Microsoft Certified Professional MCP MAG

POWERSHELL PIPELINE

Managing Services, Part 1: Investigating Service Information Using PowerShell

Here's a few different ways to finding services info through PowerShell.

By Boe Prox • 12/12/2014

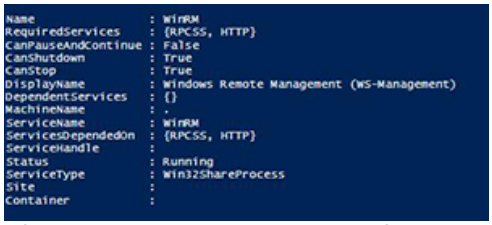
When a key service goes down on a server, you will usually hear about it from a user or another office such as the help desk when they begin receiving calls from users. It's also entirely possible that these services may have been down for a while and no one really noticed until they happened to access a specific resource or service from a server and realized that it was no longer available. Typically it is something like a service that is set to be an automated service, meaning that when a server is rebooted for something like an update, the services will automatically start back up while those which are set to manual or disabled will not start back up.

Fortunately, we can take a look at our services to find out if our servers have services which might be set to automatic and not running or we want to take a look at some other properties of a service to get more information from them.

Get-WMIObject/CIMInstance or Get-Service?
We have a couple of options here on what we can use to look at services on a local or remote system and each of these has their pros/cons depending on what you are looking for.

For instance, if you are interested in looking at finding out what the StartMode of a service is such as whether it is configured to automatically startup at system startup or if it should be disabled, you will have to look at WMI as the appropriate method to find this information out. Using Get-Service will not pull this information for you as shown in the examples below.

```
Get-Service -Name WinRM | Select-Object *
```



[Click on image for larger view.] Figure 1. *No visible StartMode with Get-Service*

```
Get-WMIObject -Class Win32_Service -Filter  "Name='WinRM'" |

  Select-Object Name, DisplayName, StartMode, Status
```

[Click on image for larger view.] Figure 2. *StartMode is set to Auto using Get-WMIObject approach*

We can see that using Get-WMIObject and looking at the Win32_Service class, or more specifically, the WinRM service shows that it has a start mode of Automatic while looking at the services using Get-Service and listing all of the properties does not show this as an available property.

The differences don't stop there! We can also look at what account is actually running the service by looking at the WMI class vs. using Get-Service.

```
Get-WMIObject -Class Win32_Service -Filter  "Name='WinRM'" |

  Select-Object Name, DisplayName, StartMode, Status, StartName
```

[Click on image for larger view.] ***Figure 3.*** *Looking at the account running the service.*

This is very handy if you are looking for all of the accounts that are running as a service in your environment. A quick command can be used to report on accounts when it comes time for a password reset on those accounts or if you are looking for service accounts that are running as Domain Admins so they can be reviewed and (hopefully) lower their privileges.

Ok, it is not all gloom and doom with Get-Service and its returning back a ServiceController object that doesn't provide anything useful. Using Get-Service, I can very easily find out what services depend on a service or the opposite of what service depends on specific services.

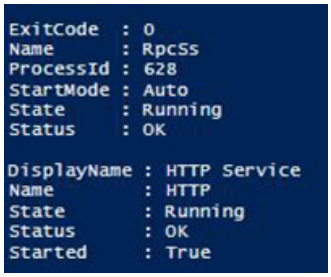
For instance, if I want to see what services have to be running before a specific service can start, I can simply use the –RequiredServices parameter and find this information out.

```
Get-Service -Name WinRM -RequiredServices
```

[Click on image for larger view.] ***Figure 4.*** *Required services before WinRM will start up.*

Can I do this through WMI? Sure, but it isn't quite as simple ...well unless you know your way around WMI queries, then maybe it might be simple.

