

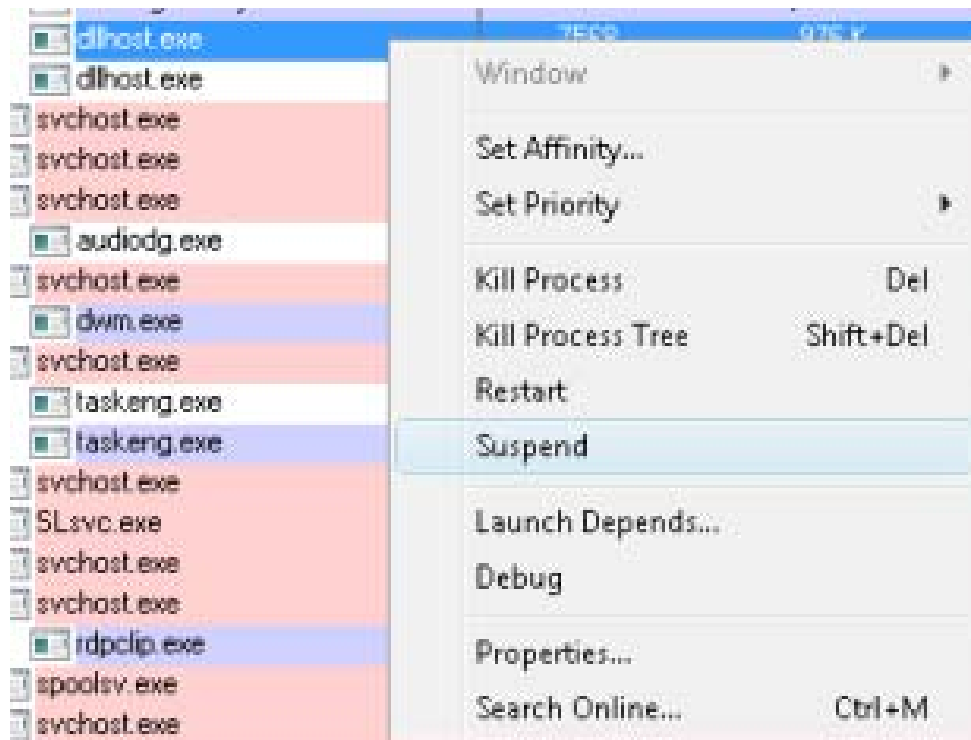
# The Case of the Sloooooow System

Mark Russinovich  
(From Mark Russinovich Blog)

A few weeks ago my wife complained that her Vista desktop was not responding to her typing or mouse clicks. Given the importance of the customer, I immediately sat down at the system to troubleshoot. It wasn't completely hung, but extremely sluggish. For example, the mouse moved and when I clicked on the start button the start menu opened after about 30 seconds. I suspected that something was hogging the CPU and likely could have resolved the problem simply by logging off or rebooting, but knew that if I didn't determine the root cause and address it, she'd likely be calling on my technical support services again in the near future. In any case, stooping to that kind of troubleshooting hack is beneath my dignity. I therefore set out to investigate.

My first step was to run Process Explorer to see which process was using the CPU. After a few minutes Process Explorer finally appeared and showed that not one, but two processes were involved, each consuming 50% of the CPU: Iexplore.exe and Dllhost.exe. Iexplore is Internet Explorer (IE) and I suspected that IE itself wasn't the problem, but that it was a browser helper object (BHO), ActiveX control, or some other plugin loaded into IE. Similarly, Dllhost.exe is the host process for out-of-process COM server DLLs, so it was probably not at fault, but the COM server loaded into it. Both required digging deeper and I decided to tackle IE first.

In order to try and get some CPU headroom in which to operate, I suspended the Dllhost process by selecting it in Process Explorer, right-clicking to open the process context menu, and selecting the Suspend entry:

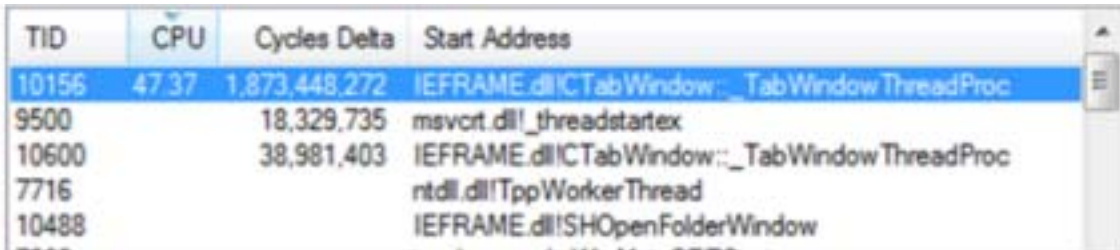


That put the Dllhost process to sleep and, as I expected, that freed up 50% of the CPU. That's because the computer was a dual-core system and so to consume 100% of the available CPU cycles a process would have to have two threads, each hogging one of the cores. Most bugs I've seen that result in the CPU being pegged are caused by a single thread.

