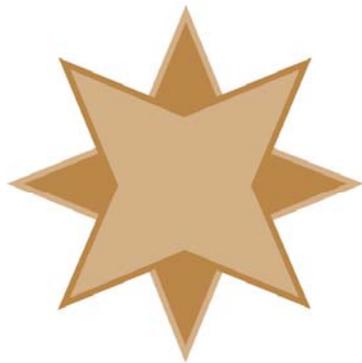


Capturing Data with Process Monitor

A White Paper From



**GOLDSTAR
SOFTWARE**

www.GoldstarSoftware.com

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

Capturing Data with Process Monitor

Last Updated: 04/15/2021

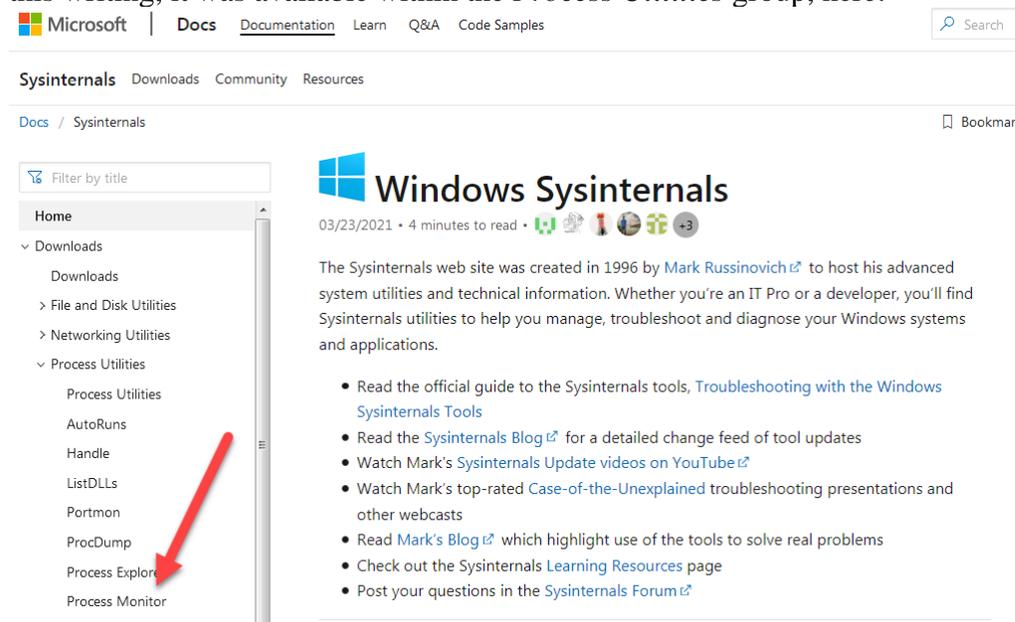
Once in a while, strange things will occur on Windows servers and workstations. In many of these cases, simply rebooting can do wonders to resolve the issue, but in other cases, recent changes in the OS, security, or perhaps even a corrupted file can create some strange results which seem to defy logic. When such a problem hits your system, it can simply be maddening that something that used to work for years is now broken, especially when the error message is not quite helpful or may be confusing.

Luckily, there is a way to peer deep inside of your Windows environment to see exactly what is going on inside – kind of like an X-ray for your computer (but less damaging than hitting your server with an X-ray burst, of course). The solution here is to use a tool from SysInternals called **Process Monitor**.

This paper will describe how to run Process Monitor (ProcMon) to capture data, how to save off the data for future analysis, and how to look at the raw data to troubleshoot problems.

Downloading Process Monitor

The first step is to get Process Monitor onto the computer. While SysInternals used to be a separate group, they are now fully owned by Microsoft. Luckily, though, Microsoft kept the old domain name as a direct portal to this suite of tools, so point your browser to www.sysinternals.com and find the download for Process Monitor there. At the time of this writing, it was available within the *Process Utilities* group, here:



The screenshot shows the Sysinternals website interface. At the top, there is a Microsoft logo and navigation links for Docs, Documentation, Learn, Q&A, and Code Samples. Below this is a search bar. The main content area features the Sysinternals logo and a date of 03/23/2021. A sidebar on the left contains a navigation menu with categories like Downloads, File and Disk Utilities, Networking Utilities, and Process Utilities. Under Process Utilities, the items listed are Process Utilities, AutoRuns, Handle, ListDLLs, Portmon, ProcDump, Process Explorer, and Process Monitor. A red arrow points to the Process Monitor link. The main content area contains an introductory paragraph and a list of links to guides, blogs, and videos.