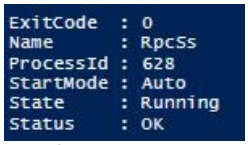
```
Get-WMIObject -Query "Associators of {Win32_Service.Name='WinRM'} Where  AssocClass=Win32_DependentService Role
```



[Click on image for larger view.] ***Figure 5.*** *The WMI approach to finding required services.*

There is actually another way to do this with WMI, but I will warn you that its accuracy can sometimes be a little off depending on how you choose to perform this search. We can take the object of the service and use the GetRelated() method to find any WMI object that has a relationship to this particular service. This also takes an optional parameter string which is a WMI class to narrow down the scope of what you want to return.

```
(Get-WMIObject -Class Win32_Service -Filter  "Name='WinRM'").GetRelated('Win32_Service')
```

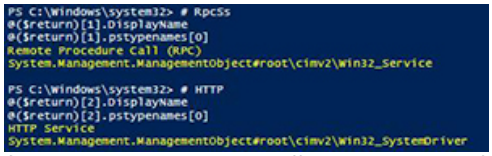


[Click on image for larger view.] ***Figure 6.*** *Single object returned from using GetRelated('Win32_Service')*

Wait a minute! What happened to my other service? Surely it is still related to WinRM? It turns out that if we just use GetRelated(), we get back three WMI classes: Win32_Service, Win32_ComputerSystem and Win32_SystemDriver in which the Win32_SystemDriver is actually the HTTP Service that we saw using Get-Service. If you aren't sure that you believe me, give this a try and see what you get.

```
$return = (Get-WMIObject -Class Win32_Service -Filter  "Name='WinRM'").GetRelated()

  # RpcSs

  @($return)[1].DisplayName

  @($return)[1].pstypenames[0]

  # HTTP

  @($return)[2].DisplayName
```

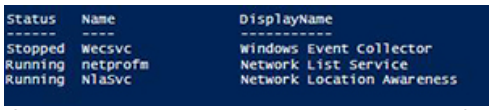
```
@($return)[2].pstypenames[0]
```



[Click on image for larger view.] Figure 7. *Different WMI classes for related objects.*

Lastly, we can also find out what services depend on a service using the –DependentServices parameter on Get-Service.

```
Get-Service -Name eventlog -DependentServices
```



[Click on image for larger view.] Figure 8. *Dependent services for eventlog service.*

As with the required services, getting the dependent services from WMI requires a little more work.

```
Get-WMIObject -Query "Associators of {Win32_Service.Name='eventlog'}  Where AssocClass=Win32_DependentService F
```

[Click on image for larger view.] Figure 9. *Dependent services from WMI.*

Even this approach appears to be not as accurate as going the Get-Service route. This is something that you might want to keep in mind when you are generating reports on services and perhaps a mix of Get-Service and looking at the WMI classes may present the most accurate information.

One last thing. If you are interesting in knowing what services that you can actually stop on a server, then you can run a simple query to locate those services:

```
Get-WMIObject -Class Win32_Service -Filter  "AcceptStop = 'true'" |

  Select-Object DisplayName, Name, AcceptStop
```

The Get-Service approach would be something like this:

```
Get-Service | Where-Object {

  $_.CanStop

}  | Select Name, DisplayName,  CanStop
```

In this article I have shown you a few different approaches to finding out information about the services on your system and also showed that with these options, combining the approaches could lead you to providing the most accurate information on the state of your services.

About the Author

Boe Prox is a Microsoft MVP in Windows PowerShell and a Senior Windows System Administrator. He has worked in the IT field since 2003, and he supports a variety of different platforms. He is a contributing author in PowerShell Deep Dives with chapters about WSUS and TCP communication. He is a moderator on the Hey, Scripting Guy! forum, and he has been a judge for the Scripting Games. He has presented talks on the topics of WSUS and PowerShell as well as runspaces to PowerShell user groups. He is an Honorary Scripting Guy, and he has submitted a number of posts as a to Microsoft's Hey, Scripting Guy! He also

has a number of open source projects available on Codeplex and GitHub. His personal blog is at http://learn-powershell.net.