# The Case of the Slooooo System

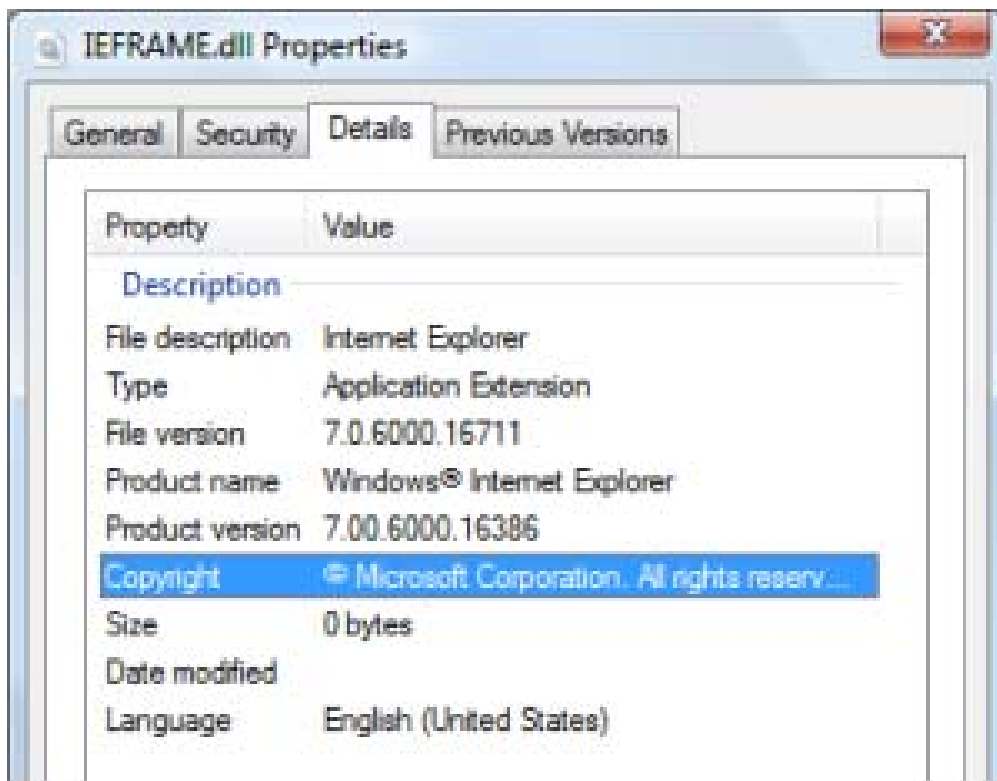
Mark Russinovich  
(From Mark Russinovich Blog)

Processes don't execute code, threads do, so I needed to look inside the IE process to see what thread or threads were running. I double-clicked on `ieframe.dll` in Process Explorer to open its process properties dialog and switched to the Threads page. Several threads were running, but one was dominating the CPU:



| TID   | CPU   | Cycles Delta  | Start Address                                 |
|-------|-------|---------------|---|
| 10156 | 47.37 | 1,873,448,272 | IEFRAMES.dll!CTabWindow::_TabWindowThreadProc |
| 9500  |       | 18,329,735    | msvcrt.dll!_threadstartex                     |
| 10600 |       | 38,981,403    | IEFRAMES.dll!CTabWindow::_TabWindowThreadProc |
| 7716  |       |               | ntdll.dll!TppWorkerThread                     |
| 10488 |       |               | IEFRAMES.dll!SHOpenFolderWindow               |

From past experience I knew that `ieframe.dll` was part of IE, but to be sure I clicked on the modules button on the Threads tab of the Properties dialog and switched to the Details page of the resulting Shell properties dialog:



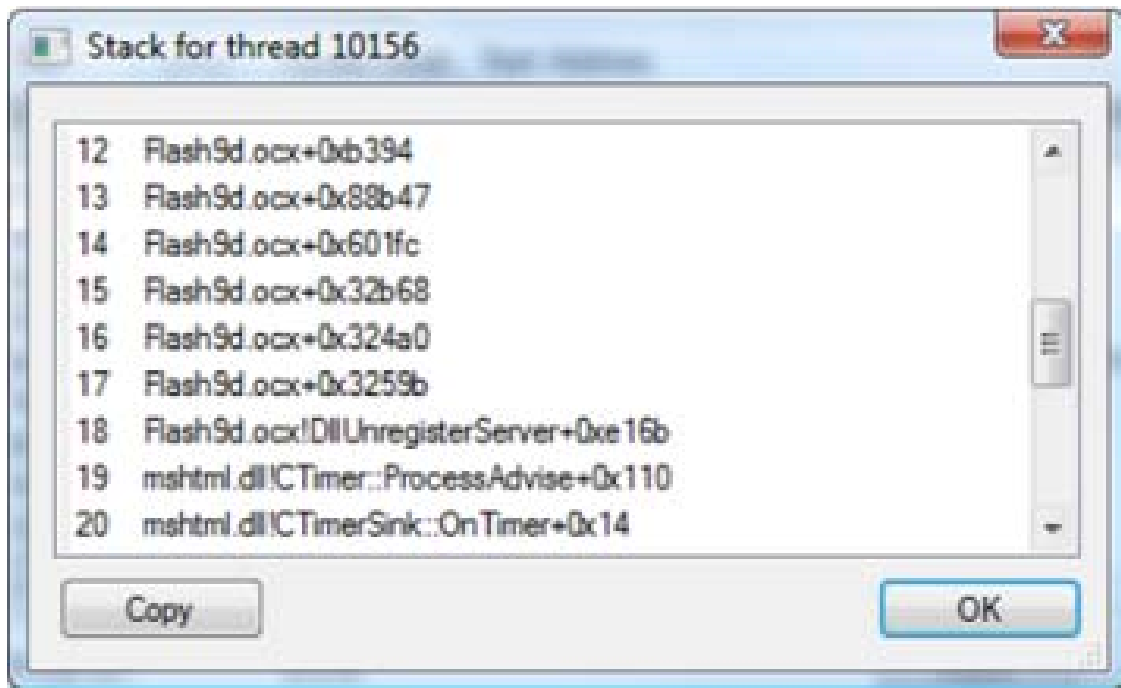
The description didn't give me a clue as the thread's specific purpose, so I moved to the second clue about the thread, its start function. Because I had configured Process Explorer to retrieve symbols for Windows images from the Microsoft symbol server in Options->Configure Symbols, Process Explorer showed the name of the function where each thread began executing. Sometimes the DLL or function where a thread starts executing is enough to identify the thread's purpose or the software causing a problem. In this case, the thread began in a function named `CTabWindow::_TabWindowThreadProc`.

## The Case of the Sloooooow System

Mark Russinovich  
(From Mark Russinovich Blog)

The function name hints that it's the one in which the main thread of a tab starts running, but that still wasn't enough to tell me why the thread was running so much; I needed to dig even deeper and look inside the thread to see where it was executing.

To look at what the thread was up to, I double-clicked on it in the Threads list to open the Thread Stack dialog, which shows the functions on the thread's stack. A stack is essentially an execution history, where each function listed called the one above it on the list and the function at the top of the list is the one most recently executed by the thread at the time of Process Explorer looks at the stack. I scrolled through the list, looking for frames that referenced 3rd-party DLLs or Microsoft IE plugins, since they would be far more likely to have a bug than IE's own code. Sure enough, I found frames pointing at a popular 3rd-party ActiveX control, Adobe Flash:



Just to be sure that I hadn't happened to catch Flash running when a different component was using most of the CPU time, I closed and reopened the stack dialog several times, but all of them pointed at Flash.

The first thing I do when I suspect that some software is causing a problem is to check the vendor's web site to make sure that I have the latest version. I opened the Process Explorer DLL view and looked at Flash.ocx's version, went to Adobe's site and looked at the version of the current Flash download, and they were the same.

I was at a dead end. I couldn't know for sure if Flash had a bug or, more likely, there was a Flash application that had a bug, nor could I be sure that the problem wouldn't recur. I tried to determine which site was hosting the Flash content by closing tabs one by one, but when I had close them all the thread was still running.

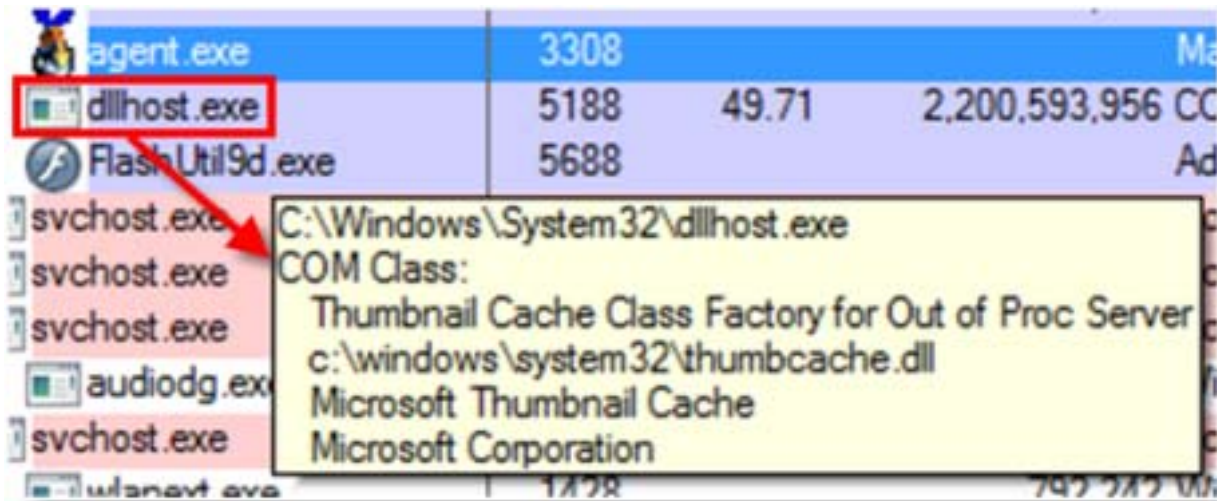
At this point the only options I had were to uninstall Flash and leave my wife with a degraded web experience, or terminate IE to stop the current CPU usage and hope that it wouldn't happen again. I