While you are there, consider downloading the entire SysInternals Suite – there are a lot of useful tools hiding here, and some digging around here can locate the gems just waiting to be discovered.

Information Provided By **Goldstar Software Inc.**

<http://www.goldstarsoftware.com>

Capturing Events

Once you have ProcMon downloaded, go ahead & run it. You will see a screen similar to the one shown here:



I have indicated 2 important buttons in the tool bar here.

- The **magnifying glass icon** at the left side is used to start and stop the data capture. Most Windows environments are VERY busy, so as soon as you launch this tool you will want to stop capturing data until you are ready. (I really wish that it defaulted to off.) Note that sometimes it takes a few seconds for the data capture to stop, because the tool is backlogged in processing the requests.
- The **eraser icon** is used to erase the contents of the capture window. Click this button to erase the existing capture and get ready to start a new one.

One more thing that I like to do is to use the Advanced Output display. Before starting your data capture, go to the **Filter** menu and set a checkmark by the **Enable Advanced Output** menu option. This will give you a bit more detail in certain areas, which can be helpful to see what is happening.

You are now ready to capture data. To start the capture, click the magnifying glass icon, and then click it again to make it stop.

The number of captured events will be shown in the lower left corner of the screen. If you are randomly checking the data within a server, then you may wish to grab a few hundred thousand events at a time, and then stop it. If you are trying to capture an error for a specific process, then you'll want to let it run while you duplicate the error and then stop it as soon as the error message pops up.

Remember that the more data you are capturing, the harder it will be to dig through all the data to find what you want, so keep it short and sweet!

Showing and Filtering Captured Events

Now that you have data captured, you'll want to filter the raw data so that you are looking at ONLY those events that are important to you.

The first thing to do is to show only the relevant activity, which is done with the five rightmost icons. Of these, the two most common ones are Registry Events and File System Events, which are controlled by these two buttons:

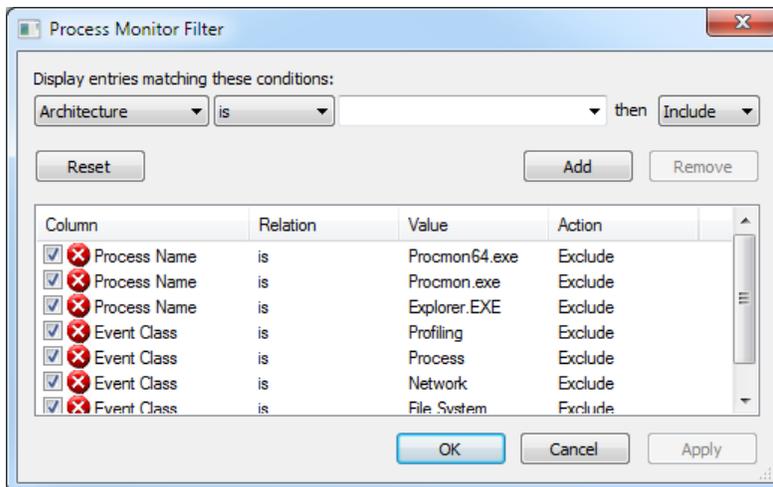
Information Provided By **Goldstar Software Inc.**

<http://www.goldstarsoftware.com>



If you are looking for an issue related to registry accesses, then enable **ONLY** the *Show Registry Activity* button. If you are looking for file access issues, use the second *Show File System Activity* button. The other three buttons are for Network Activity, Process and Thread Activity, and Profiling Events, respectively. Again, simplicity will save you time here, so you should always disable any events that you don't care about first.

While you are looking at the right events, the next step is to enable the filtering to examine only those events that are important to you. There are several ways to do this, of course. You should start with the Filter dialog box, which can be found using the **filter icon** on the tool bar, selecting *Filter...* from the *Filter* menu, or just by pressing Ctrl-L.



This dialog lets you see the currently-applied filters, quickly exclude them (using the checkmark on the left), modify a filter by clicking on one of the rows and changing the data in the top section, or remove filters entirely by clicking on one of the rows and hitting the *Remove* button.

