

Troubleshooting Windows Application Crashes or Hangs

Bruce Mackenzie-Low

One of the most challenging issues for Windows administrators to troubleshoot is when a user application unexpectedly hangs or crashes. Due to the intermittent nature of crashes and hangs, it can be very difficult to "catch" the application misbehaving, leaving you with very few clues as to what caused the problem.

Fear not! There are a few simple tools you can use to help isolate the issue to a particular program, DLL, error, or condition that may lead you to a documented workaround or patch. This article will survey a variety of free tools, including Mark Russinovich's new ProcDump utility, which can assist you with troubleshooting applications that crash or hang so you can intelligently search the World Wide Web for a solution.

Free Tools

Everyone loves free tools, but sometimes there's still a price to pay for them on the Internet. Free tools often require you to provide an email address before you download them so you can be spammed by product offerings for years to come. They can also open the floodgates for spyware or other Trojans which can negatively impact your server. For these reasons, I rarely download non-Microsoft tools.

Fortunately, Microsoft offers a variety of free tools that can be used to troubleshoot hanging applications and outages. For years now, a tool called Dr. Watson has been available as part of the Windows operating system. When properly configured, Dr. Watson will detect applications that crash and provide a log file and user dump file for troubleshooting. Analyzing this data will often lead to a known error code or condition that has a documented workaround or hotfix. For more details on using Dr. Watson, you can refer to the Microsoft KB articles 246084 and 278689, or Drwtsn32.exe online help. You can also review my previous article on installing and using the Windows debugger, also known as Windbg.

Perhaps a little more useful than Dr. Watson is a tool called ADPlus, which can be downloaded with the Debugging Tools for Windows. ADPlus is a VBScript that can be used to monitor an application for an unexpected condition and capture a user dump file when it occurs. The tool can also be used to force a crash dump on a hung user application, allowing you to analyze the dump with the Windows debugger. Extensive documentation can be found on using ADPlus in Microsoft KB article 286350 or my tip on troubleshooting Windows print spooler outages.

If your troublesome application involves Microsoft Internet Information Services (IIS), then the tool of choice would be Microsoft's DebugDiag. This is a comprehensive tool used to identify a variety of problems including Web server hangs, slow performance, crashes and memory leaks. The tool can also be used with simple Win32 applications that don't involve IIS. You can download DebugDiag from Microsoft and there is abundant documentation on its use including Microsoft KB article 931370 and Tim Fenner's tip on using Debug Diagnostics to troubleshoot IIS issues.

Finally, there is a new tool from Microsoft called ProcDump. This utility combines many of the features found in the tools mentioned above, but also contains a very handy feature to dump a process when CPU activity spikes to a predetermined level for a specified period of time. The remainder of this article will examine ProcDump in detail, using Windbg to analyze the dump.

Using the ProcDump Utility

ProcDump is a command line tool that can be used to troubleshoot a variety of issues. Like Dr. Watson, ADPlus and DebugDiag, ProcDump can be used to capture a process memory dump when an unexpected condition or exception occurs. Also like ADPlus and DebugDiag, it can be used to force a process dump on a hung application. But unlike any of its predecessors, ProcDump can be

