



"Knowing the name of a WMI provider DLL can help you find out whether a problem is a known issue or updates for the binary are available."

## Find the Binary File for Any WMI Class

Identify a WMI class's DLL binary to help you troubleshoot WMI-process-related problems

One of the really frustrating aspects in troubleshooting any Windows Management Instrumentation (WMI) issue is trying to determine which binaries are responsible for supporting the hundreds of classes registered in any given system. Each class in WMI has a corresponding WMI provider, which is simply a COM object and usually in the form of a DLL binary. Since WMI providers are COM objects, the main task will be to find the GUID registered for the binary, then search the registry for that GUID.

One typical problem with WMI is high CPU utilization. You may have experienced a high-CPU problem where the `WMIPrvse.exe` process was spiking the CPU, and you might have even known what WMI query generated all the CPU activity. But how do you find the actual DLL binary responsible for the class being queried? And why is knowing the DLL important? Knowing the name and location of a DLL will provide you information such as which vendor the binary belongs to, which will enable you to find out whether updates are available for the binary and whether the problem you're experiencing is a known issue and has been fixed already. Simply doing an Internet search on "high cpu `wmiPrvse.exe`" might return too many hits, none of which could be your problem. However, if you can include the exact DLL name in your search—for example, "high cpu `wmiPrvse provider.dll`"—the search results are more likely to return more accurate and relevant information. Let's look at how to find the DLL binary for a WMI class by walking through an example.

### Enabling Logging to Find the Class Namespace

The first step in finding the DLL of a particular class is to understand which namespace the class resides in. To accomplish this, we must enable logging. Here's how you do it.

**For Windows Server 2003 and Windows XP:** You can enable verbose logging through the WMI Control called `WMIMgmt.msc`. Click the control's Logging tab, then click the Verbose radio button. You'll notice entries in the `Wbemcore.log` file such as the following, indicating which namespace is being queried.

```
ConnectionLogin:NTLMLogin - wszNetworkResource = root\cimv2
```

**For Windows Vista and Windows Server 2008:** The logging

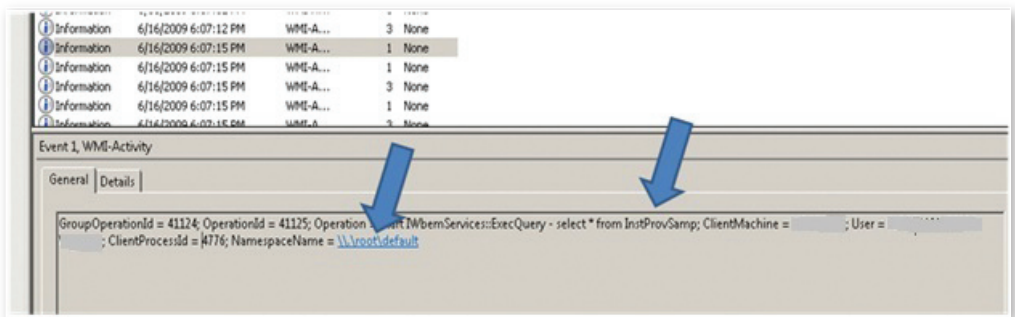


Figure 1: WMI logging

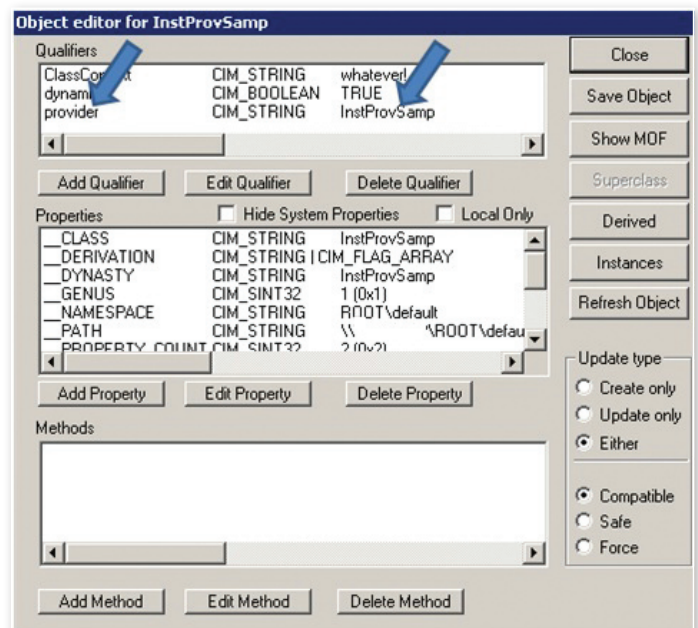


Figure 2: Object editor for InstProvSamp

## ■ WHAT WOULD MICROSOFT SUPPORT DO?

mechanism is new in Windows Vista and Server 2008, which now have what are called Analytic and Debug logs. These logs have replaced the previous verbose logging in Windows 2003 and XP. For detailed steps on how to enable the WMI Analytic and Debug logs on Windows Vista and later, see the Windows Management Infrastructure blog post "Is WMIprvse a real villain?" at [blogs.msdn.com/wmi/archive/2009/05/27/is-wmiprvse-a-real-villain.aspx](http://blogs.msdn.com/wmi/archive/2009/05/27/is-wmiprvse-a-real-villain.aspx).

For testing purposes, I've created a sample WMI class called `InstProvSamp`. In Figure 1, page 14, notice the event that logs the WMI query to the class (select \* from `InstProvSamp`) and the namespace where the class resides (`\\root\\default`).