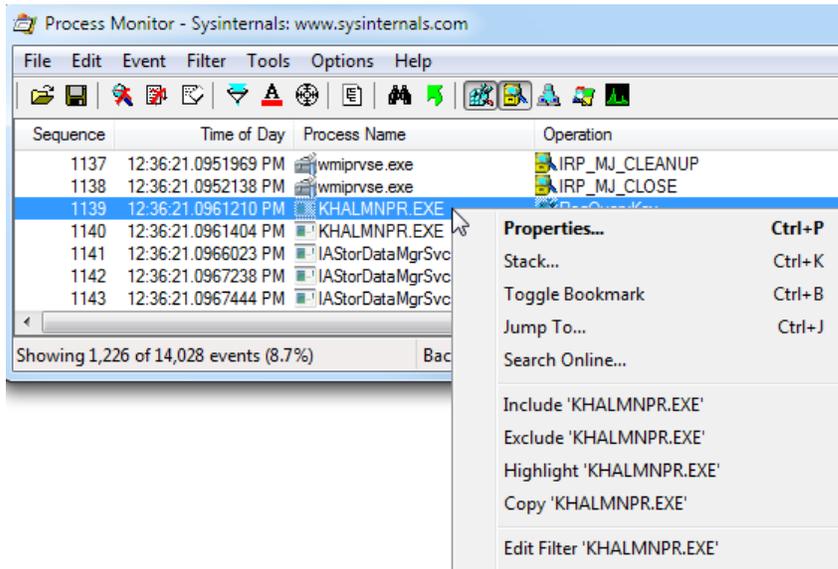
When you need to add more filters, the drop-down boxes in the top section will allow you to manually specify your filter from the full range of conditions and match types (i.e. *is* (equal to), *begins with*, *ends with*, and many more). The “then” part of the filter can be set to either *Exclude* or *Include*. Exclude filters eliminate that data from the list. Include filters are a bit odd – they actually exclude ALL BUT the matching items.

Luckily, there is an easier way to create filters. If you capture data for a short time, you can right-click on any line in the filter and set up a filter automatically. The filter will be

Information Provided By **Goldstar Software Inc.**

<http://www.goldstarsoftware.com>

built depending on the exact column in which you right-click, too. Here is an example of right-clicking on a Process Name entry in the list:



Here you can see the quick-filter options to *Include* or *Exclude* this process directly with a second click on the appropriate entry.

While we are here, the *Stack...* selection can be enlightening, as it shows you which calls are on the process stack (and an indication of whether the call is in User or Kernel space). Even better, the *Jump To...* selection will take you to the path (off-screen to the right in this view). If that path is a registry key, then RegEdit will be opened for you in the correct location. If that path is a file location, then Explorer will be opened for you in that location. This is a great time-saver, to be sure!

Of course, if you see a process with a strange name, don't flip over to your browser and start typing the name – just select the *Search Online...* option, and a browser will open up with a web search window, with the process name already in the search bar! Sweet!

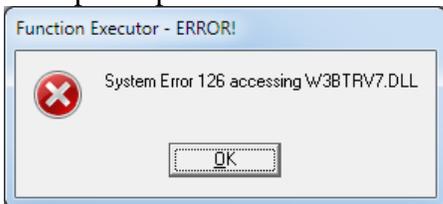
Showing Other Columns

In some cases, you may need to modify the columns being displayed. To do this, right-click the column header row and click on *Select Columns...*. Here, you can remove extraneous columns, or even add new columns. One column I recommend adding is the "Duration" column, which will show you how long each request took to complete. This can be especially helpful when looking for disk latency issues.

Using Process Monitor To Find a Missing DLL

Now that we have covered the basics of actually using Process Monitor, let's run an application that is failing due to a missing DLL and find out which file is actually missing.

Setup: If you would like to perform this exact same test on your own system, then go to the C:\Program Files(x86)\Actian\Zen\Bin folder and rename the W3BTRV7.DLL file to something different. Then, launch the **Function Executor** (WBEXEC32.EXE) and attempt to open a file. You will see the following dialog box:



This dialog box tells us which file is missing, of course, but we can use ProcMon to determine WHERE the operating system looked for the file, just in case the file exists, but the path was wrong.

