

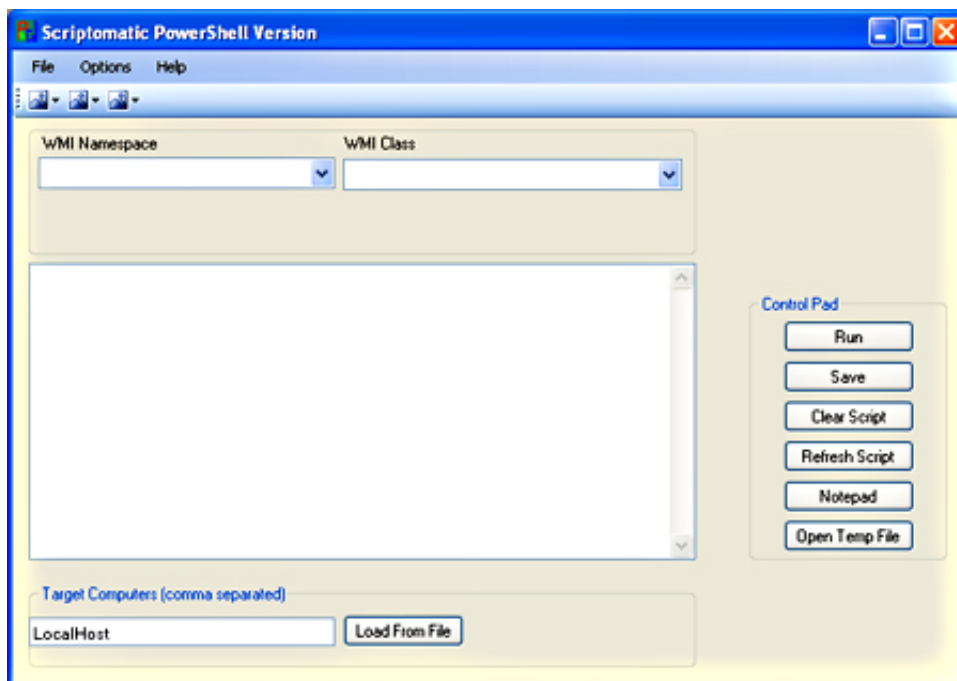
PowerShell Scriptomatic

29 out of 33 rated this helpful

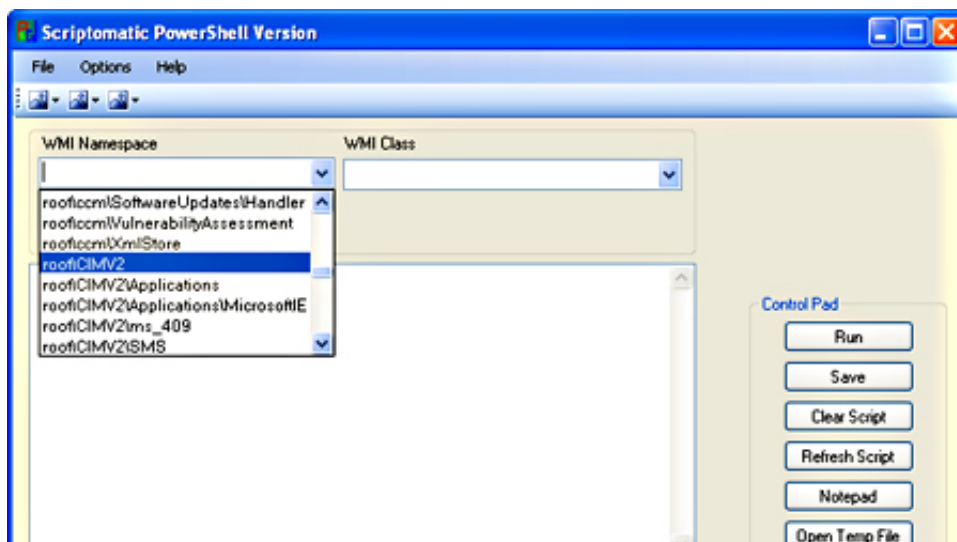
Have you ever found yourself thinking, “I wonder when the Scripting Guys are going to write a PowerShell Scriptomatic”? Well, you can stop wondering: the Scripting Guys will probably *never* write a PowerShell Scriptomatic, a utility that would make it a snap to create WMI scripts using Windows PowerShell. Is that because the Scripting Guys don’t believe such a tool would be useful? Heck no; we think a PowerShell Scriptomatic would be *incredibly* useful. So then why aren’t we going to write such a tool? One reason and one reason only: Ed Wilson has already written a PowerShell Scriptomatic for us.

