Understanding Process Monitor

Version 3

Created by Will Coffey on Sep 20, 2016 3:12 PM. Last modified by David Holland on Nov 13, 2016 1:23 PM.

Overview

Process Monitor (procmon) is an advanced monitoring/logging utility that provides visibility into the who, what, when, where and how behind the events executed on the Windows OS. It's free and provided by Windows Sysinternals. With insight into what low-level operation a process is performing, the user privilege the operation is being executed under, when it occurred, how long it took and the result of the event, you'll find root causing a difficult issue much easier.
A procmon capture will record real-time file system, registry, process/thread activity and minimal network operations. This tool requires administrative rights, including the Load and Unload Device Drivers privilege (assigned via Local Security Policy or Group Policy). This document will outline some of the features offered as well as a few tips and tricks I've found to be useful.

Toolbar Options and Shortcuts

Toolbar Options and Shortcuts	
Save (ctrl + s)	Filter (ctrl + l) (ctrl + r) l = load r = reset
Open (ctrl + o)	Highlight (ctrl + h)
Capture (ctrl + e)	Show Process Tree (ctrl + t)
Autoscroll (ctrl + a)	Find (ctrl + f)
Clear (ctrl + x)	Jump to Object (ctrl + j)

Events Classes

Registry – This could be creating keys, reading them, deleting them, or querying them. You'll be surprised just how often this happens.

File System – This could be file creation, writing, deleting, etc, and it can be for both local hard drives and network drives.

Network – This will show the source and destination of TCP/UDP traffic. It doesn't show the data traffic.

Process and Thread – These are events for processes and threads where a process is started, a thread starts or exits, etc. This can be useful information in certain instances, but is often something you'd want to look at in Process Explorer instead.

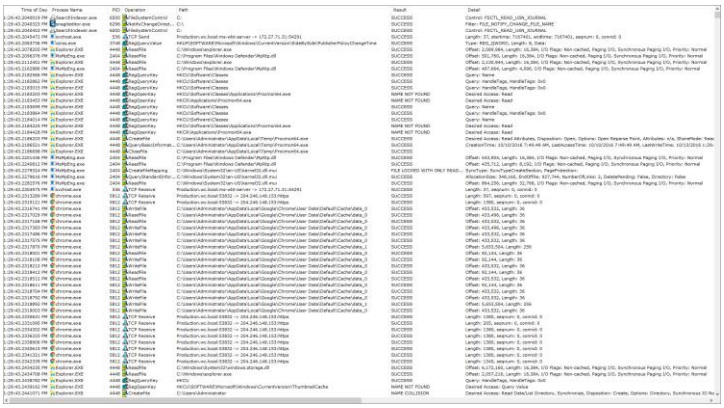
Profiling – These events are captured by Process Monitor to check the amount of processor time used by each process, and the memory use. Again, you would probably want to use Process Explorer for tracking these things most of the time, but it's useful here if you need it.

Capturing Events

In order to analyze the actions of an event, you must first capture those events while procmon is running. To access.download procmon.exe, navigate to the following location:

SysInternalsTools

Procmon does not have to be downloaded to capture events, however, I recommend downloading it. Once you have access to promon, launch the application on the device to view the operations being performed. Here's an example of how a procmon trace looks:

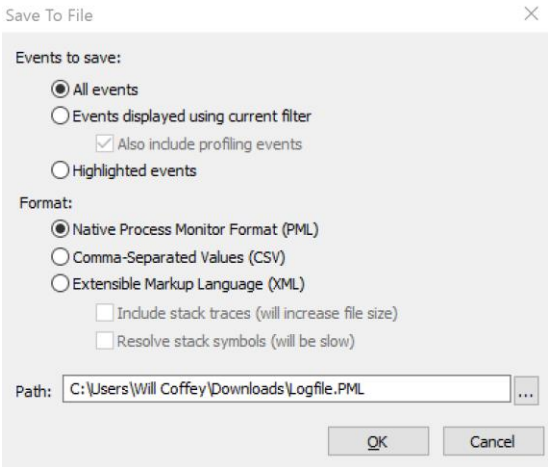


As you can see, there's a lot of data collected and at first glance can be a bit overwhelming. Later we'll go over options we have to isolate the captured events.