After you've set this up and duplicated the issue, you're ready to capture data with ProcMon. Again, you want to limit the amount of data actually captured, so launch the WBEXEC32 application and open the File/Open dialog box BEFORE turning on the data capture. As soon as you start tracing, try to open the file (which generates the missing DLL error) and immediately stop the trace.

Since we are looking for a DLL issue, we know that we want to Show File System Events only, so make sure that the registry and other events are all turned off. Even better, we are troubleshooting a single, known application, so we can also go into the Filter dialog and set up an Include filter on the process WBEXE32.exe. (If you are lucky, you can even find a line for this process, right-click it, and select Include WBEXEC32.exe and save some time, too.)

When we are done with this, we will have a display like the following:

Sequence	Process Name	Operation	Path	Result	Detail
0	WBEXEC32.exe	IRP_MJ_CREATE	D:\BigFile	SUCCESS	Desired Access: Read Data/List D...
1	WBEXEC32.exe	IRP_MJ_DIRECTORY_CONTROL	D:\BigFile\BIGFILE.MKD	SUCCESS	Type: QueryDirectory, FileInformati...
2	WBEXEC32.exe	IRP_MJ_CLEANUP	D:\BigFile	SUCCESS	
3	WBEXEC32.exe	IRP_MJ_CLOSE	D:\BigFile	SUCCESS	
4	WBEXEC32.exe	IRP_MJ_CLOSE	D:\BigFile	SUCCESS	
5	WBEXEC32.exe	FASTIO_NETWORK_QUERY_OPEN	C:\Program Files (x86)\Actian\Zen\bin\W3BTRV7.DLL	FAST IO DISALLOW...	
6	WBEXEC32.exe	IRP_MJ_CREATE	C:\Program Files (x86)\Actian\Zen\bin\W3BTRV7.DLL	NAME NOT FOUND	Desired Access: Read Attributes, D...
7	WBEXEC32.exe	FASTIO_NETWORK_QUERY_OPEN	C:\Windows\SysWOW64\W3BTRV7.DLL	FAST IO DISALLOW...	
8	WBEXEC32.exe	IRP_MJ_CREATE	C:\Windows\SysWOW64\W3BTRV7.DLL	NAME NOT FOUND	Desired Access: Read Attributes, D...
9	WBEXEC32.exe	FASTIO_NETWORK_QUERY_OPEN	C:\Windows\system\W3BTRV7.DLL	FAST IO DISALLOW...	
10	WBEXEC32.exe	IRP_MJ_CREATE	C:\Windows\system\W3BTRV7.DLL	NAME NOT FOUND	Desired Access: Read Attributes, D...
11	WBEXEC32.exe	FASTIO_NETWORK_QUERY_OPEN	C:\Windows\W3BTRV7.DLL	FAST IO DISALLOW...	
12	WBEXEC32.exe	IRP_MJ_CREATE	C:\Windows\W3BTRV7.DLL	NAME NOT FOUND	Desired Access: Read Attributes, D...
13	WBEXEC32.exe	FASTIO_NETWORK_QUERY_OPEN	C:\Users\Bill.EMPIRE\W3BTRV7.DLL	FAST IO DISALLOW...	
14	WBEXEC32.exe	IRP_MJ_CREATE	C:\Users\Bill.EMPIRE\W3BTRV7.DLL	NAME NOT FOUND	Desired Access: Read Attributes, D...
15	WBEXEC32.exe	FASTIO_NETWORK_QUERY_OPEN	C:\Program Files (x86)\Actian\Zen\bin\W3BTRV7.DLL	FAST IO DISALLOW...	
16	WBEXEC32.exe	IRP_MJ_CREATE	C:\Program Files (x86)\Actian\Zen\bin\W3BTRV7.DLL	NAME NOT FOUND	Desired Access: Read Attributes, D...

Here, we see some accesses for the database file that was being opened, but then we see the process searching for the file W3BTRV7.DLL in multiple folders, and each time the file is not found. In fact, on my own system, this process generated event numbers 5 through 132 to search for this file – always failing. After that last request, the application simply stopped responding (because it was waiting for the user to click on the dialog box), so this is the end of the trace data. Now we know which file was missing and where it looked, and we can figure out how to fix it (by renaming the file back, of course).

Using Process Monitor To See Registry Accesses

We are now going to use the same tool to open a file, but this time capture the registry accesses, to get a feeling for which settings are used by this environment.

