

# Examining Xperf

Michael Morales

(Reprinted from WindowsItPro Magazine)

## Executive Summary

Event Tracing for Windows (ETW) includes Xperf (xperf.exe), a useful tool for discovering system and application process information, which you can use to troubleshoot performance issues. Explore using Xperf to diagnose two common issues: high system usage and high disk I/O problems.

Event Tracing for Windows (ETW) is a fast, built-in Windows tracing mechanism for recording activity events provided by both user-mode applications and kernel device drivers. These events aren't what you'll find in the System or Application event Logs. Rather they're component-specific activity events that let administrators and developers account for specific execution states to help diagnose workload, software, and configuration problems. ETW tracing can be enabled and disabled quickly, which makes obtaining tracing information in production environments convenient. Additionally, you can enable and disable ETW tracing without having to restart the system or process. Let's look at xperf.exe (Xperf), a tool that's part of ETW, which you can use to learn more about how your system or application works. Before we dig much further, though, we need to review architecture briefly.

## ETW and Xperf

ETW comprises four main components:

1. Event providers—components that generate activity-specific events. The OS has many built-in event providers.
2. Event controllers—programs or utilities that can enable or disable events or groups of events.
3. Consumers—can be realtime or post-processing. Post-processing consumers read information from an .etl file.
4. Event trace sessions—where buffering and logging occur. Events are buffered and written to an .etl trace file or a realtime event consumer.

The Windows Fundamentals team created Xperf, an ETW controller and consumer that's part of the Windows Performance Toolkit, which you can download via the link. Xperf is built over the ETW infrastructure in Windows and provides some valuable information to help administrators and developers understand how applications and systems operate in a production environment. I'll introduce you to basic usage of Xperf and some common scenarios where Xperf is useful in revealing how your system or application operates under the covers.

Xperf is designed primarily to work on Windows Server 2008 and Windows Vista; however, some of Xperf's functionality will work on Windows Server 2003 and Windows XP. To install Xperf on Windows 2003 or XP, you'll have to first install Xperf on a Vista or Server 2008 system, then manually copy all the files from the installation directory to the Windows 2003 or XP system. For example, if you installed Xperf in a directory called c:\xperf on Vista or Server 2008, you'd simply copy the c:\xperf folder to the Windows 2003 or XP system.

## Scenario 1: High CPU-Usage Problem

Say you want to find the process that's hogging a large percentage of your system's CPU. Performance Monitor is a commonly used tool to help determine which process is consuming the CPU when the processor spikes. But you might not be able to correlate the high CPU usage to one process. The Processor\%DPC Time counter in Performance Monitor can help you determine whether the CPU spike is a result of a high level of Deferred Procedure Calls (DPCs)—system interrupts that run in the kernel.

# Examining Xperf

Michael Morales

(Reprinted from WindowsItPro Magazine)

DPCs are issued by kernel drivers, so the challenge in such cases is to identify which driver is responsible for the load on the system from DPC activity. By using Xperf as follows, you can make the driver-identification process much easier.

1. Install Xperf. I suggest installing it in an easy-to-navigate directory, such as c:\xperf.
2. Add the following environment variable:

```
NT_SYMBOL_PATH =  
srv*c:\symbols*http://msdl.microsoft.com/download/symbols
```

3. From a command prompt, navigate to the Xperf directory and type

```
C:\xperf>xperf -on latency
```

The latency flag tells Xperf to turn on a group of providers to start logging events. These events will be used to help diagnose which driver is consuming the highest percentage of CPU time.

4. Wait for the high DPC activity to occur, by monitoring it with Performance Monitor or Task Manager.
5. Then run this command:

```
c:\xperf -I \kernel.etl -a dpcisr
```

The command tells Xperf to process the default .etl—kernel.etl, and specifies an action (-a). Here the action specified is dpcisr, which will produce a report showing DPC and interrupt service routine (ISR) statistics, as Figure 1 shows.