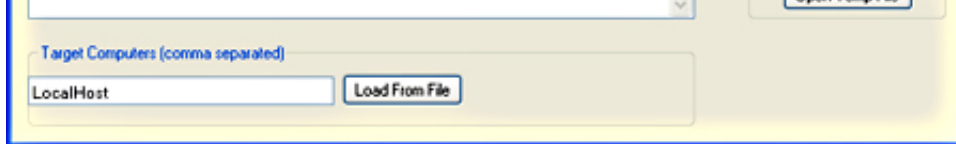
For those of you who don’t know the prolific Mr. Wilson, Ed is the author of about half-a-zillion books for Microsoft Press, including **Microsoft Windows PowerShell Step-by-Step** and the forthcoming **Windows PowerShell Scripting Guide**. In conjunction with his latest book, Ed has also put together the Windows PowerShell Scriptomatic, a scripting utility you can [download from here](#).

Ah, good question: what exactly *is* a Windows PowerShell Scriptomatic? Well, when you first load the PowerShell Scriptomatic you see a window very similar to this:

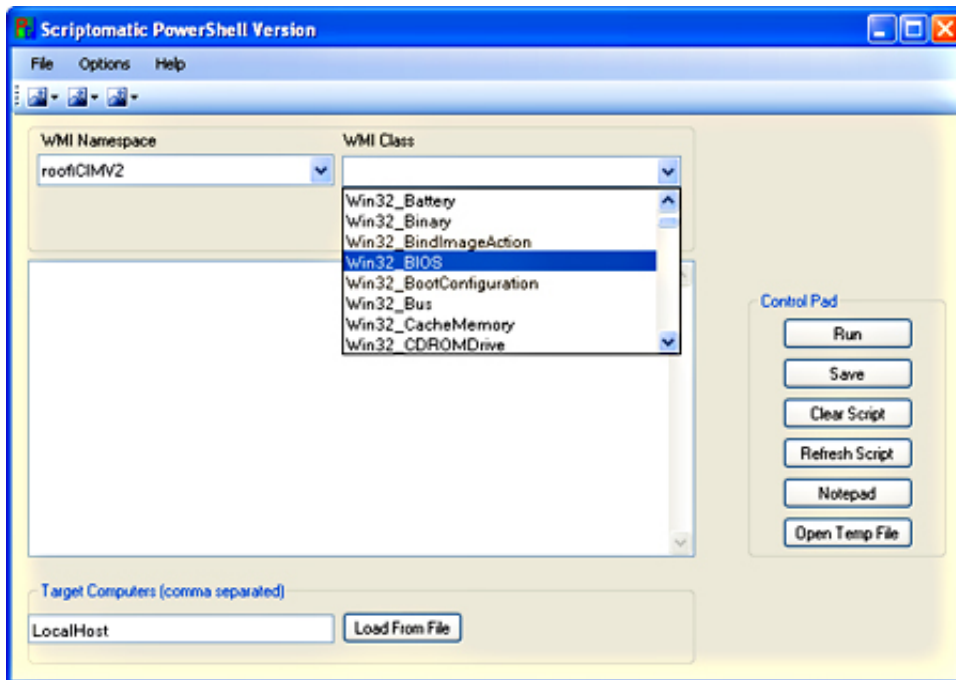


Don’t let the small size and the clean, crisp interface fool you; Ed has packed quite a bit of power into this little package. For example, do this: click the dropdown list labeled **WMI Namespace**. When you do that, and in a matter of seconds, you should see all the WMI namespaces available on the local computer:



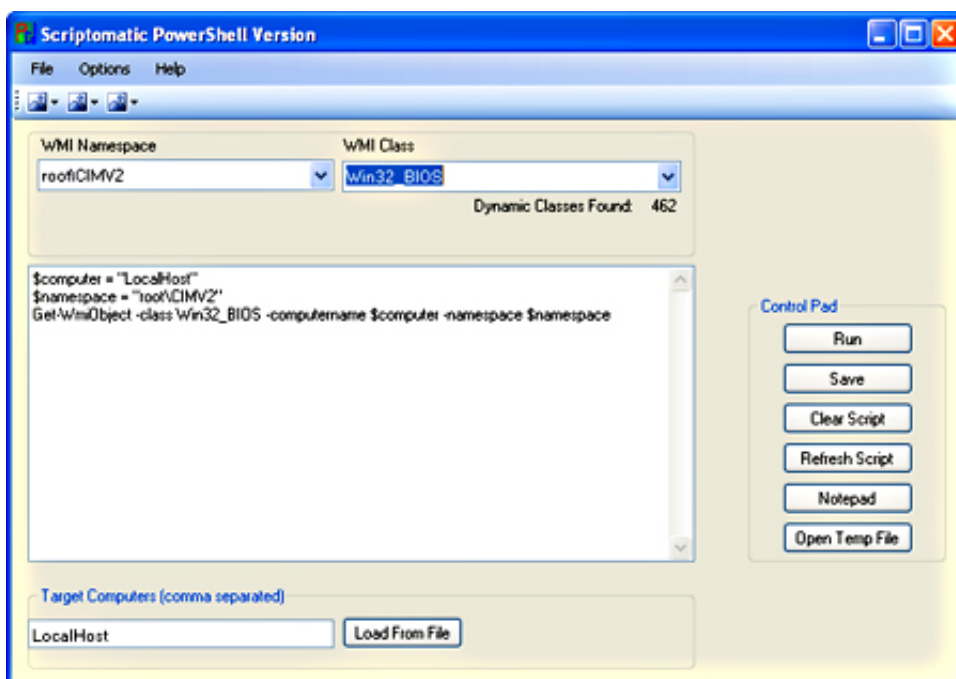


Select the namespace **root\CIMV2**. Now click the dropdown labeled **WMI Class**. When you do *that*, you should see a list of all the dynamic WMI classes found in the root\CIMV2 namespace:



Note. What's a dynamic class? In WMI, a dynamic class is a class that actually returns information. There are plenty of other classes found in the root\CIMV2 namespace, but many of these do not return data. (Instead, they often function as templates from which dynamic classes are derived.) These classes are of minimal value to system administrators, so Ed has chosen not to display them in the Scriptomatic.

Now pick a class; for example, choose **Win32_BIOS**. The moment you make a selection, the Scriptomatic writes a Windows PowerShell script designed to return all the information that the Win32_BIOS class can return:



Nice, huh? But it gets even nicer. Click the **Run** button and the PowerShell Scriptomatic will start an instance of Windows PowerShell and run your script for you:

```
C:\WINDOWS\system32\WindowsPowerShell\v1.0\PowerShell.exe
Description      : EPP runtime BIOS - Version 1.1
IdentificationCode :
InstallableLanguages :
InstallDate      :
LanguageEdition  :
ListOfLanguages  :
Manufacturer     : Hewlett-Packard
Name             : EPP runtime BIOS - Version 1.1
OtherTargetOS    :
PrimaryBIOS      : True
ReleaseDate      : 20060615000000.000000+000
SerialNumber     :
SMBIOSBIOSVersion : 68DIT Ver. F.0F
SMBIOSMajorVersion : 2
SMBIOSMinorVersion : 3
SMBIOSPresent    : True
SoftwareElementID : EPP runtime BIOS - Version 1.1
SoftwareElementState : 3
Status           : OK
TargetOperatingSystem : 0
Version          : HP      - 15060620

PS C:\Program Files\MrEdSoftware\PowerShellScriptOMATIC v.1.0>
```

See? We told you it would get even nicer.