Setup: If you would like to perform this exact same test on your own system, then launch the **Function Executor** (WBEXEC32.EXE) and attempt to open any database file.

As with the first test, we will enable a filter on the Process Name field for WBEXEC32.exe, but this time we will select the Registry Events only. Next, scroll down through the data until you see this section:

Sequence	Process Name	Operation	Path	Result	Detail
445	WBEXEC32.exe	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
446	WBEXEC32.exe	RegOpenKey	HKLM\SOFTWARE\Action\Zen\InstallInfo\Server	SUCCESS	Desired Access: Read
447	WBEXEC32.exe	RegSetInfoKey	HKLM\SOFTWARE\Action\Zen\InstallInfo\Server	SUCCESS	KeySetInformationClass: KeySetHandleT
448	WBEXEC32.exe	RegQueryValue	HKLM\SOFTWARE\Action\Zen\InstallInfo\Server\InstallDir	SUCCESS	Type: REG_SZ, Length: 70, Data: C:\Pn
449	WBEXEC32.exe	RegCloseKey	HKLM\SOFTWARE\Action\Zen\InstallInfo\Server	SUCCESS	
450	WBEXEC32.exe	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
451	WBEXEC32.exe	RegOpenKey	HKLM\SOFTWARE\Action\Zen\Btrieve Requester	SUCCESS	Desired Access: Read
452	WBEXEC32.exe	RegSetInfoKey	HKLM\SOFTWARE\Action\Zen\Btrieve Requester	SUCCESS	KeySetInformationClass: KeySetHandleT
453	WBEXEC32.exe	RegQueryValue	HKLM\SOFTWARE\Action\Zen\Btrieve Requester\Splash Screen	SUCCESS	Type: REG_SZ, Length: 6, Data: no
454	WBEXEC32.exe	RegCloseKey	HKLM\SOFTWARE\Action\Zen\Btrieve Requester	SUCCESS	
455	WBEXEC32.exe	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
456	WBEXEC32.exe	RegOpenKey	HKLM\SOFTWARE\Action\Zen\Btrieve Requester	SUCCESS	Desired Access: Read
457	WBEXEC32.exe	RegSetInfoKey	HKLM\SOFTWARE\Action\Zen\Btrieve Requester	SUCCESS	KeySetInformationClass: KeySetHandleT
458	WBEXEC32.exe	RegQueryValue	HKLM\SOFTWARE\Action\Zen\Btrieve Requester\Embedded Spaces	SUCCESS	Type: REG_SZ, Length: 8, Data: yes
459	WBEXEC32.exe	RegCloseKey	HKLM\SOFTWARE\Action\Zen\Btrieve Requester	SUCCESS	
460	WBEXEC32.exe	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
461	WBEXEC32.exe	RegOpenKey	HKLM\SOFTWARE\Action\Zen\Btrieve Requester	SUCCESS	Desired Access: Read
462	WBEXEC32.exe	RegSetInfoKey	HKLM\SOFTWARE\Action\Zen\Btrieve Requester	SUCCESS	KeySetInformationClass: KeySetHandleT
463	WBEXEC32.exe	RegQueryValue	HKLM\SOFTWARE\Action\Zen\Btrieve Requester\Verify Key Length	SUCCESS	Type: REG_SZ, Length: 8, Data: yes
464	WBEXEC32.exe	RegCloseKey	HKLM\SOFTWARE\Action\Zen\Btrieve Requester	SUCCESS	
465	WBEXEC32.exe	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
466	WBEXEC32.exe	RegOpenKey	HKLM\SOFTWARE\Action\Zen\Microkernel Router	SUCCESS	Desired Access: Read
467	WBEXEC32.exe	RegSetInfoKey	HKLM\SOFTWARE\Action\Zen\Microkernel Router	SUCCESS	KeySetInformationClass: KeySetHandleT
468	WBEXEC32.exe	RegQueryValue	HKLM\SOFTWARE\Action\Zen\Microkernel Router\Local	SUCCESS	Type: REG_SZ, Length: 8, Data: yes
469	WBEXEC32.exe	RegCloseKey	HKLM\SOFTWARE\Action\Zen\Microkernel Router	SUCCESS	
470	WBEXEC32.exe	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
471	WBEXEC32.exe	RegOpenKey	HKLM\SOFTWARE\Action\Zen\Microkernel Router	SUCCESS	Desired Access: Read
472	WBEXEC32.exe	RegSetInfoKey	HKLM\SOFTWARE\Action\Zen\Microkernel Router	SUCCESS	KeySetInformationClass: KeySetHandleT
473	WBEXEC32.exe	RegQueryValue	HKLM\SOFTWARE\Action\Zen\Microkernel Router\Requester	SUCCESS	Type: REG_SZ, Length: 8, Data: yes
474	WBEXEC32.exe	RegCloseKey	HKLM\SOFTWARE\Action\Zen\Microkernel Router	SUCCESS	