### Finding the Provider DLL

Now that we have the WMI class and namespace, we can start our search for the provider DLL. Every WMI class has a qualifier called *provider*, which is the name of the provider but not necessarily the binary name. We need to find out what the qualifier name is for our test class (`InstprovSamp`), and we can use `wbemtest.exe` to do so.

Run `wbemtest.exe`, connect to `root\\default`, then click the Open Class button. Enter the class name—in our example, `InstProvSamp`. Click OK, and now notice that the Object editor opens for this class. As you can see in Figure 2, page 14, under the Qualifiers, there is a "provider" string called `InstProvSamp`. Here, the provider name is the same as the class name, but usually this isn't the case.

Now we need to get the class identifier or CLSID of the provider from the provider name. To do so, you submit a WMI query for the system class called `__Win32Provider`, which will contain the CLSID or

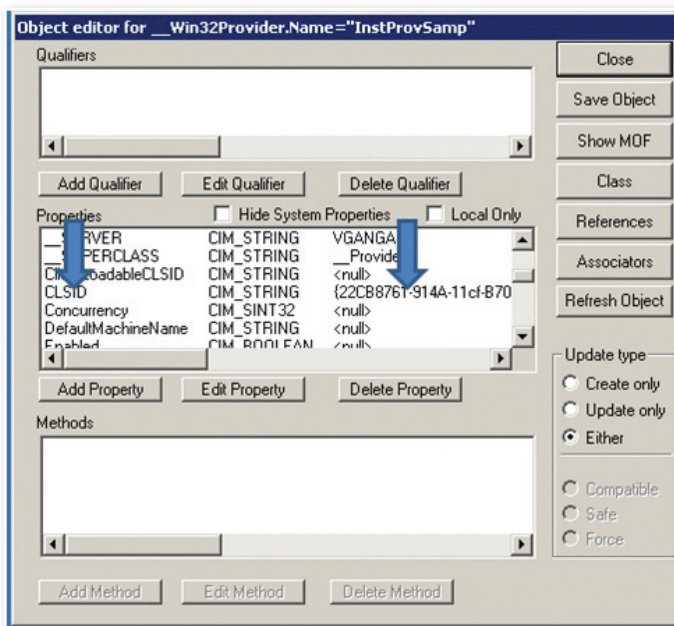


Figure 3: Object editor for `__Win32Provider.Name`

GUID of the provider COM object. Run `wbemtest.exe`, connect to the target namespace (`root\\default`), click the Query button, and issue the following query:

```
Select * from __Win32Provider where  
name="InstProvSamp"
```

In the query result box, you'll see an entry like the following:

```
__Win32Provider.Name="InstProvSamp"
```


Open the Object editor by double-clicking this entry, as Figure 3 shows.

Notice the CLSID entry pointing to the GUID that we need; every COM object places a registration entry under `HKEY_LOCAL_MACHINE\\SOFTWARE\\Classes\\`

CLSID. We can use `regedit.exe` to pinpoint our search directly to the COM registration information and get right to the exact name and location of the WMI provider binary, as Figure 4 shows.

As I mentioned, the provider binary name doesn't always match the class name. For instance, the registry provider is defined under the `root\\default` class, and the `__Win32Provider` name is `RegProv`. However, the actual DLL name is `STDPROV.DLL` and is located under the `c:\\windows\\system32\\wbem` directory.

### More Uses

There are several reasons why you might need to find the binary name for a given WMI class; I only mentioned CPU utilization as one possible scenario. Although your scenario may be different, the steps I've described will work to find the WMI provider binary name and location for every scenario. Good luck, and as usual please feel free to contact me regarding this article or any of my previous articles. 

InstantDoc ID 102615

**MICHAEL MORALES** ([morales@microsoft.com](mailto:morales@microsoft.com)) is a senior escalation engineer for Microsoft's Global Escalation Services team. He specializes in advanced Windows debugging and performance-related issues. For information about Windows debugging, visit [blogs.msdn.com/ntdebugging](http://blogs.msdn.com/ntdebugging).

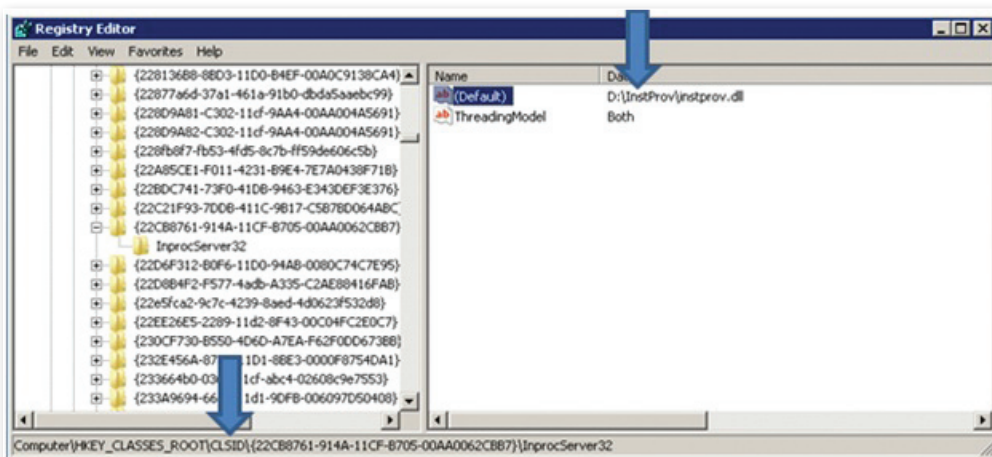


Figure 4: CLSID registration

Copyright of *Windows IT Pro* is the property of Penton Publishing and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.