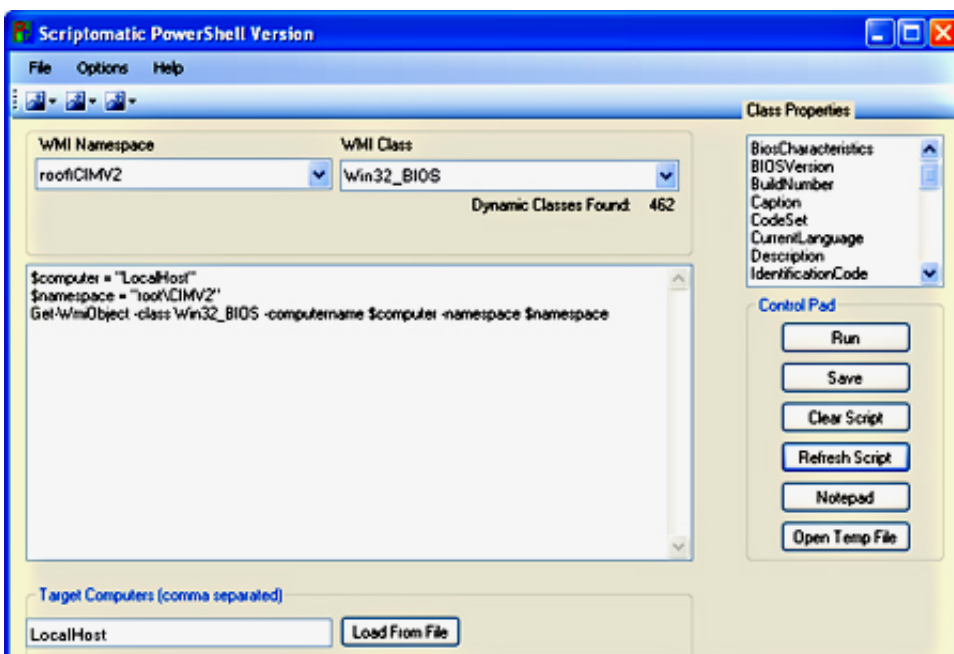
That's the basic idea behind the PowerShell Scriptomatic: it makes it easy to create, run, and save WMI scripts written in Windows PowerShell. Of course, what would a PowerShell Scriptomatic be without some additional options? For example, if you click the first icon on the toolbar you can do such things have the data saved to a text file, an XML file, or a CSV (comma-separated values) file. Here's another example of what the Scriptomatic can do. Click the option **Use All Properties** and the Scriptomatic will return *all* the properties of the WMI class. (By default, PowerShell typically returns only selected properties of a WMI class.)

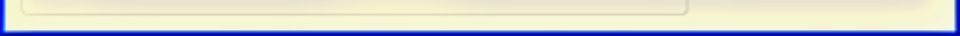
What's that? You say you'd like to run this script against a *remote* computer? That's fine; just type the name of that computer (or its IP address) into the text box labeled **Target Computers** and then click **Refresh Script**. Want to run this script against *multiple* computers? Again, no problem: just type the names of each computer into the text box, making sure to separate the computer names with commas (and with no spaces between those commas):

```
atl-fs-001,atl-fs-002,atl-fs-003
```

Etc.

Oh, and try this: click the third icon on the toolbar and select **Display Class Properties**, then click **Refresh Script**. Now take a look at your Scriptomatic window: in addition to writing a script for you, it also displays the properties of the selected WMI class:





In other words, you now have an easy-to-use WMI browser. *Very cool.*

But don't just take our word for it; download the **PowerShell Scriptomatic** and give it a try. Will this prove to be the best thing you've ever done in your life? Well, maybe. But, at the very least, it should be one of the 4 or 5 best things you've ever done in your life.

[Top of page](#)

© 2013 Microsoft. All rights reserved.