ATTACK DEFENSE

by PentesterAcademy

Name	WMI: Namespaces and Classes
URL	https://attackdefense.com/challengedetails?cid=2076
Type	Services Exploitation: WMI

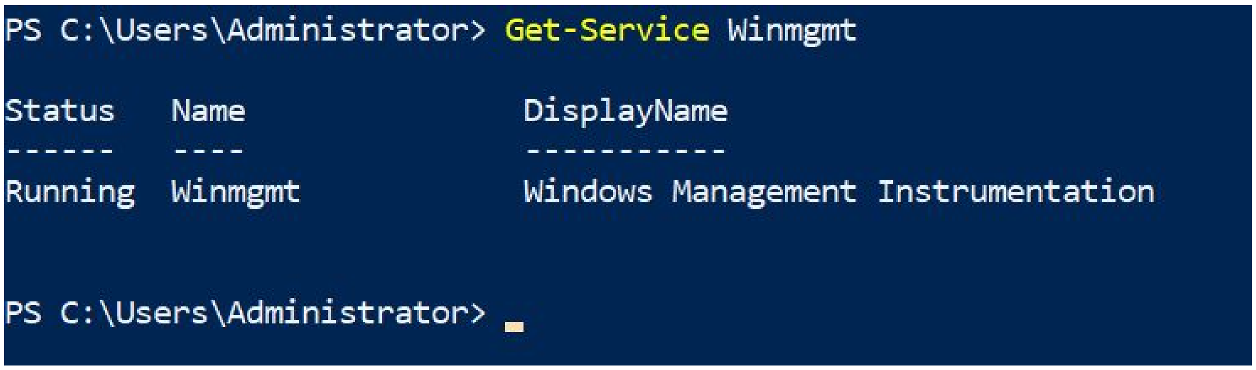
Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Note: By default, if you are using Windows Server then, the WMI service is already up and running. You need to configure the service in order to access it remotely. In this manual, we are demonstrating how to configure WMI service and making necessary changes for learning purposes.

Exploring WMI Namespaces and Classes

Step 1: Run powershell.exe to check for wmi service status, if it's running or not.

Command: Get-Service Winmgmt



```
PS C:\Users\Administrator> Get-Service Winmgmt

Status   Name               DisplayName
------   ----               -----------
Running  Winmgmt            Windows Management Instrumentation


PS C:\Users\Administrator> _
```

The Windows Management Instrumentation i.e WMI service is running.

We will be using the "**Get-WmiObject**" cmdlet to get WMI class information.

About Get-WmiObject:

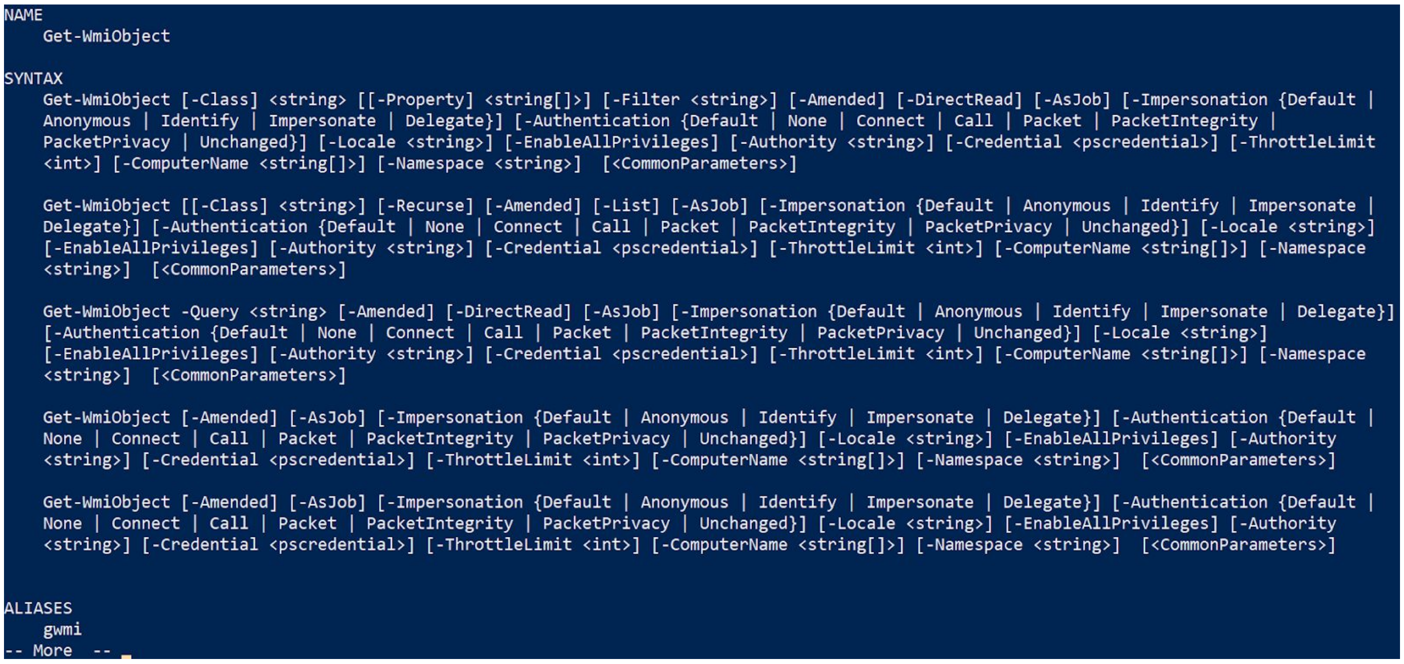
"The `Get-WmiObject` cmdlet gets instances of WMI classes or information about the available WMI classes. To specify a remote computer, use the ComputerName parameter. If the List parameter is specified, the cmdlet gets information about the WMI classes that are available in a specified namespace. If the Query parameter is specified, the cmdlet runs a WMI query language (WQL) statement.

The `Get-WmiObject` cmdlet does not use Windows PowerShell remoting to perform remote operations. You can use the ComputerName parameter of the `Get-WmiObject` cmdlet even if your computer does not meet the requirements for Windows PowerShell remoting or is not configured for remoting in Windows PowerShell."

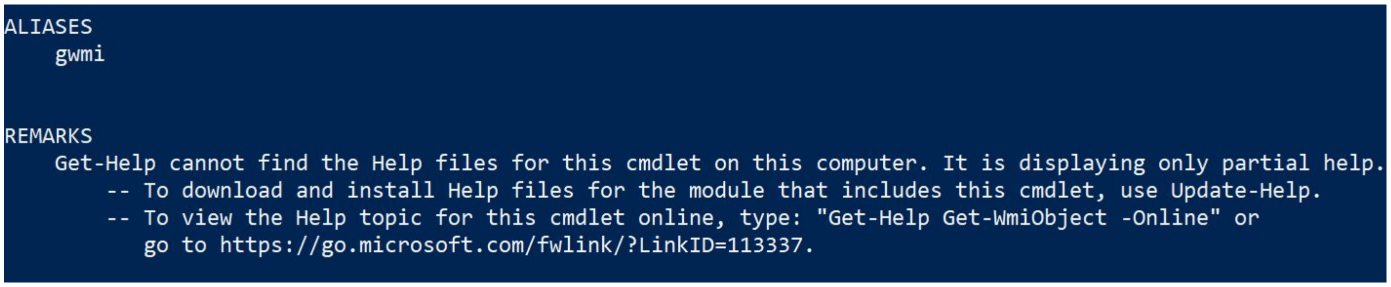
Source:
https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-wmiobject?view=powershell-5.1