# Examining Xperf

Michael Morales

(Reprinted from WindowsItPro Magazine)

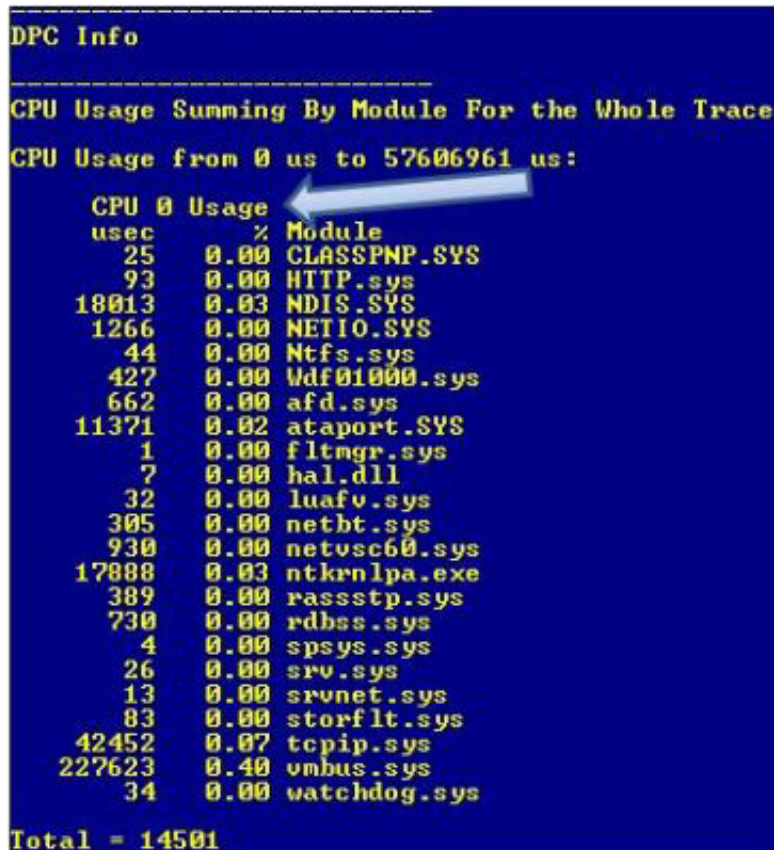


Figure 1  
Xperf Displaying The DPC Activity Per Driver

In Figure 1, the important area to look at is the Usage column, which tells you the percentage of CPU activity resulting from DPCs on a per-driver basis. This report provides a convenient means for quickly identifying which driver is responsible for the high amount of CPU usage. What makes this a nice story is that there are no other debuggers to install and no data file that a support professional needs to review. Using Xperf in this way provides an easy method for diagnosing a problem common in production environments.

Xperf is capable of so much more than simply reporting DPC information. Let's look at another usage scenario.

## Scenario 2: Uncovering Disk I/O Activity

Have you ever seen your hard drive lights constantly flashing and wondered what files were being accessed and what activity generated so much disk I/O? With Xperf, you can view spikes in disk I/O, then drill down to a specific I/O spike to determine what processes were accessing the disk at that time and what files were being accessed. Here's how to do it:

1. Use the same command as in step 3 in the previous section. If you don't specify a trace file, Xperf uses the default, kernel.etl.
2. Wait for the disk activity to spike.

# Examining Xperf

Michael Morales

(Reprinted from WindowsItPro Magazine)

3. Stop and merge the kernel .etl trace by issuing the command  
`c:\>xperf -d itpro.etl`
4. Now open the trace in the Xperf viewer by issuing the command

```
c:\>xperf itpro.etl
```

After running this command, you'll see Xperf's Windows Performance Analyzer window, which Figure 2 shows.