Saving Captured Events

To save a capture you can select **File | Save** from the menu bar or use the **ctrl+s** keyboard shortcut.

The default trace file will be in a Native Process Monitor (PML) format.



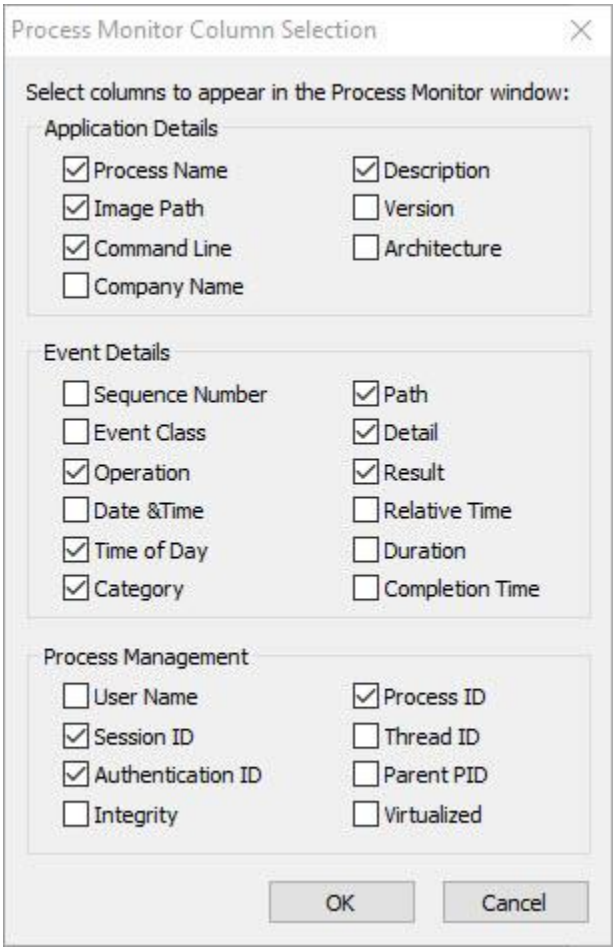
Customizing the Captured Events

The data captured is customizable in the sense of column order and font type. To adjust the order in which the columns appear simply drag them to your desired location. The below listed columns are included by default.

Columns:

- **Time of Day –** this column is fairly self-explanatory; it shows the exact time that an event occurred
- **Process Name –** the name of the process performing the operation along with an icon identifying the event class
- **PID -** the process ID of the process that generated the event.
- **Operation -** this is the name of the operation that is being logged, and there is an icon that matches up with one of the event types (registry, file, network, process).
- **Path** – this is not the path of the process; it is the path to whatever was being worked on by this event. For instance, if there was a WriteFile event, this field will show the name of the file or folder being touched. If this was a registry event, it would show the full key being accessed.
- **Result -** This shows the result of the operation, which codes like SUCCESS or ACCESS DENIED, NAME NOT FOUND, END OF FILE, BUFFER OVERFLOW etc.
  Buffer Over Flow: This occurs when a program copies more data into a memory buffer than the program was designed to accommodate. When looking at the Windows NTSTATUS result code, Status_Buffer_Overflow “The data was too large to fit into the specified buffer."
  Don’t confuse this with the malicious buffer_overflow in the sense of exploiting computer security. When contained in a procmon trace think of this result as “Buffer Too Small”.
- **Detail -** additional information related to the operation of the event.

The below image (Process Monitor Column Selection) outlines all of the available column options. To include the additional column options, right-click on any column header and ***select columns***. You will then be presented with the following interface allowing you to enable more items:



Application Details – static info determined at process startup, this info won't change