Step 2: Check the help of the "Get-WmiObject" cmdlet

Command: help Get-WmiObject



```
NAME
    Get-WmiObject

SYNTAX
    Get-WmiObject [-Class] <string> [[-Property] <string[]>] [-Filter <string>] [-Amended] [-DirectRead] [-AsJob] [-Impersonation {Default |
    Anonymous | Identify | Impersonate | Delegate}] [-Authentication {Default | None | Connect | Call | Packet | PacketIntegrity |
    PacketPrivacy | Unchanged}] [-Locale <string>] [-EnableAllPrivileges] [-Authority <string>] [-Credential <pscredential>] [-ThrottleLimit
    <int>] [-ComputerName <string[]>] [-Namespace <string>]  [<CommonParameters>]

    Get-WmiObject [[-Class] <string>] [-Recurse] [-Amended] [-List] [-AsJob] [-Impersonation {Default | Anonymous | Identify | Impersonate |
    Delegate}] [-Authentication {Default | None | Connect | Call | Packet | PacketIntegrity | PacketPrivacy | Unchanged}] [-Locale <string>]
    [-EnableAllPrivileges] [-Authority <string>] [-Credential <pscredential>] [-ThrottleLimit <int>] [-ComputerName <string[]>] [-Namespace
    <string>]  [<CommonParameters>]

    Get-WmiObject -Query <string> [-Amended] [-DirectRead] [-AsJob] [-Impersonation {Default | Anonymous | Identify | Impersonate | Delegate}]
    [-Authentication {Default | None | Connect | Call | Packet | PacketIntegrity | PacketPrivacy | Unchanged}] [-Locale <string>]
    [-EnableAllPrivileges] [-Authority <string>] [-Credential <pscredential>] [-ThrottleLimit <int>] [-ComputerName <string[]>] [-Namespace
    <string>]  [<CommonParameters>]

    Get-WmiObject [-Amended] [-AsJob] [-Impersonation {Default | Anonymous | Identify | Impersonate | Delegate}] [-Authentication {Default |
    None | Connect | Call | Packet | PacketIntegrity | PacketPrivacy | Unchanged}] [-Locale <string>] [-EnableAllPrivileges] [-Authority
    <string>] [-Credential <pscredential>] [-ThrottleLimit <int>] [-ComputerName <string[]>] [-Namespace <string>]  [<CommonParameters>]

    Get-WmiObject [-Amended] [-AsJob] [-Impersonation {Default | Anonymous | Identify | Impersonate | Delegate}] [-Authentication {Default |
    None | Connect | Call | Packet | PacketIntegrity | PacketPrivacy | Unchanged}] [-Locale <string>] [-EnableAllPrivileges] [-Authority
    <string>] [-Credential <pscredential>] [-ThrottleLimit <int>] [-ComputerName <string[]>] [-Namespace <string>]  [<CommonParameters>]


ALIASES
    gwmi
-- More  --
```



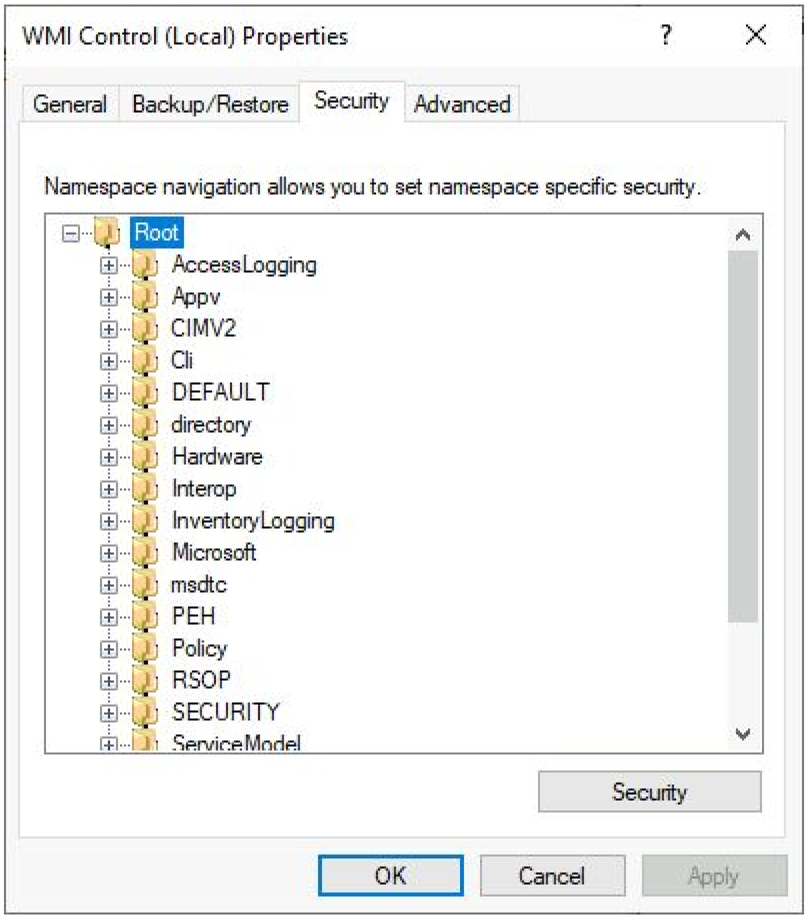
```
ALIASES
    gwmi


REMARKS
    Get-Help cannot find the Help files for this cmdlet on this computer. It is displaying only partial help.
        -- To download and install Help files for the module that includes this cmdlet, use Update-Help.
        -- To view the Help topic for this cmdlet online, type: "Get-Help Get-WmiObject -Online" or
           go to https://go.microsoft.com/fwlink/?LinkID=113337.
```