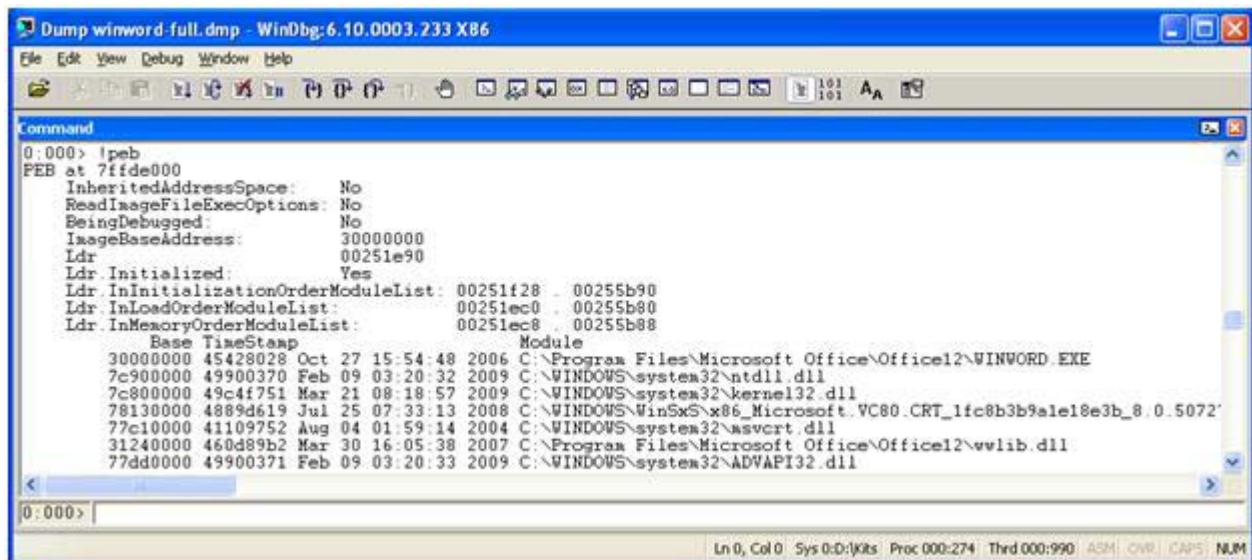
Troubleshooting Windows Application Crashes or Hangs

Bruce Mackenzie-Low

used to dump a process when its CPU activity spikes to a particular level. This can be especially useful for those intermittent performance issues where it is hard to predict when the problem will occur. A single executable (procdump.exe) is provided which accepts a number of different options. Without any options, the ProcDump tool will force a memory dump and leave the application running. For example, the following command will force a dump of the Microsoft Outlook application, capturing the memory contents in outlook.dmp:

```
C:\> procdump outlook.exe
```

By default, only thread and handle information is capture in the process memory dump. By using the option -ma, a complete process memory dump will be performed. This will allow the debugger to identify more information about the application including the thread environment (!teb), the process environment (!peb), and the locking information (!locks -v) as illustrated in the following debugger output of winword.exe:



```
Dump winword-full.dmp - WinDbg: 6.10.0003.233 X86
File Edit View Debug Window Help
Command
0:000> !peb
PEB at 7ffde000
InheritedAddressSpace: No
ReadImageFileExecOptions: No
BeingDebugged: No
ImageBaseAddress: 30000000
Ldr 00251e90
Ldr.Initialized: Yes
Ldr.InInitializationOrderModuleList: 00251f28 - 00255b90
Ldr.InLoadOrderModuleList: 00251ec0 - 00255b80
Ldr.InMemoryOrderModuleList: 00251ec8 - 00255b88
Base TimeStamp Module
30000000 45428028 Oct 27 15:54:48 2006 C:\Program Files\Microsoft Office\Office12\WINWORD EXE
7c900000 49900370 Feb 09 03:20:32 2009 C:\WINDOWS\system32\ntdll.dll
7c800000 49c4f751 Mar 21 08:18:57 2009 C:\WINDOWS\system32\kernel32.dll
78130000 4889d619 Jul 25 07:33:13 2008 C:\WINDOWS\WinSxS\x86_Microsoft_VC80_CRT_1fc8b3b9a1e18e3b_8_0_5072
77c10000 41109752 Aug 04 01:59:14 2004 C:\WINDOWS\system32\asvcrt.dll
31240000 460d89b2 Mar 30 16:05:38 2007 C:\Program Files\Microsoft Office\Office12\wvlib.dll
77dd0000 49900371 Feb 09 03:20:33 2009 C:\WINDOWS\system32\ADVAPI32.dll
0:000>
```

By using the -h option, ProcDump will detect a hung Windows application and force a memory dump. This is similar to the functionality found in ADPlus and DebugDiag. Using the -e option will cause ProcDump to detect an unhandled exception with the application and capture a process dump. By subsequently analyzing the process dump, you can determine what program, DLL and error condition were present at the time of the outage. This will allow you to intelligently search the Web for similar scenarios to determine if a known issue exists or whether you need to contact the vendor.

