**Getting WMI Objects (Get-CimInstance)**

* Article
* 10/08/2021
* 3 minutes to read
* 2 contributors

**Getting WMI Objects (Get-CimInstance)**

Windows Management Instrumentation (WMI) is a core technology for Windows system administration because it exposes a wide range of information in a uniform manner. Because of how much WMI makes possible, the PowerShell cmdlet for accessing WMI objects, Get-CimInstance, is one of the most useful for doing real work. We are going to discuss how to use the CimCmdlets to access WMI objects and then how to use WMI objects to do specific things.

**Listing WMI Classes**

The first problem most WMI users encounter is trying to find out what can be done with WMI. WMI classes describe the resources that can be managed. There are hundreds of WMI classes, some of which contain dozens of properties.

Get-CimClass addresses this problem by making WMI discoverable. You can get a list of the WMI classes available on the local computer by typing:

PowerShellCopy

Get-CimClass -Namespace root/CIMV2 |

Where-Object CimClassName -like Win32\* |

Select-Object CimClassName

OutputCopy

CimClassName

------------

Win32\_DeviceChangeEvent

Win32\_SystemConfigurationChangeEvent

Win32\_VolumeChangeEvent

Win32\_SystemTrace

Win32\_ProcessTrace

Win32\_ProcessStartTrace

Win32\_ProcessStopTrace

Win32\_ThreadTrace

Win32\_ThreadStartTrace

Win32\_ThreadStopTrace

...

You can retrieve the same information from a remote computer by using the **ComputerName** parameter, specifying a computer name or IP address:

PowerShellCopy

Get-CimClass -Namespace root/CIMV2 -ComputerName 192.168.1.29

The class listing returned by remote computers may vary due to the specific operating system the computer is running and the particular WMI extensions added by installed applications.

**Note**

When using CIM cmdlets to connect to a remote computer, the remote computer must be running WMI and the account you are using must be in the local administrators group on the remote computer. The remote system does not need to have PowerShell installed. This allows you to administer operating systems that are not running PowerShell, but do have WMI available.

**Displaying WMI Class Details**

If you already know the name of a WMI class, you can use it to get information immediately. For example, one of the WMI classes commonly used for retrieving information about a computer is **Win32\_OperatingSystem**.

PowerShellCopy

Get-CimInstance -Class Win32\_OperatingSystem

OutputCopy

SystemDirectory Organization BuildNumber RegisteredUser SerialNumber Version

--------------- ------------ ----------- -------------- ------------ -------

C:\WINDOWS\system32 Microsoft 18362 USER1 00330-80000-00000-AA175 10.0.18362

Although we are showing all of the parameters, the command can be expressed in a more succinct way. The **ComputerName** parameter is not necessary when connecting to the local system. We show it to demonstrate the most general case and remind you about the parameter. The **Namespace** defaults to root/CIMV2, and can be omitted as well. Finally, most cmdlets allow you to omit the name of common parameters. With Get-CimInstance, if no name is specified for the first parameter, PowerShell treats it as the **Class** parameter. This means the last command could have been issued by typing:

PowerShellCopy

Get-CimInstance Win32\_OperatingSystem

The **Win32\_OperatingSystem** class has many more properties than those displayed here. You can use Get-Member to see all the properties. The properties of a WMI class are automatically available like other object properties:

PowerShellCopy

Get-CimInstance -Class Win32\_OperatingSystem | Get-Member -MemberType Property

OutputCopy

TypeName: Microsoft.Management.Infrastructure.CimInstance#root/cimv2/Win32\_OperatingSystem

Name MemberType Definition

---- ---------- ----------

BootDevice Property string BootDevice {get;}

BuildNumber Property string BuildNumber {get;}

BuildType Property string BuildType {get;}

Caption Property string Caption {get;}

CodeSet Property string CodeSet {get;}

CountryCode Property string CountryCode {get;}

CreationClassName Property string CreationClassName {get;}

CSCreationClassName Property string CSCreationClassName {get;}

CSDVersion Property string CSDVersion {get;}

CSName Property string CSName {get;}

CurrentTimeZone Property short CurrentTimeZone {get;}

DataExecutionPrevention\_32BitApplications Property bool DataExecutionPrevention\_32BitApplications {get;}

DataExecutionPrevention\_Available Property bool DataExecutionPrevention\_Available {get;}

...

**Displaying Non-Default Properties with Format Cmdlets**

If you want information contained in the **Win32\_OperatingSystem** class that is not displayed by default, you can display it by using the **Format** cmdlets. For example, if you want to display available memory data, type:

PowerShellCopy

Get-CimInstance -Class Win32\_OperatingSystem |

Format-Table -Property TotalVirtualMemorySize, TotalVisibleMemorySize,

FreePhysicalMemory, FreeVirtualMemory, FreeSpaceInPagingFiles

OutputCopy

TotalVirtualMemorySize TotalVisibleMemorySize FreePhysicalMemory FreeVirtualMemory FreeSpaceInPagingFiles

---------------------- ---------------------- ------------------ ----------------- ----------------------

33449088 16671872 6451868 18424496 16285032

**Note**

Wildcards work with property names in Format-Table, so the final pipeline element can be reduced to Format-Table -Property Total\*Memory\*, Free\*

The memory data might be more readable if you format it as a list by typing:

PowerShellCopy

Get-CimInstance -Class Win32\_OperatingSystem | Format-List Total\*Memory\*, Free\*

OutputCopy

TotalVirtualMemorySize : 33449088

TotalVisibleMemorySize : 16671872

FreePhysicalMemory : 6524456

FreeSpaceInPagingFiles : 16285808

FreeVirtualMemory : 18393668

Name : Microsoft Windows 10 Pro|C:\WINDOWS|\Device\Harddisk0\Partition2