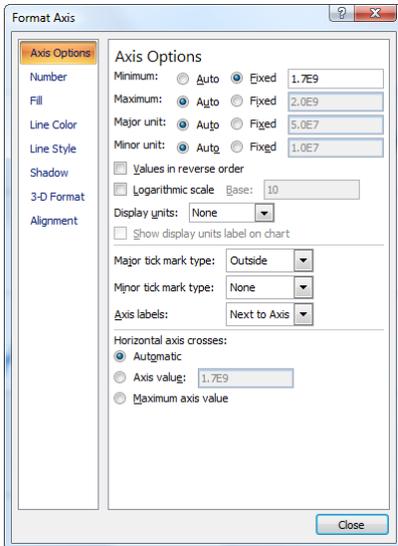
In this case, we've manually modified the minimum value to "1.7E9", or about 1.7GB, which will give us a better graph picture. Next, do the same with the horizontal axis across the bottom of the graph.
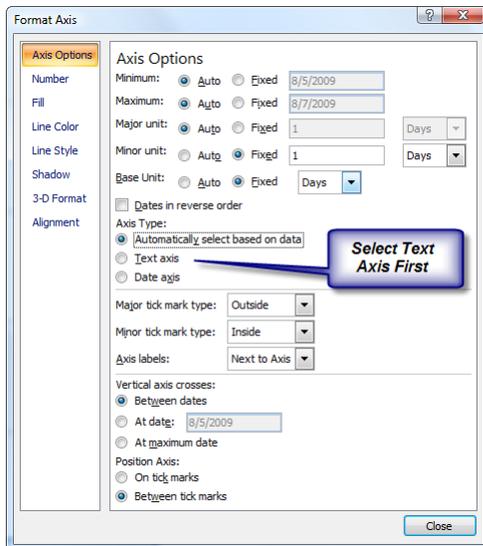
For the date axis, we need to get rid of the automated formating, so first click on the Text axis option radio button. This changes the dialog box considerably.
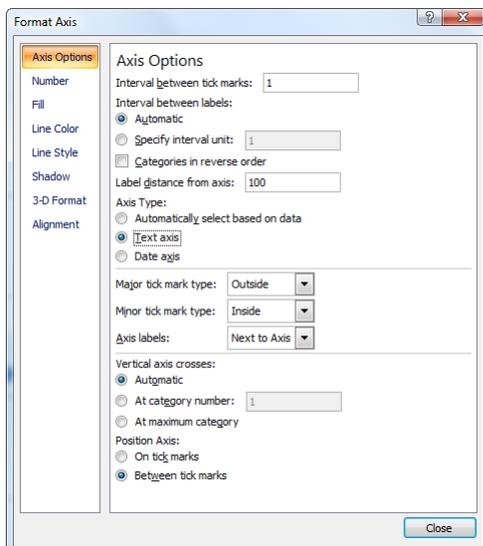
Now, that THOSE changes are out of the way, the entire graph will have changed to show a LOT more detail. You can expand the size of the window and then the size of the graph to get a much better picture of the changes to the Virtual Bytes.
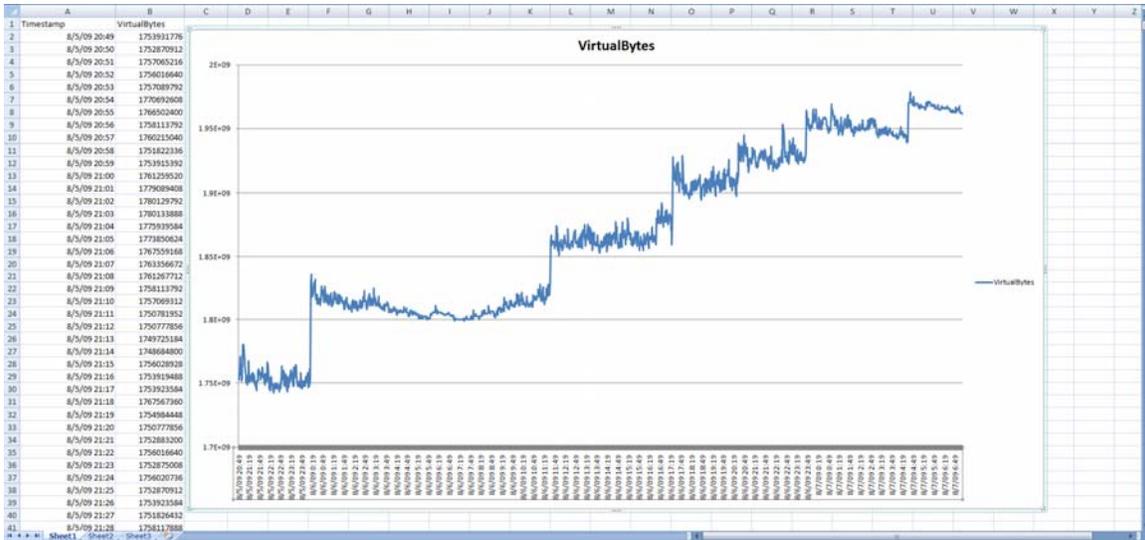
What we really see now is a lot more interesting. We can see several points in time where the Virtual Bytes value jumps up quickly, and it never seems to go back down. By looking for these points in the raw data, we can then ascertain the exact time the memory was used up, and then try to determine what was going on during that time within the engine. This information can be then used to help pinpoint the processes and/or queries which might be leaking memory in the process.