Event Details – dynamic info specific to the event

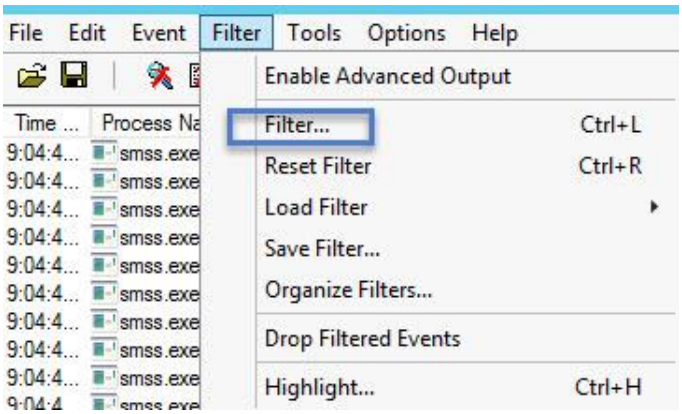
Process Management – RunTime info about the process

Filtering and Highlighting

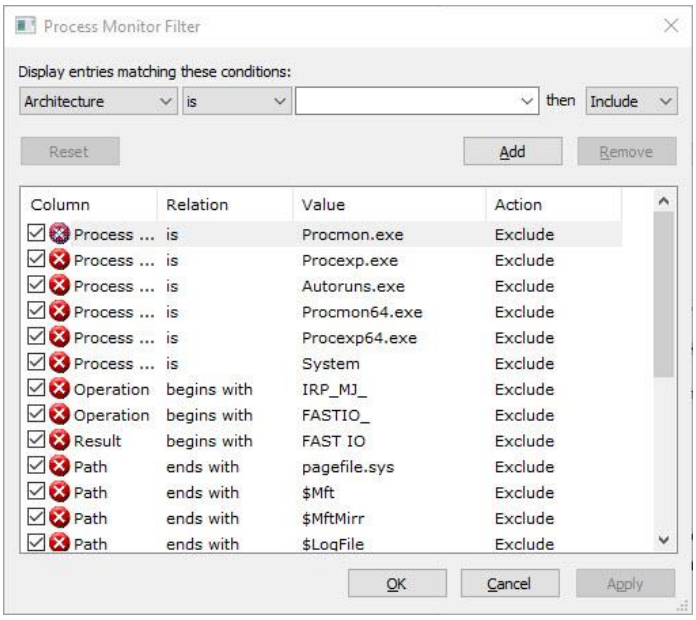
When performing a capture, a ton of events will be loaded in a relatively short period of time. In efforts to isolate the events and only show items you deem important, filtering and highlighting options can be specified. Filtering these events do not drop them from the capture, it simply removes them from the display. Highlighting works the same as filtering from a configuration standpoint but also adds a visual distinction to the selected events.

Filtering

To view the filter options you can select **Filter | Filter** from the menu or use the **ctrl-l** keyboard shortcut:



There are (4) configurable sections under ***Display entries matching these conditions***:



- The first field includes all of the available **columns**, the first entry in this list is *Architecture*. There are a total of (27) columns to choose from. Keep in mind that you can filter by these columns even if they are not enabled in the display.

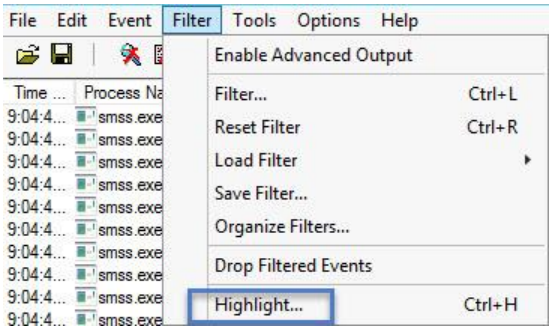
- The second field has a list of **expressions** you can choose from. This list includes the following options:
  - Is
  - Is not
  - Less than
  - More than
  - Begins with
  - Ends with
  - Contains
  - Excludes

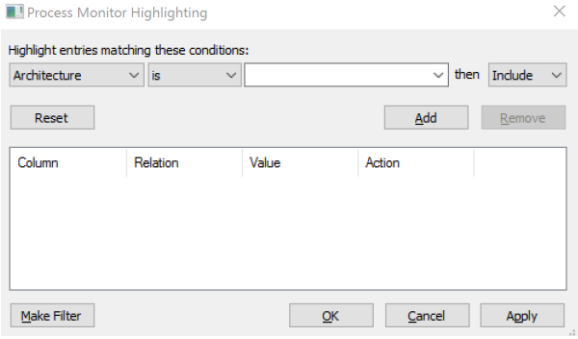
- The third field contains a “drop down” of objects for you to choose from. These are dependent on the column type you specified in the first field. In some cases, the drop down list will be blank. In this case you will need to type in the condition you wish to filter off of.