We have received all the syntax, aliases, and remarks related information.

Before we start running WMI commands, there are several terms that are quite important to know for the understanding of WMI.

- Namespace
- Classes
- Method
- Query

Namespace - WMI is divided into a directory-style hierarchy, the \root container, with other directories under \root. These "**directory paths**" are called namespaces.



Namespace navigation allows you to set namespace specific security.

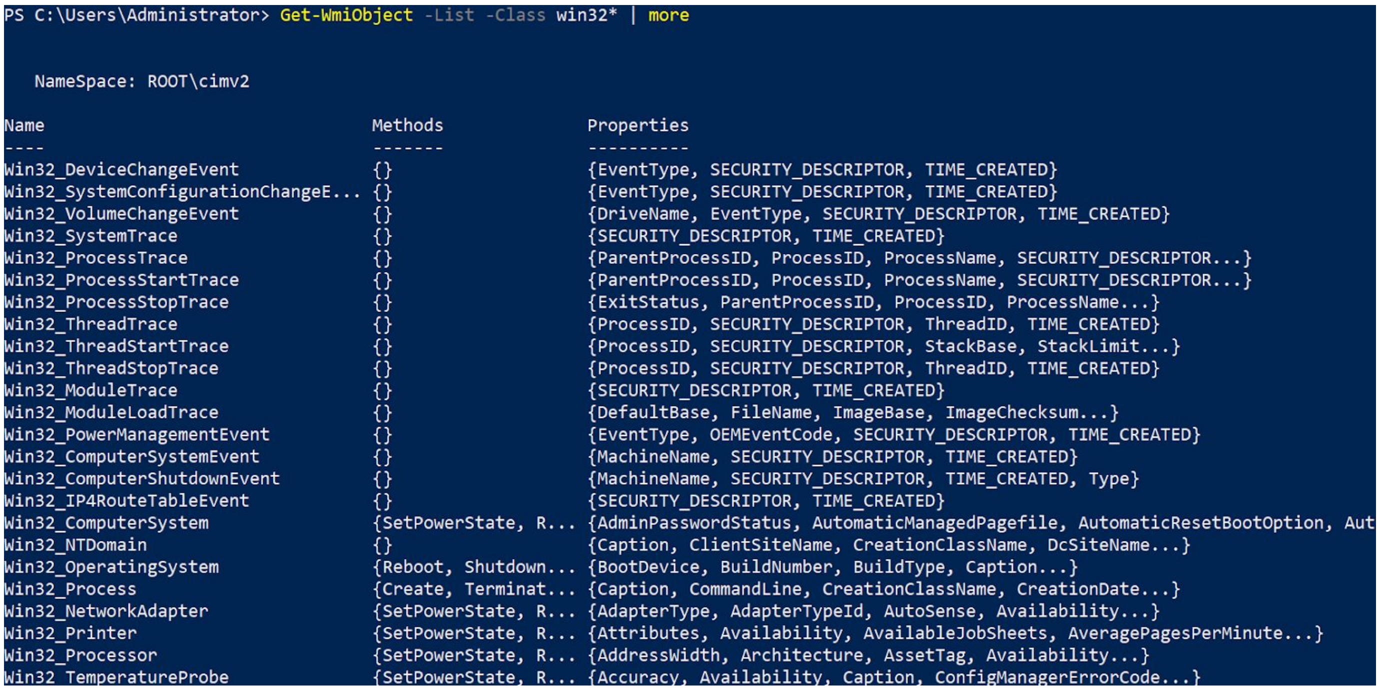
```
Root
  AccessLogging
  Appv
  CIMV2
  Cli
  DEFAULT
  directory
  Hardware
  Interop
  InventoryLogging
  Microsoft
  msdtc
  PEH
  Policy
  RSOP
  SECURITY
  ServiceModel
```

WMI Namespaces

Classes - The WMI class name eg: **win32_process** is a starting point for any WMI action. We always need to know a **Class Name** and the **Namespace** where it is located.

Use the following command to list all the classes in the CIMv2 namespace, having a name starting with "win32":

Command: Get-WmiObject -List -class win32* | more



```
PS C:\Users\Administrator> Get-WmiObject -List -Class win32* | more


   NameSpace: ROOT\cimv2

Name                                Methods              Properties
----                                -------              ----------
Win32_DeviceChangeEvent             {}                   {EventType, SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_SystemConfigurationChangeE... {}                   {EventType, SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_VolumeChangeEvent             {}                   {DriveName, EventType, SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_SystemTrace                   {}                   {SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_ProcessTrace                  {}                   {ParentProcessID, ProcessID, ProcessName, SECURITY_DESCRIPTOR...}
Win32_ProcessStartTrace             {}                   {ParentProcessID, ProcessID, ProcessName, SECURITY_DESCRIPTOR...}
Win32_ProcessStopTrace              {}                   {ExitStatus, ParentProcessID, ProcessID, ProcessName...}
Win32_ThreadTrace                   {}                   {ProcessID, SECURITY_DESCRIPTOR, ThreadID, TIME_CREATED}
Win32_ThreadStartTrace              {}                   {ProcessID, SECURITY_DESCRIPTOR, StackBase, StackLimit...}
Win32_ThreadStopTrace               {}                   {ProcessID, SECURITY_DESCRIPTOR, ThreadID, TIME_CREATED}
Win32_ModuleTrace                   {}                   {SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_ModuleLoadTrace               {}                   {DefaultBase, FileName, ImageBase, ImageChecksum...}
Win32_PowerManagementEvent          {}                   {EventType, OEMEventCode, SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_ComputerSystemEvent           {}                   {MachineName, SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_ComputerShutdownEvent         {}                   {MachineName, SECURITY_DESCRIPTOR, TIME_CREATED, Type}
Win32_IP4RouteTableEvent            {}                   {SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_ComputerSystem                {SetPowerState, R... {AdminPasswordStatus, AutomaticManagedPagefile, AutomaticResetBootOption, Aut
Win32_NTDomain                      {}                   {Caption, ClientSiteName, CreationClassName, DcSiteName...}
Win32_OperatingSystem               {Reboot, Shutdown... {BootDevice, BuildNumber, BuildType, Caption...}
Win32_Process                       {Create, Terminat... {Caption, CommandLine, CreationClassName, CreationDate...}
Win32_NetworkAdapter                {SetPowerState, R... {AdapterType, AdapterTypeId, AutoSense, Availability...}
Win32_Printer                       {SetPowerState, R... {Attributes, Availability, AvailableJobSheets, AveragePagesPerMinute...}
Win32_Processor                     {SetPowerState, R... {AddressWidth, Architecture, AssetTag, Availability...}
Win32_TemperatureProbe              {SetPowerState, R... {Accuracy, Availability, Caption, ConfigManagerErrorCode...}
```

CIMV2 Namespace Classes

Method - WMI classes have one or more functions that can be executed. These functions are called methods.

Query - A WMI Query Language (WQL) statement to run.

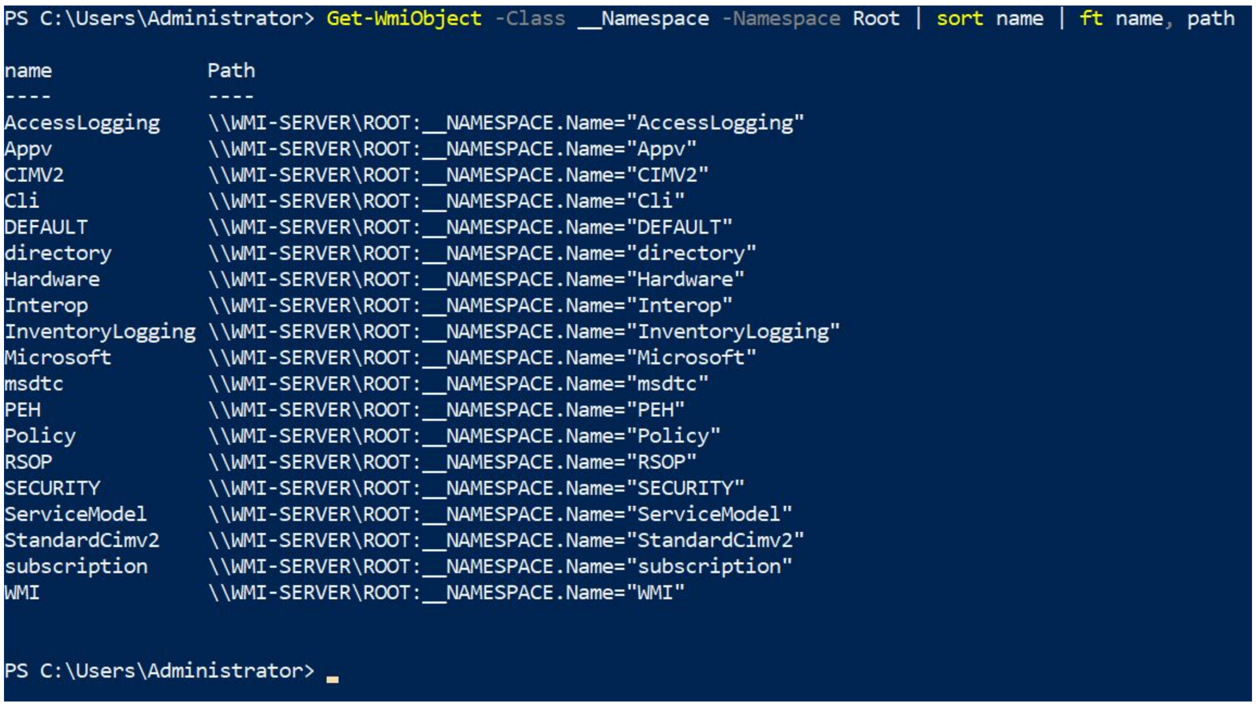
We are now familiar with several terms of the WMI.

Running WMI Commands by Get-WmiObject cmdlets

Step 1: Getting the list of namespaces

Command: Get-WmiObject -Class __Namespace -Namespace Root | sort name | ft name, path