What sets ProcDump apart from its predecessors is the ability to detect CPU spikes and collect a process dump when they occur. This is especially useful for intermittent issues when no one is around to intervene. Three options are available to implement this functionality:

- c specifies the CPU threshold to be reached before generating a process dump
- s specifies the number of consecutive seconds the threshold must be reached
- u specifies the CPU usage should be treated as a single core

To clarify the usage, let's take a look at a simple example. The first command below will start a separate window with a directory command that traverses the system disk. You can then use the tlist command to quickly find the PID (Process ID). Finally, using the ProcDump command specifying a

Troubleshooting Windows Application Crashes or Hangs

Bruce Mackenzie-Low

CPU threshold of 10% for at least two seconds on PID 1432 (in this example) should generate a process dump:

```
C:\> start "dir window" dir /s c:\
```

```
C:\> tlist
[...]
```

1432	cmd.exe	dir window - dir /s c:\
------	---------	-------------------------

```
C:\> procdump -c 10 -s 2 1432
```

```
ProcDump v1.4 - Writes process dump files
Copyright (C) 2009 Mark Russinovich
Sysinternals - www.sysinternals.com
```

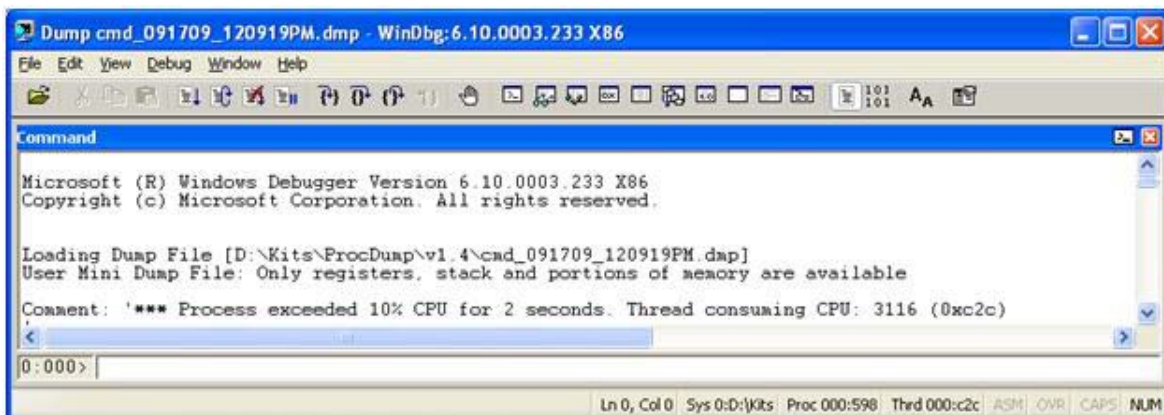
```
Process: cmd.exe (1432)
CPU threshold: 10% of system
Duration threshold: 2s
Number of dumps: 1
Hung window check: Disabled
Exception monitor: Disabled
Dump file:          D:\Kits\ProcDump\v1.4\cmd.dmp
```

Time	CPU	Duration
[12:09.18]	15%	1s
[12:09.19]	10%	2s

```
Process has hit spike threshold.
Writing dump file D:\Kits\ProcDump\v1.4\cmd_091709_120919PM.dmp...
Dump written.
```

Of course this is an unrealistic example of its usage, as one would typically specify a much higher CPU threshold for a longer period of time to avoid transient CPU spikes. But the example does provide a simple illustration of how these options work together to trigger a process dump.

Once the process dump has been written, you can use Windbg to analyze the dump. You will notice in the figure below how the debugger comments that a process has exceeded 10% CPU for two seconds and the corresponding thread ID (0xc2c). The thread ID is important because a process may have multiple threads, so you should focus on the stack pattern of the thread that caused the CPU spike.