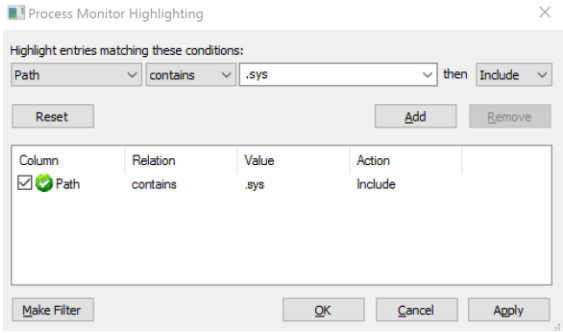
- The fourth field allows an Include/Exclude option for the conditions you configured. The specified condition has to be **included** if you want the filter to apply.

Highlighting

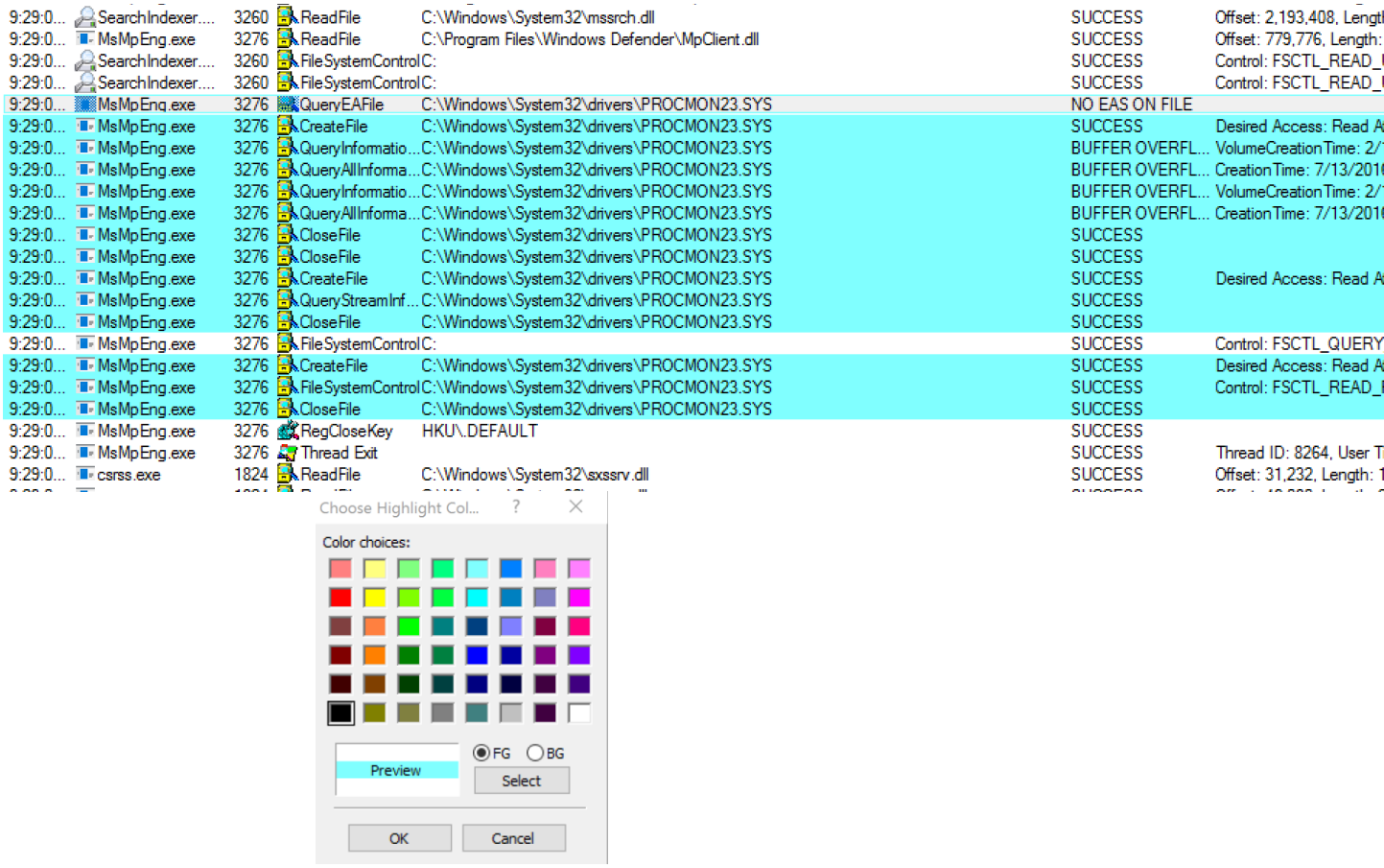
To view the highlight option you can select **Filter | Highlight** from the menu or use the **ctrl-h** keyboard shortcut:







By default, the highlight color is bright blue, this can be modified by navigating to the menu and selecting **Options | Highlight Colors**.



Common Result Codes

The below listed table outlines the known results and their descriptions. Most of them are self-explanatory but this will hopefully bring understanding to all of the results.

Result Code	Description
Success	The operation succeeded
Access Denied	The operation failed due to insufficient permissions from the requester.
Sharing Violation	The operation failed because the object is already opened and doesn't allow sharing mode
Name Collision	An attempt to create an object that already exist.
Name Not Found / Path Not Found / No Such File	An attempt was made to open an object that doesn't exist was made. Routinely DLL files are compiled to search recursively or for specific directories.
Name Invalid	A request was attempted for an object with an invalid name
No More Entries / No More Files	The caller has finished enumerating the contents of a folder or registry key
End of File	The caller has read to the end of a file
Buffer Too Small	Essentially the same as Buffer Overflow
Re-parse	The caller has requested an object that links to another object. Ex: HKLM\System\CurrentControlSet might redirect to HKLM\System\ControlSet001
Not Re-parse Point	The requested object does not link to another object
FAST IO Disallowed	Indicates a low-level optimized mechanism is not available for the requested file system object.
File Locked with Only Readers	Indicates that a file or file mapping was locked and that all users of the file can only read from it.
File Locked with Writers	Indicates that a file or file mapping was locked and that at least one user of the file can write to it.
IS Directory	The requested object is a file system folder
Invalid Device Request	The specified request is not a valid operation for the target device
Invalid Parameter	An invalid parameter was passed to a service or function
Not Granted	A requested file lock cannot be granted because of other existing locks.
Cancelled	An I/O request was cancelled – ex: the monitoring of a file system folder for changes
Bad Network Path	The network pat cannot be located.
Bad Network Name	The specified share name cannot be found on the remote server
Media Write Protected	The disk cannot be written to because it's write protected
Key Deleted	Illegal operation attempted on a registry key that has been marked for deletion.
Not Implemented	The requested operation is not implemented

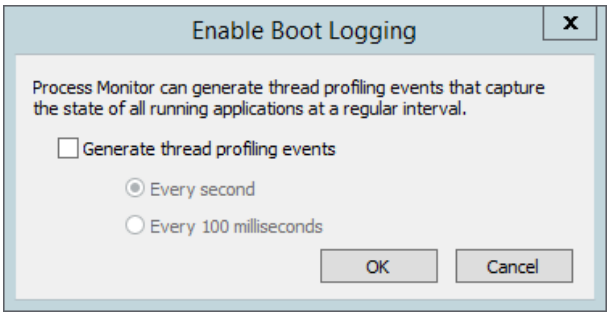
Boot Logging

Procmon is capable of motoring system activity when no one is logged on and after users have logged off. You are also able to capture events occurring during system shutdown. The following activity can be captured before, during or when no user has logged into the device:

- Boot-start device drivers
- Auto-start services
- Logon sequence

Boot logging does not run during safe mode. If the system crashes early in the boot process, logging can be deactivated by selecting the Last Known Good option from the Windows boot menu. To access this option press F8 during startup.

To enable Boot logging select **Options | Enable Boot Logging** can from the menu and select **OK**. Selecting Generate thread profiling events will provide more data to the boot log regarding the state of the applications running at during the capture.