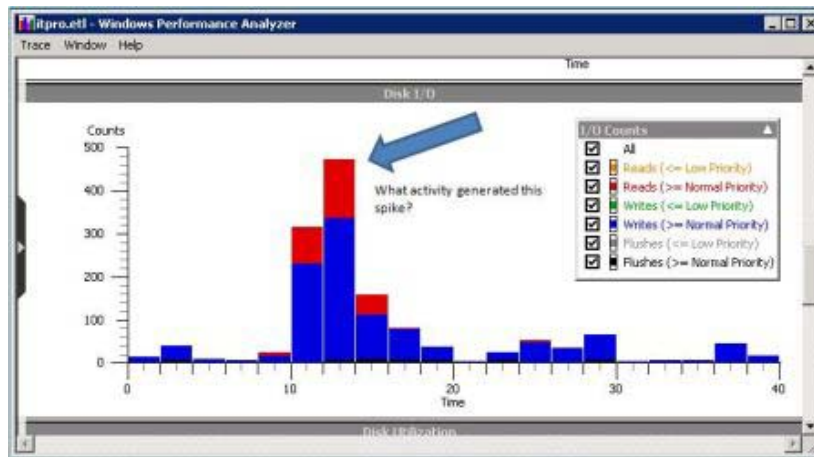


Figure 2  
Xperf Disk I/O view

In Figure 2, you can see Xperf makes it easy to identify the various spikes in disk I/O activity; however, we need to understand what activity on the system caused these spikes. To do so, simply highlight the spiked activity from within the Windows Performance Analyzer window, right-click it, and select Summary Table. Note that you'll now see a list of processes that generated disk I/O for the time period you highlighted. Click the plus sign next to each process to view the files accessed by these processes and the size of the I/O request, as Figure 3 shows.

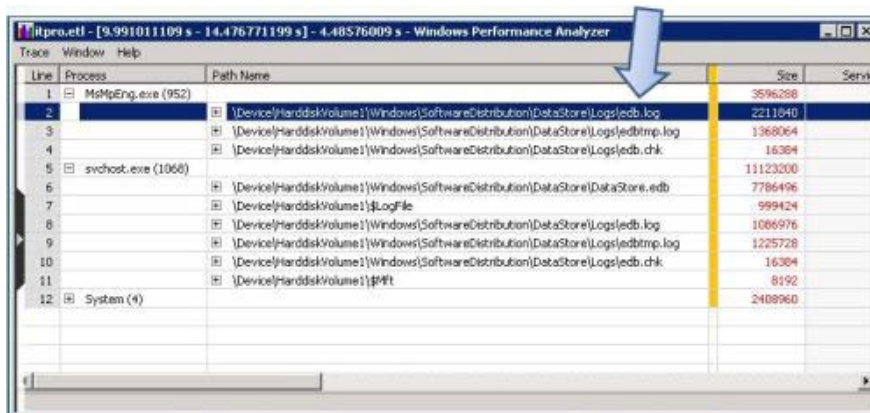


Figure 3  
Xperf summary detail view

# Examining Xperf

Michael Morales

(Reprinted from WindowsItPro Magazine)

From Figure 3, you can clearly see the processes involved in generating the I/O activity and the files accessed. For this example, the MsMpEng.exe process is from the Microsoft Forefront Client Security application, which accessed the edb.log file and the svchost.exe process for Windows Update, then accessed the DataStore.edb and the edb.log file.

I did a quick Internet search on edb.log and DataStore.edb and learned that they're files associated with Windows Update. So at the time of my test, Windows Update was running—which requires access to these specific files. And through Xperf, I determined which processes were responsible for generating specific disk activity and what files were accessed during that time. Although this was just a test, you can follow these same steps to determine what processes and files are responsible for generating a high amount of disk activity on your systems.

## Power-Packed Utility

There are other utilities that will generate similar information; however, Xperf is a single utility that's capable of much more than what I can cover in one article. In an upcoming article, I'll cover one of the best features of Xperf, the stackwalking profiler, which can help solve process-spike problems by letting you see which modules and functions inside a process are consuming the most CPU.