Troubleshooting Windows Application Crashes or Hangs

Bruce Mackenzie-Low

Finally, by issuing the debugger ~*kv command, the stack pattern will be revealed for all threads, providing you with the names of the functions being executed. In the example below, you can see how the directory command executes several functions such as cmd!FileIsConsole, cmd!WriteEol, cmd!NewDisplayFile and cmd!WalkTree. While these function names may mean little to us, they can be used as keywords when searching the Web for possible solutions to our runaway process.

```
0:000> ~*kv
. 0 Id: 598.c2c Suspend: 0 Teb: 7ffde000 Unfrozen
ChildEBP RetAddr Args to Child
0012e0a0 7c90daaa 7c912de8 000007ec 0012e0e8 ntdll!KiFastSystemCallRet (FPO: [0.0.0])
0012e0a4 7c912de8 000007ec 0012e0e8 0012e0e8 ntdll!NtRequestWaitReplyPort+0xc (FPO: [3.0.0])
0012e0c4 7c81af73 0012e0e8 00000000 00020208 ntdll!CsrClientCallServer+0x8c (FPO: [4.0.4])
0012e1a4 4ad05baa 00000007 0012e1b8 00155aa0 kernel32!GetConsoleMode+0x4f (FPO: [Non-Fpo])
0012e1bc 4ad0cae2 00000001 0013eb88 0013f7dc cmd!FileIsConsole+0x54 (FPO: [1.1.4])
0012e1e8 4ad1cfbd 00155aa0 0013e844 4ad1860c cmd!WriteEol+0x56 (FPO: [1.5.4])
0012e1f4 4ad1860c 00155aa0 001de958 0013eb88 cmd!WriteFlushAndEol+0x15 (FPO: [1.0.0])
0013e844 4ad111bd 0013eb88 001de958 00155aa0 cmd!NewDisplayFile+0x2db (FPO: [4.16782.0])
0013e8ac 4ad1136f 0013eb88 00155aa0 00000006 cmd!ExpandAndApplyToFS+0x22a (FPO: [9.145.4])
0013eb4c 4ad0d0f5 0013eb88 00155aa0 00000006 cmd!WalkTree+0x40 (FPO: [10.27.0])
0013ebf4 4ad0d0f5 00000002 00155aa0 00000006 cmd!WalkTree+0x215 (FPO: [10.27.0])
0013ec9c 4ad0d0f5 00000002 00155aa0 00000006 cmd!WalkTree+0x215 (FPO: [10.27.0])
0013ed44 4ad0d0f5 00000004 00155aa0 00000006 cmd!WalkTree+0x215 (FPO: [10.27.0])
0013edec 4ad0d0f5 00000004 00155aa0 00000006 cmd!WalkTree+0x215 (FPO: [10.27.0])
0013ee94 4ad0d0f5 00000039 00155aa0 00000006 cmd!WalkTree+0x215 (FPO: [10.27.0])
0013ef3c 4ad0d420 00000004 00155aa0 00000006 cmd!WalkTree+0x215 (FPO: [10.27.0])
0013f7c4 4ad0d2c1 0013f7dc 001583d0 001537b0 cmd!PrintPatterns+0x1c4 (FPO: [1.531.0])
0013fc60 4ad0d143 001583a8 0013fe9c 4ad05aa2 cmd!Dir+0x1e1 (FPO: [1.289.4])
0013fc6c 4ad05aa2 001583d0 00000000 001583d0 cmd!eDirectory+0x10 (FPO: [1.0.0])
0013fe9c 4ad013eb 001583d0 001583d0 00000001 cmd!FindFixAndRun+0x1f5 (FPO: [1.134.0])
```

As you can see, there are several free tools available to troubleshoot application crashes and hangs. We saw how Dr. Watson, ADPlus, DebugDiag and ProcDump all provide the capability to capture a process dump. Then by using WinDbg, you can analyze the dump and review the stack pattern of the current thread for a crash scenario, or the runaway thread as identified by ProcDump.