Here, we see the Zen Client attempting to access several different registry settings in a row. This is because the Client needs to know certain things about the configuration. Even better, for successful registry lookups, the value from the registry is also displayed in the far right, so we can even answer each question as we go:

- Where is the Client installed? (C:\Program Files (x86)\Action\Zen\)
- Should it display a Splash Screen? (No)
- Are embedded spaces allowed in file names? (Yes)
- Should key lengths be checked? (Yes)
- Should a local engine be used? (Yes)
- Should a requester be used to find a remote engine? (Yes)

There will, of course, be many more lines, but this tool will at least give you an idea of what is happening under the covers of that pretty Windows front end.

Using Process Monitor To See Disk Access Times

In this final sample, we are going to use a database application to open a file in order to see the difference between a “physical open” and a “logical open”. A physical open is used when the data file is currently closed by the database, so the database engine will

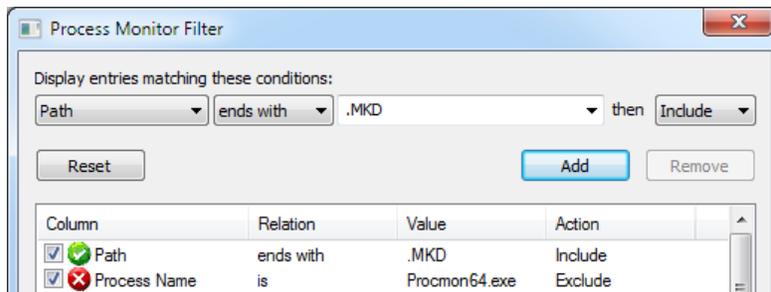
Information Provided By **Goldstar Software Inc.**

<http://www.goldstarsoftware.com>

need to retrieve certain details from the physical disk to see if it matches what it has in cache for the file or not. A logical open occurs when a file is *already* opened by the database engine, and so the engine is simply opening a second handle to the file. This does not require any actual disk reads, but a server engine will still need to check the access control list (ACL) to verify that the user has rights to the file in question.

Setup: If you would like to perform this exact same test on your own system, then launch the **Function Executor** (WBEXEC32.EXE). Start the File/Open dialog and open up a copy of the PERSON.MKD file from the DEMODATA database. For Actian Zen v14, this will be located in C:\ProgramData\Actian\Zen\Demodata\Person.mkd. Once that file is open, start up a second File/Open dialog box and open the same file a second time.

In this case, we're going to be looking at File System Events, so configure ProcMon accordingly with all other events disabled. This time, though, we are NOT going to track events from WBEXEC32! This is because the WBEXEC32 process is actually a client process, which will be talking with the local database engine, and the ENGINE will be actually doing the disk events to open the file. Luckily, we do know which file we want, so we can still add a filter to make life easy, so set up a filter where the Path ends with ".MKD" to get this:



Now, run the two File/Open commands while capturing data, and you should see something similar to the following screen:

Sequence	Process Name	Operation	Path	Result	Detail	Duration
0	WBEXEC32.exe	IRP_MJ_DIRECTORY_CONTROL	C:\ProgramData\Actian\Zen\Demodata\Person.MKD	SUCCESS	Type: QueryDirectory, FileInformationClass: FileBoth...	0.0000277
1	zenengsvc.exe	IRP_MJ_CREATE	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS	Desired Access: Generic Read/Write, Disposition: ...	0.0001293
2	zenengsvc.exe	IRP_MJ_QUERY_INFORMATION	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS	Type: QueryStandardInformationFile, AllocationSize:...	0.0000053
3	zenengsvc.exe	IRP_MJ_SET_INFORMATION	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS	Type: SetPositionInformationFile, Position: 1,077,248	0.0000047
4	zenengsvc.exe	IRP_MJ_READ	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS	Offset: 0, Length: 512, I/O Flags: Non-cached, Prio...	0.0003792
5	zenengsvc.exe	IRP_MJ_READ	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS	Offset: 0, Length: 16,384, I/O Flags: Non-cached, ...	0.0003495
6	zenengsvc.exe	IRP_MJ_QUERY_INFORMATION	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS	Type: QueryStandardInformationFile, AllocationSize:...	0.0000061
7	zenengsvc.exe	IRP_MJ_QUERY_INFORMATION	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS	Type: QueryStandardInformationFile, AllocationSize:...	0.0000028
8	zenengsvc.exe	IRP_MJ_CREATE	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS	Desired Access: Read Control, Disposition: Open, ...	0.0000113
9	zenengsvc.exe	IRP_MJ_QUERY_SECURITY	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	BUFFER OVERI	Information: Owner, Group, DACL	0.0000036
10	zenengsvc.exe	IRP_MJ_CLEANUP	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS		0.0000067
11	zenengsvc.exe	IRP_MJ_CLOSE	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS		0.0000036
12	zenengsvc.exe	IRP_MJ_CREATE	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS	Desired Access: Read Control, Disposition: Open, ...	0.0000114
13	zenengsvc.exe	IRP_MJ_QUERY_SECURITY	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS	Information: Owner, Group, DACL	0.0000027
14	zenengsvc.exe	IRP_MJ_CLEANUP	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS		0.0000069
15	zenengsvc.exe	IRP_MJ_CLOSE	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS		0.0000036
16	WBEXEC32.exe	IRP_MJ_DIRECTORY_CONTROL	C:\ProgramData\Actian\Zen\Demodata\Person.MKD	SUCCESS	Type: QueryDirectory, FileInformationClass: FileBoth...	0.0000183
17	zenengsvc.exe	IRP_MJ_CREATE	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS	Desired Access: Read Control, Disposition: Open, ...	0.0000133
18	zenengsvc.exe	IRP_MJ_QUERY_SECURITY	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	BUFFER OVERI	Information: Owner, Group, DACL	0.0000039
19	zenengsvc.exe	IRP_MJ_CLEANUP	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS		0.0000125
20	zenengsvc.exe	IRP_MJ_CLOSE	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS		0.0000097
21	zenengsvc.exe	IRP_MJ_CREATE	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS	Desired Access: Read Control, Disposition: Open, ...	0.0000139
22	zenengsvc.exe	IRP_MJ_QUERY_SECURITY	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS	Information: Owner, Group, DACL	0.0000050
23	zenengsvc.exe	IRP_MJ_CLEANUP	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS		0.0000203
24	zenengsvc.exe	IRP_MJ_CLOSE	C:\ProgramData\Actian\Zen\Demodata\PERSON.MKD	SUCCESS		0.0000072

I've highlighted a few items worth noting here:

1. Here we see the Zen Engine Service attempt to “CREATE” the file PERSON.MKD. Note that this is not a true “create file” operation, but rather a request to create a file handle at the OS layer.
2. Here we see two requests to read from the physical file, the first requesting the first 512 bytes of the file (which is needed to know the file version and page size) and the second requesting the first 16K, or two pages of the file (to load the File Control Record and verify if the data in cache is identical to that on disk). If you look at the far right column, you can add up these duration times to see that these requests took a total of just over 728us (microseconds) on the local SSD volume. (Try this on a VM/SAN to see just how much slower virtual disks really are!)
3. After confirming that the data in memory is the same, the database engine must then check the ACL to ensure that this user has rights, so we see separate requests for this data. Note how fast this is, though – this data is often cached by the OS.
4. With the second request (the logical file open), the database engine already has the file open, so we do not need to re-read any data physically from the disk, and we can simply create a new file handle without physical reads.
5. And finally we check the ACL for the second open, which is still fast.

What Else Can I Do With This?

The next time you’re bored, do some random ProcMon captures on your server or key workstations, and see what you can learn. You may find strange processes hammering the registry, applications reading or writing data on disk one byte at a time, actual disk access times to your SSD’s and HDD’s, and many other fun and exciting tidbits!

If you still have other questions, contact the technical wizards at Goldstar Software (www.goldstarsoftware.com/contact.asp) and let our professionals work with you – the optimizations you are able to make may just save you some time!