# The Case of the Sloooooow System

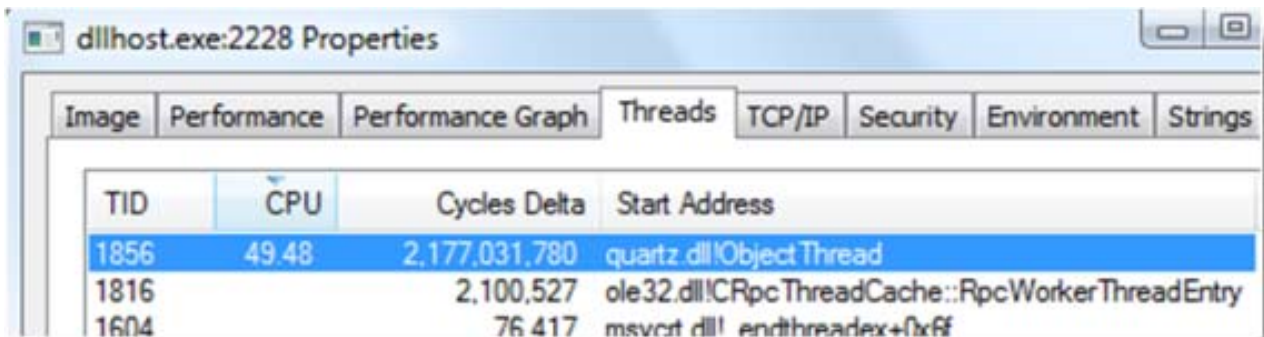
Mark Russinovich  
(From Mark Russinovich Blog)

chose the latter and the case remains open. Since investigating this I've seen the same Flash behavior again on my wife's system and on my own, so have been vigilantly watching the Adobe site for a new version just in case its due to a bug in Flash itself. I was disappointed that there was no actionable result of the investigation, but at least I knew what had caused the CPU usage.

I now turned my attention the Dllhost problem with the hope that I'd meet with better success. Process Explorer lists in a tooltip the component or components loaded into hosting processes like Svchost.exe (the Windows service host process), Rundll32 (the Control Panel applet hosting process), Taskeng.exe (the scheduled task hosting process on Vista and Server 2008), and Dllhost.exe. I moved the mouse over Dllhost.exe to see what COM server it was running:



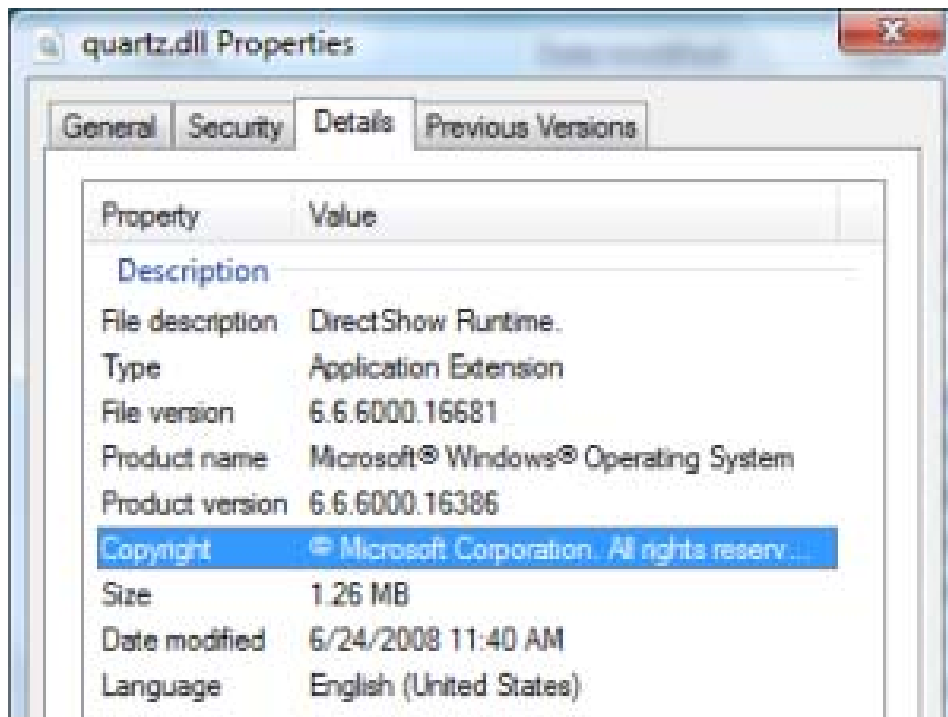
It was running the Thumbnail Cache COM server, whose job it is to create Explorer thumbnails for image and media files. It is part of Windows, so once again I had to look inside the process for more clues. I resumed the Dllhost process I had suspended earlier and opened the process properties threads page:



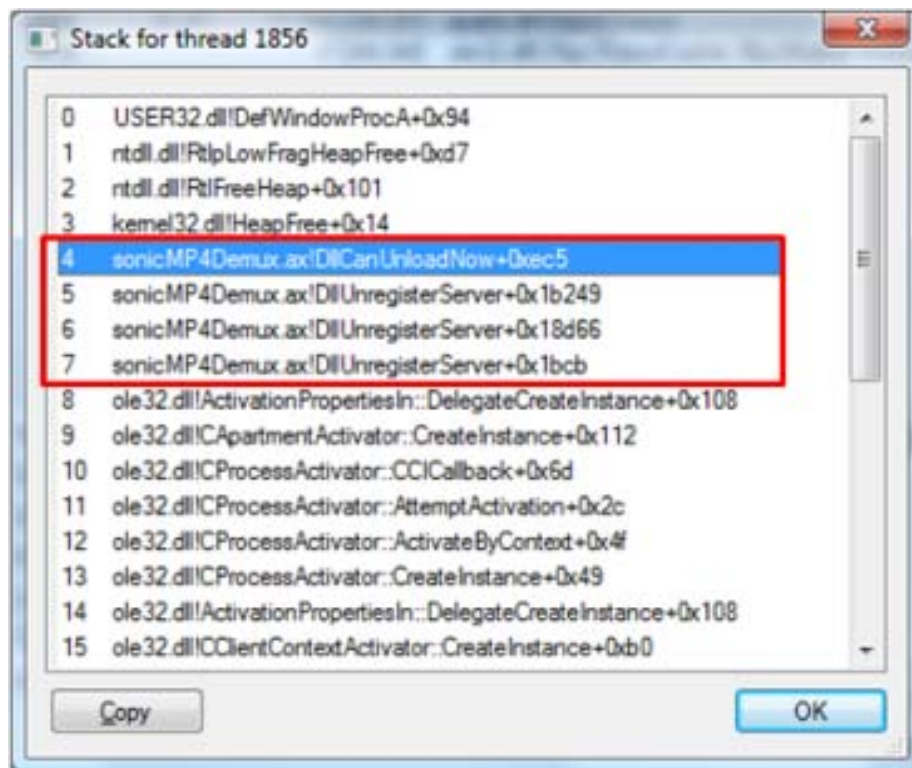
The thread consuming the most CPU in this case started in Quartz.dll's ObjectThread function. I looked at its properties and saw that it was another Windows DLL, the DirectShow Runtime, with a generic function name:

# The Case of the Slooooo System

Mark Russinovich  
(From Mark Russinovich Blog)



Next, I double-clicked to look at the thread stack:

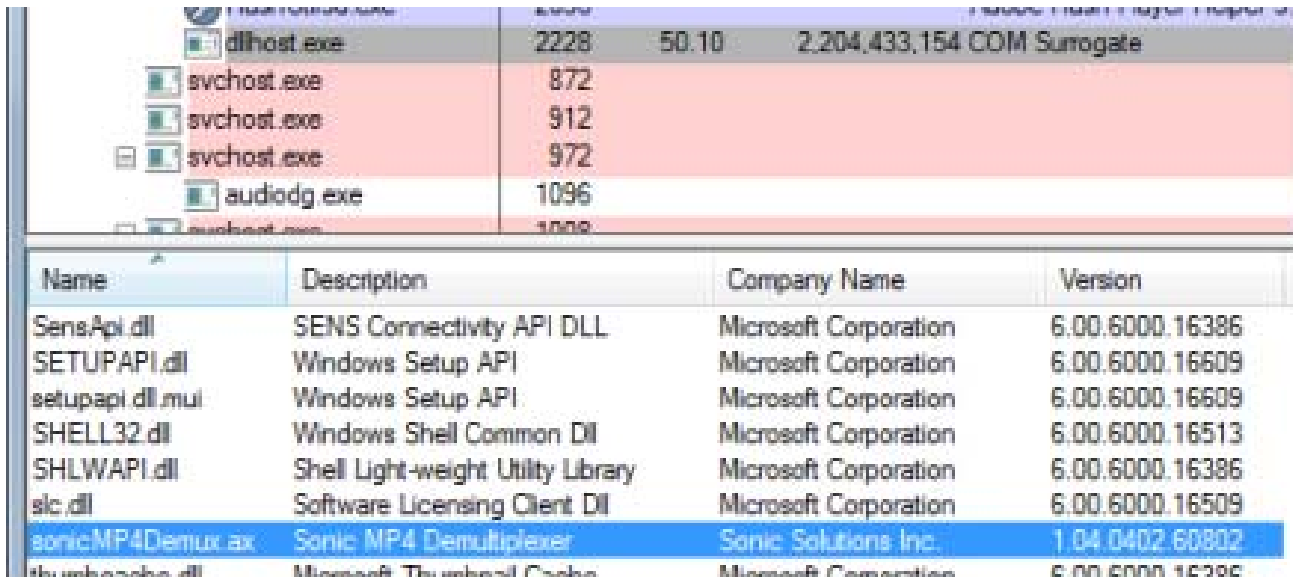


# The Case of the Sloooooow System

Mark Russinovich  
(From Mark Russinovich Blog)