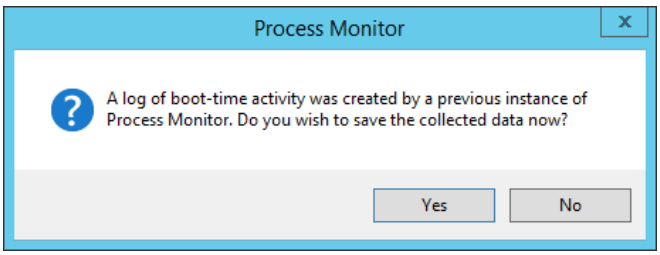


When boot logging is enabled it's only for the next boot process. You must explicitly enable boot logging for each subsequent boot process. Once enabled, logging will continue until you launch procmon again and the captured activity will be contained in a PMB file in the following location:

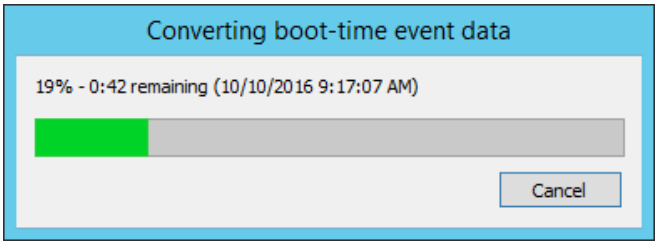
```
%windir%\Procmon.pmb
```

Boot logging should only be enabled for troubleshooting purposes. The procmon.pmb file will continue to capture data until procmon.exe is re-ran.

Once boot logging is enabled you will be ready to capture events occurring during the boot process. After you log off and log back on to the device or restart the device and log back in, open procmon. An auto-detection will sense boot logging was enabled and the below listed prompt will appear:



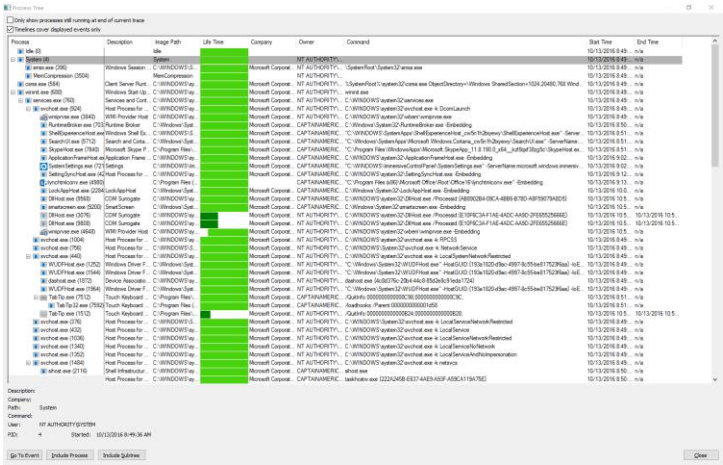
After selecting **Yes**, the boot time data will be converted into a PML file you can filter and use for troubleshooting.



Tips and Tricks

In order to effectively troubleshoot issues it's very beneficial to understand what the sequence of events look like when things are successful. Knowing the expected behavior and what process are involved as well as the processing order, will allow you to pinpoint the failure quicker. Sometime you may not want to focus too much on the error message you are being presented. As we all have come to realize, the root issue may be stemming from another underlying process or dll file. To have insight on the processes included and the order of operations during the life of a job you can capture the activity and review the process tree. This will show you the parent-child relationship of the running processes.

To view the process tree select **Tools | Process Tree** from the menu bar or use the **ctrl+t** keyboard shortcut.



Another option I've found extremely helpful is the ability to filter by event classes.

 **Registry**  **File System Activity**  **Network Activity**  **Process and Thread**

The event classes have been covered above and toggling these options allows you to immediately filter out non-pertinent data.

 **Include Process from Windows**. This is also referred to as a Bulls-eye. This feature allows you to drag and drop the Bulls-eye onto any application, automatically including it as a filtered event.

Troubleshooting with Process Monitor Video

The purpose of this video is to outline how you can use filtering in process monitor to isolate events for a more focused troubleshooting approach.

For additional information regarding uses for Procmon please reference the following articles: