Use Windows Server 2008 Powershell to perform network commands

There are many things you still do every day while administering a Windows network, but if you are asked to perform those tasks from the command line, most Windows administrators may feel it is a difficult request.

David Davis

There are many things you still do every day while administering a Windows network, but if you are asked to perform those tasks from the command line, most Windows administrators may feel it is a difficult request. Windows is always weak when it comes to command line tools, but this has changed with the appearance of Windows Powershell. Powershell (formerly known as PS) can do many things that were previously impossible. In this article, we will talk about Powershell and how it can help you perform some of the functions in general network administration from the command prompt.

What is Powershell?

Powershell is a feature that can be installed in Windows Server 2008. To install Powershell, you must install the Powershell feature in the Add Features Wizard. It only takes a few minutes to install and when this feature is installed you can access an interesting command line scripting language. Unlike other scripting languages ??in Windows, Powershell is designed for system administrators. Powershell uses .NET and 'cmdlets' (or 'command-lets') to perform its work. As a Powershell user, you can use the cmdlet itself or you can connect them together to perform more powerful tasks.

Once you've installed Powershell, you should be able to go to **Start** -> **All Programs** -> **Windows Powershell 1.0** , and click **Windows PowerShell** . Here, you will see a blue CLI screen appear as shown below:

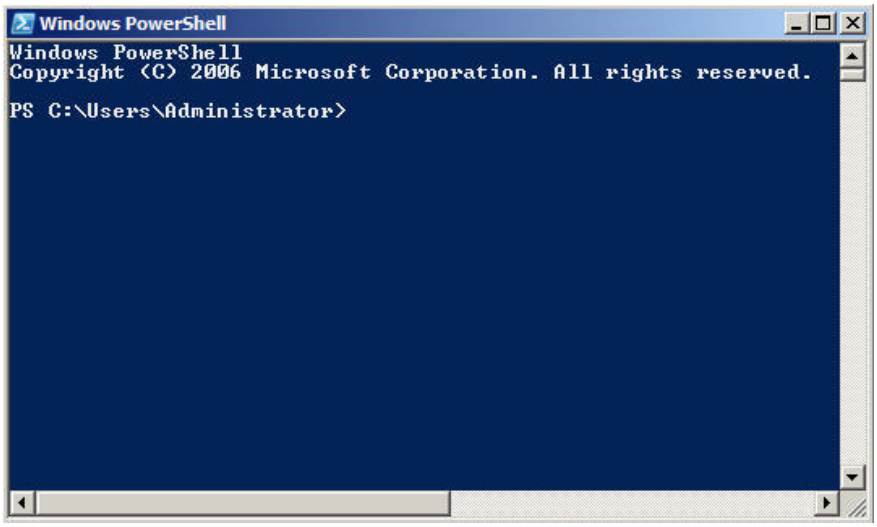

Figure 1: Windows Powershell command prompt

You can always know you're in Powershell because 'PS' is at the beginning of the command prompt:

PS C: UsersAdministrators

Now that you have installed PowerShell and are in the command window, I will show you some common network tasks that can be done with Powershell.

List the IP address on the server

To list IP addresses on Windows 2008 Server, use the following command string:

Get-WmiObject -Class Win32_NetworkAdapterConfiguration -Filter IPEnabled = TRUE -ComputerName. | Select-Object -Property IPAddress

This is the result after executing the above command with Windows Server 2008:

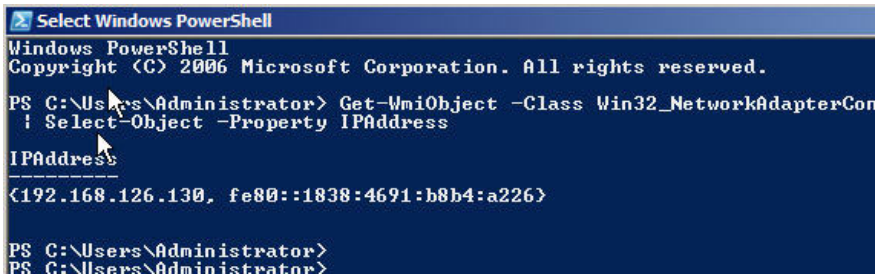

Figure 2: List IP address with Windows Powershell

As you can see, the output shows us that there is an adapter with V4 IP address and an IPv6 address on Windows Server 2008. What can you do when combined with other scripting functions.

As in the manual for Powershell users, the output is an array and you only see the IP address by directing the output to 'Select-Object' (after disabling IPV6), like this:

Get-WmiObject -Class Win32_NetworkAdapterConfiguration -Filter IPEnabled = TRUE -ComputerName. | Select-Object -ExpandProperty IPAddress

Figure 3: List IP address with Windows Powershell only

List network adapter configuration with Powershell

To show the basic configuration of the network adapter, you can use the following command:

Get-WmiObject -Class Win32_NetworkAdapterConfiguration -Filter IPEnabled = TRUE –ComputerName

While this article focuses on using Powershell in Windows Server 2008, it can also be used in Windows XP, Vista or Windows Server 2003. In fact, this is the sample output for the above command, executed on Windows XP workstation:

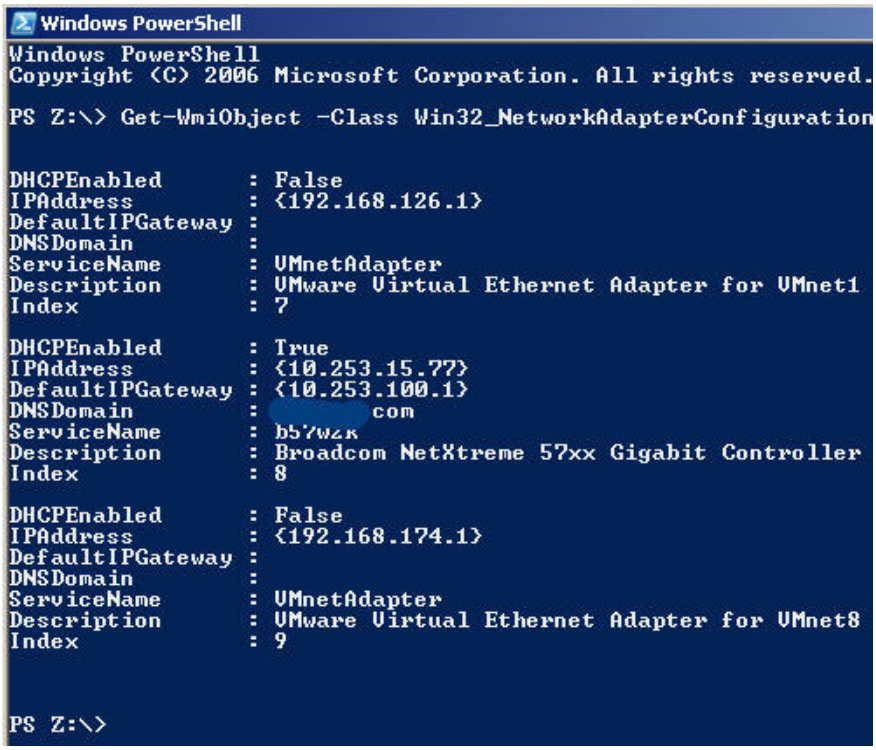

Figure 4: Powershell displays the network adapter configuration in Windows XP

Ping the computer with Powershell

While Powershell is still able to perform all of the usual Windows commands (like the ping command), Powershell's power is also expressed in that you can take that output and change it easily.

Here's an example of that, thanks for help in the Windows Powershell user guide. In this example, the output of Win32_PingStatus is separated by Select-Object. In this case, the output only shows the response time and status code.

This is the newly used command:

> **Get-WmiObject -Class Win32_PingStatus -Filter "Address = '127.0.0.1'" -ComputerName. | Select-Object -Property Address, ResponseTime, StatusCode**

And this is the output on the computer:

Figure 5: PowerShell output shows ping status with Select-Object

Share folders with Windows Powershell

It must be said that using Powershell commands is not always as easy as existing Windows commands that you are still used to. Here is an example.

The Powershell command below will share a directory with the path C: temp like 'davidtemp' and place a command on it:

> **(Get-WmiObject -List -ComputerName. Where-Object -FilterScript {$ _. Name -eq "Win32_Share"}) InvokeMethod ("Create", ("C: temp", "davidtemp", 0.25, " David's Temp Folder ")))**

Alternatively, you can only use the **net share** command as follows:

> **net share davidtemp = C: temp / remark: "David's Temp Shared Folder"**

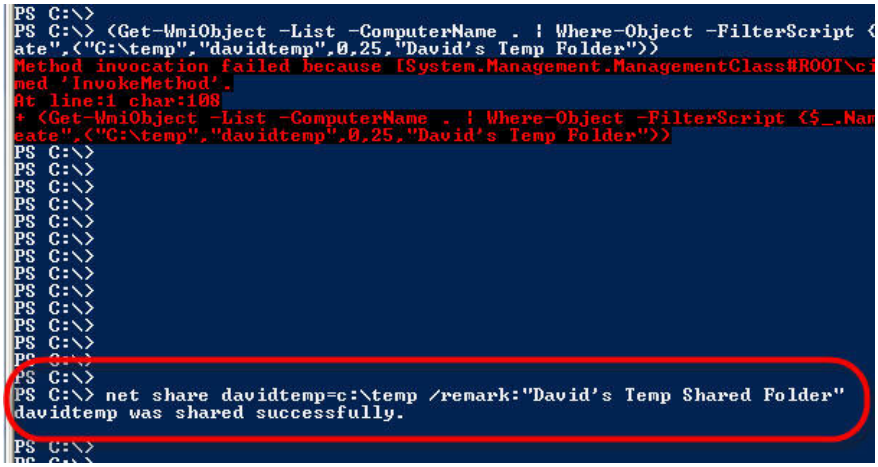

Figure 6: Powershell output and common sharing commands

Notice how the Powershell command did not work and gave a very annoying error message.

With a simple **net share** command, it worked immediately. So while we show you all that can be done with Powershell, you should not forget that Powershell can not only run all network-related commands, but also in many cases. It also allows for easier use.

More complex commands

Next, if you only perform basic network administration commands with Powershell, you might be bored and don't want to use it. It is also because like scripting languages, there are some places that you need to learn.

However, once you learn how to use it, you will get all the advantages of efficiency as well as save time from a newly learned language.

As you can note from some of the previous examples, one of the parameters for most Powershell commands is 'computername'. When placing a dot ('.') As the computer name, it is a local host (our computer). However, we can also replace any IP address or computer name in the domain to not only use these commands on your computer but also create powerful scripts with Powershell so that you can run on all computers in your network.

For example, this is a command to drag all an IP address from a Dell computer to a LAN:

> **Get-WmiObject -Class Win32_NetworkAdapterConfiguration -Filter IPEnabled = TRUE -ComputerName DELL9400**

Here are the results:

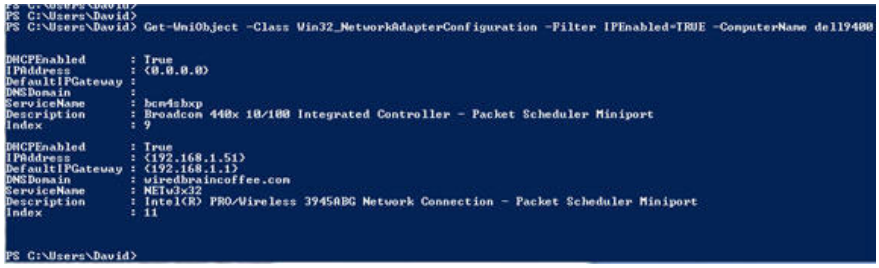

Figure 7: Powershell results from querying the remote computer's IP address

And here is the enlarged screen to display the information inside:

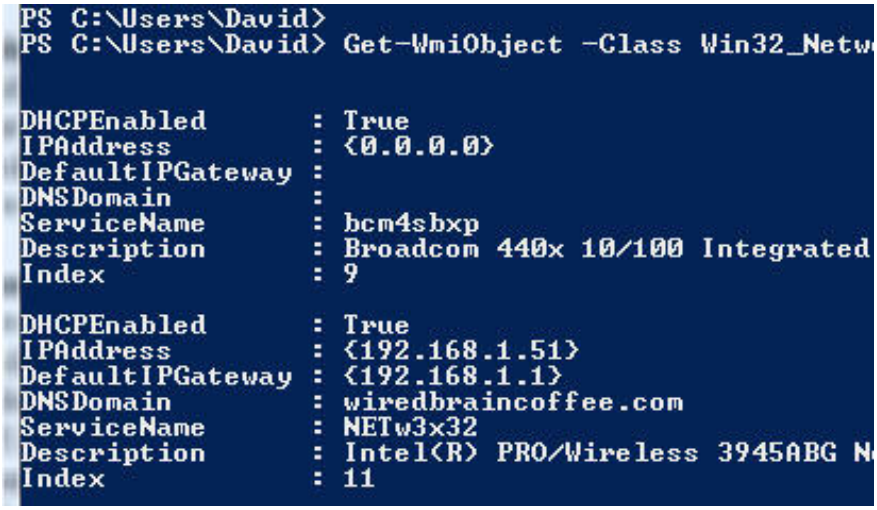

Figure 8: Closing Powershell results against querying the remote computer's IP address

So the ability to perform on remote computers is a key feature of Powershell, but another feature is the ability to filter output and combine output from one command to another.

We will see that problem in an example:

> **"127.0.0.1", "localhost", "research.microsoft.com" | ForEach-Object -Process {Get-WmiObject -Class Win32_PingStatus -Filter ("Address = '" + $ _ + "'") -ComputerName.} | Select-Object -Property Address, ResponseTime, StatusCode**

In this example, the list is provided with IP addresses and domain names. That list is sent to 'ForEach-Object'. On each of those objects (domain name or IP address), 'Get-WmiObject' PingStatus will be run. The ping output for each of these domains is then separated with 'Select-Object' and only the address, response number and status code of the instance.

Here is the output:

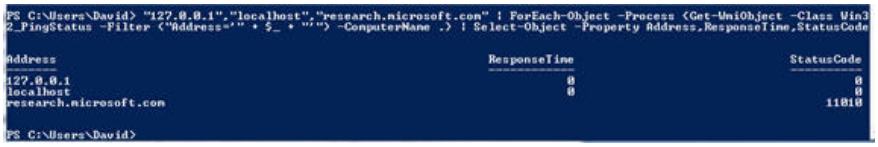

Figure 9: Ping using the list by combining and sending the output

We think this example has shown some of the powers of Powershell. As you can see, it is possible to direct or indirectly direct input and output in different directions to accomplish your system administration goals.

Need to know more?

As mentioned earlier, note that PowerShell not only runs on Windows Server 2008 but can also be used in Windows XP, Vista and Server 2003. Powershell is completely free and you can download it. It is easy with 6MB capacity.

Conclude

Windows Powershell is a powerful tool. This article cannot explain everything you can do with Powershell, but we hope it also gives you an idea of ??what can and inspire you to study more about Powershell. In addition, every day we have more new books, classes and a lot of websites that prove Powershell's usefulness to Windows administrators. For Windows administrators who are using the GUI interface, they will spend more time with Powershell for approval in companies, where it is not immediately necessary. However, we believe that administrators will use Powershell to create many shorter scripts to perform certain functions, besides being able to combine with other scenarios to perform more complex functions. .