The first few frames were in User32.dll and Ntdll.dll, core Windows system DLLs, but frames 4-7 are in the Sonicmp4demux.ax (".ax" is an extension commonly used for DirectShow filters), a 3rd-party component. The function names for those frames were the same and didn't make sense because the Microsoft symbol server only stores symbols for software included in Windows. Several more stack snapshots confirmed that it was the code causing the CPU usage.

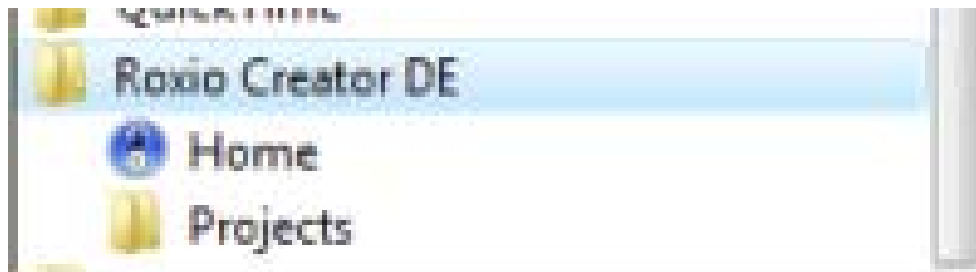
Now that I had my suspect, the next step was to check for a newer version. But first I had to figure out what software the DLL came with, which was harder than it seemed. I opened the DLL view to take a closer look at the version information, but the description didn't reveal anything:



The image shows a screenshot of Windows Task Manager. The top part shows a list of running processes with columns for Name, PID, Private Bytes, and Description. The bottom part shows a list of loaded DLLs with columns for Name, Description, Company Name, and Version.

| Name             | Description                        | Company Name          | Version         |
|------------------|------------------------------------|-----------------------|-----------------|
| SensApi.dll      | SENS Connectivity API DLL          | Microsoft Corporation | 6.00.6000.16386 |
| SETUPAPI.dll     | Windows Setup API                  | Microsoft Corporation | 6.00.6000.16609 |
| setupapi.dll.mui | Windows Setup API                  | Microsoft Corporation | 6.00.6000.16609 |
| SHELL32.dll      | Windows Shell Common DLL           | Microsoft Corporation | 6.00.6000.16513 |
| SHLWAPI.dll      | Shell Light-weight Utility Library | Microsoft Corporation | 6.00.6000.16386 |
| slc.dll          | Software Licensing Client DLL      | Microsoft Corporation | 6.00.6000.16509 |
| sonicMP4Demux.ax | Sonic MP4 Demultiplexer            | Sonic Solutions Inc.  | 1.04.0402.60802 |
| thumbcache.dll   | Microsoft Thumbnail Cache          | Microsoft Corporation | 6.00.6000.16386 |

There were no folders in the Start menu or items in the Add/Remove Programs list with Sonic in the name. I Windows-Live-searched (I expect that word to be added to Webster's any day now) for Sonic and found that it's part of the Roxio's CD and DVD authoring software suites. I looked in the start menu and sure enough, found a Roxio folder:



I ran the Roxio software to check its version number and discovered that the Creator application includes a built-in facility to check for updates. I ran it, but it came up empty:

# The Case of the Sloooooow System

Mark Russinovich  
(From Mark Russinovich Blog)



I checked the Roxio web site just to be sure and it turned out there was a newer version that the built-in updater hadn't offered, perhaps because the update, according to the page, didn't offer anything new:



I downloaded it anyway (all 640MB of it!) and waited the 15 or so minutes for it to install. Then I checked the version information of Sonicmp4demux.ax to see if it was newer, but its version number, 1.4.402.60802, was the same as the one I'd seen in the DLL view and the file was two years old:

```
Sigcheck v1.31
Copyright (C) 2004-2006 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\Program Files\Common Files\Sonic Shared\SonicMC01\sonicMP4Demux.ax:
  Verified:      Unsigned
  File date:     7:43 AM 8/9/2006
  Publisher:     Sonic Solutions Inc.
  Description:   Sonic MP4 Demultiplexer
  Product:       Sonic MP4 Demultiplexer
  Version:       1, 4, 402, 60802
  File version:  1, 4, 402, 60802
```

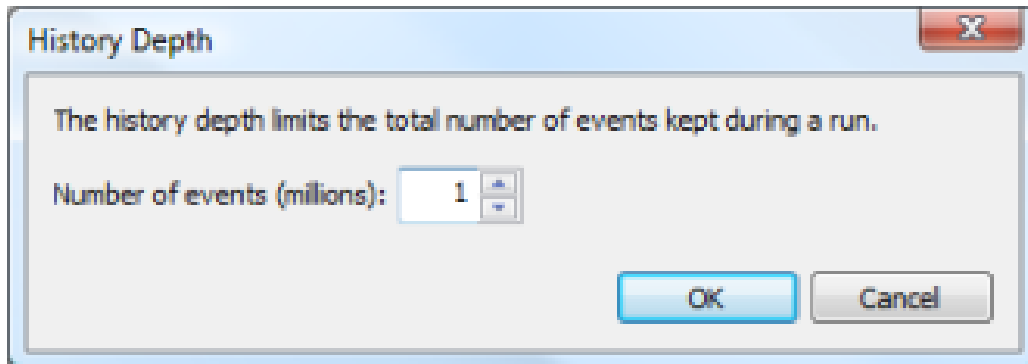
I could have uninstalled the software, which would ensure that the problem wouldn't return, but I wanted to keep Roxio for its DVD authoring functionality. I didn't care if I didn't get thumbnails for Roxio-specific image formats - I wasn't even sure there were any I'd ever see in Explorer - so I set out to see if I could disable just the Sonic demultiplexer. I could have searched the Registry for the DLL name, which is surely where it was registered, but that's a brute-force approach and if there were

# The Case of the Slooooo System

Mark Russinovich  
(From Mark Russinovich Blog)

indirect or multiple references I could easily end up disabling more than just its thumbnail generation and possibly breaking something in Windows.

Process Monitor was the perfect tool for the job. Because I didn't know when the problem might reoccur - it might takes days to reproduce - I didn't want to just run it and let it consume all available virtual memory or disk space, so I set the History Depth in the Options menu to have Process Monitor retain only the most recent 1 million events:



I also set an Include filter for paths matching C:\Windows\System32\Dllhost.exe, minimized it, and let my wife have the system back.

The next day I came home from work, sat down at the computer and saw from Process Explorer that Dllhost.exe was back at it, consuming 50% of the CPU. I suspect that because it's a dual-core system, the problem had been showing up regularly, but my wife hadn't noticed it because the remaining CPU capacity was enough to mask it (another good reason to buy multi-core processors!). I brought Process Monitor to the foreground and noted it had seen 114,000 Dllhost operations, which was obviously way too many to scan through individually. I searched for "sonicmp4" and found a reference in a Registry query near the end of the trace:

|               |   |   |                            |
|---------------|---|---|----------------------------|
| RegOpenKey    | HKCU\Software\Classes\CLSID\{A7DD2151-A645-409A-9B39-DF146D710E72}\InprocServer32 | NAME NOT FOUND  | Desired Access: Maximum Al |
| RegQueryValue | HKCR\CLSID\{A7DD2151-A645-409A-9B39-DF146D710E72}\InprocServer32\Default          | BUFFER OVERFL...  | Length: 144                |
| RegQueryValue | HKCR\CLSID\{A7DD2151-A645-409A-9B39-DF146D710E72}\InprocServer32\Default          | SUCCESS   | Type: REG_SZ, Length: 140  |
| RegQueryKey   | HKCR\CLSID\{A7DD2151-A645-409A-9B39-DF146D710E72}                                 | Type: REG_SZ  |                            |
| RegOpenKey    | HKCU\Software\Classes\CLSID\{A7DD2151-A645-409A-9B39-DF146D710E72}                | Length: 140   |                            |
| RegQueryValue | HKCR\CLSID\{A7DD2151-A645-409A-9B39-DF146D710E72}                                 | Data: C:\Program Files\Common Files\Sonic Shared\SonicMCO1\sonicMP4Demux.ax |                            |

The query is of a COM object registration for the demultiplexer. Because the COM object is a 3rd-party DLL, I was certain that that COM Class ID (CLSID) isn't hard-coded into Windows, so I went back to the first entry in the trace and searched for "A7DD215", the first few characters of the CLSID. The search found a match a few thousand operations earlier:

# The Case of the Slooooo System

Mark Russinovich  
(From Mark Russinovich Blog)

|               |  |   |
|---------------|--|---|
| Load Image    | C:\Windows\System32\devenum.dll  | SUCCESS   |
| RegQueryValue | HKCU\Software\Classes  | SUCCESS   |
| RegOpenKey    | HKCU\Software\Classes\CLSID\{083863F1-70DE-11D0-BD40-00A0C911CE86}\Instance\{A7DD2151-A645-409A-9B39-DF146D710E72} | NAME NOT FOUND  |
| RegOpenKey    | HKCR\CLSID\{083863F1-70DE-11D0-BD40-00A0C911CE86}\Instance\{A7DD2151-A645-409A-9B39-DF146D710E72}                  | SUCCESS   |
| RegQueryValue | HKCR\CLSID\{083863F1-70DE-11D0-BD40-00A0C911CE86}\Instance\{A7DD2151-A645-409A-9B39-DF146D710E72}                  | SUCCESS   |
| RegOpenKey    | HKCU\Software\Classes\CLSID\{083863F1-70DE-11D0-BD40-00A0C911CE86}\Instance\{A7DD2151-A645-409A-9B39-DF146D710E72} | NAME NOT FOUND  |
| RegQueryValue | HKCR\CLSID\{083863F1-70DE-11D0-BD40-00A0C911CE86}\Instance\{A7DD2151-A645-409A-9B39-DF146D710E72}\FriendlyName     | SUCCESS   |
| RegCloseKey   | HKCR\CLSID\{083863F1-70DE-11D0-BD40-00A0C911CE86}\Instance\{A7DD2151-A645-409A-9B39-DF146D710E72}                  | Type: REG_SZ<br>Length: 48<br>Data: Sonic MP4 Demultiplexer |
| ReadFile      | C:\Windows\System32\qedit.dll  | SUCCESS   |
| RegQueryKey   | HKCU\Software\Classes  | SUCCESS   |

The CLSID was in the name of a Registry key under another COM object registration. I Windows-Live-searched (that just rolls off the tongue, doesn't it?) for the parent CLSID and found this KB article that explains that the registry key is where DirectShow filters register: [http://msdn.microsoft.com/en-us/library/ms787560\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms787560(VS.85).aspx) I took a look at the stack for the particular query to confirm that's the reason Dllhost was reading from there:

|    |                          |  |
|----|--------------------------|--|
| 5  | ntkmlpa.exe              | NtOpenKey + 0x39                                   |
| 6  | ntkmlpa.exe              | KiFastCallEntry + 0x12a                            |
| 7  | ntdll.dll                | ZwOpenKey + 0xc                                    |
| 8  | advapi32.dll             | BaseRegOpenClassKeyFromLocation + 0xc9             |
| 9  | advapi32.dll             | BaseRegOpenClassKey + 0xa2                         |
| 10 | advapi32.dll             | LocalBaseRegOpenKey + 0xe2                         |
| 11 | advapi32.dll             | RegOpenKeyExW + 0x10f                              |
| 12 | devenum.dll              | ATL::CRegKey::Open + 0x22                          |
| 13 | devenum.dll              | CDeviceMoniker::Read + 0x140                       |
| 14 | quartz.dll               | MonGetName + 0x40                                  |
| 15 | quartz.dll               | CFilterGraph::NextFilter + 0x18f                   |
| 16 | quartz.dll               | CFilterGraph::RenderViaIntermediate + 0x256        |
| 17 | quartz.dll               | CFilterGraph::RenderRecursively + 0x37             |
| 18 | quartz.dll               | CFilterGraph::Render + 0xd6                        |
| 19 | qedit.dll                | CMediaDet::_Load + 0x1aa                           |
| 20 | qedit.dll                | CMediaDet::put_Filename + 0xd9                     |
| 21 | MediaMetadataHandler.dll | EhGetVideoThumbnail + 0x84                         |
| 22 | MediaMetadataHandler.dll | CMediaPropertyStoreFSDK::GetVideoThumbnailAsStream |
| 23 | MediaMetadataHandler.dll | CMediaPropertyStoreAVI::GetValueForMap + 0x61      |
| 24 | MediaMetadataHandler.dll | CMediaPropertyStoreAVI::GetValue + 0x55            |
| 25 | MediaMetadataHandler.dll | CMediaPropertyStoreContainer::GetValue + 0x32      |
| 26 | shell32.dll              | COplockPropertyStore::GetValue + 0x17              |
| 27 | propsys.dll              | CMultiplexPropertyStore::GetValue + 0x8f           |

I was now confident that I could simply rename the Sonic filter registration key to prevent its use. I never delete registry keys when performing this kind of troubleshooting just in case the change disables important functionality or somehow breaks something else. I had seen from the traces that the thumbnail cache generator had come across an AVI file that caused it to load the Sonic demultiplexer, a format Windows is obviously able to handle on its own, so I was pretty sure things would continue to work. After terminating the Dllhost and making the change, I browsed to the same

# The Case of the Sloooooow System

Mark Russinovich  
(From Mark Russinovich Blog)

folder, deleted the thumbnails, and confirmed that there was no reduced functionality as far as I could tell. I then used Roxio to successfully burn a DVD with a number of AVI files. This case was closed.

My wife's system was now usable again, and though I wasn't able to close the Flash-related part of the case, at least I knew the cause and could keep an eye out for updates. More importantly, by solving the Dllhost part of the case, even if Flash went crazy again, her system would still be usable and she wouldn't be filing a critical support incident for it with me - thanks to Process Explorer and Process Monitor.