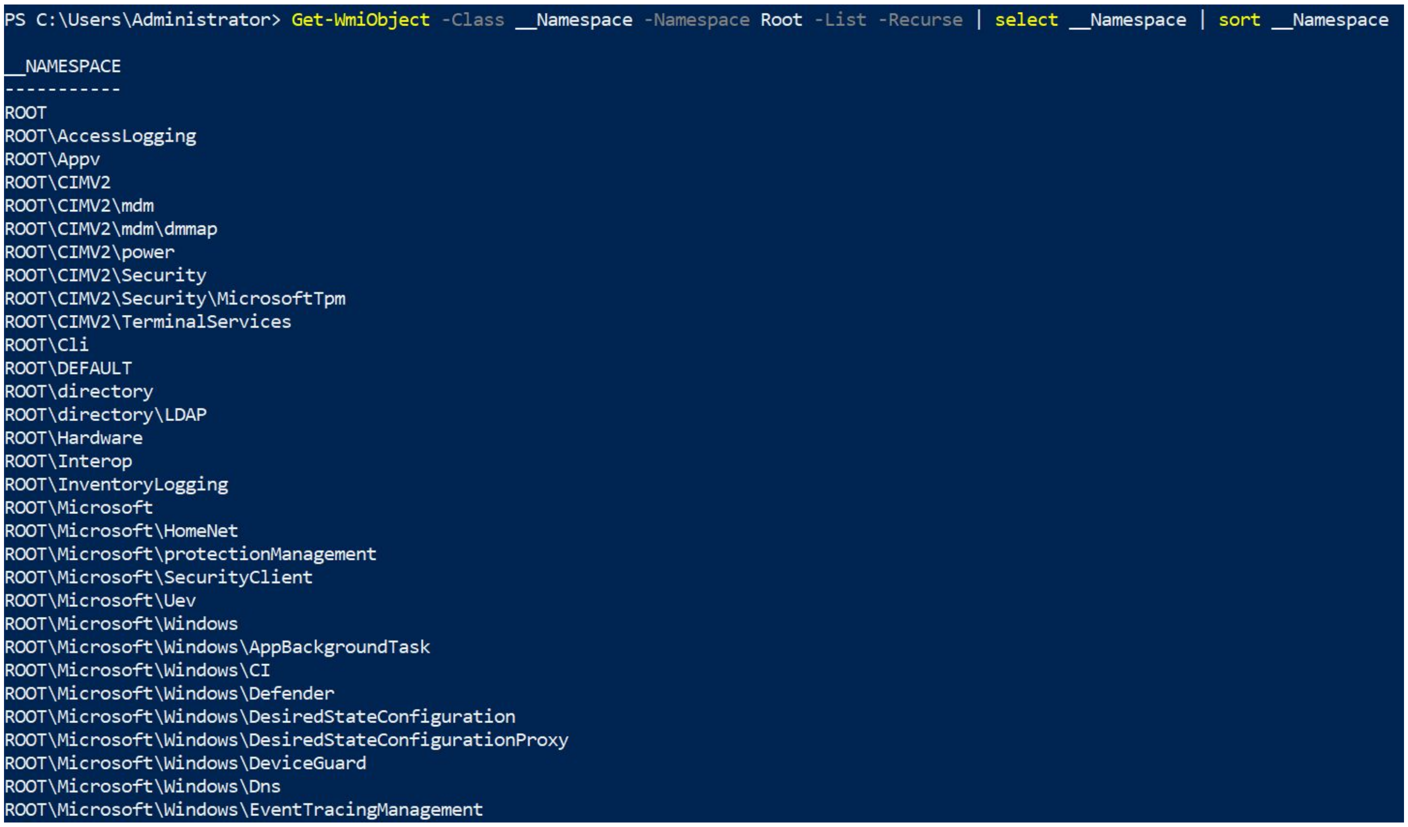
Note: If we don't define "-Namespace root" by default it would pick up the "Root/CIMV2" namespace.



```
PS C:\Users\Administrator> Get-WmiObject -Class __Namespace -Namespace Root | sort name | ft name, path

name             Path
----             ----
AccessLogging    \\WMI-SERVER\ROOT:__NAMESPACE.Name="AccessLogging"
Appv             \\WMI-SERVER\ROOT:__NAMESPACE.Name="Appv"
CIMV2            \\WMI-SERVER\ROOT:__NAMESPACE.Name="CIMV2"
Cli              \\WMI-SERVER\ROOT:__NAMESPACE.Name="Cli"
DEFAULT          \\WMI-SERVER\ROOT:__NAMESPACE.Name="DEFAULT"
directory        \\WMI-SERVER\ROOT:__NAMESPACE.Name="directory"
Hardware         \\WMI-SERVER\ROOT:__NAMESPACE.Name="Hardware"
Interop          \\WMI-SERVER\ROOT:__NAMESPACE.Name="Interop"
InventoryLogging \\WMI-SERVER\ROOT:__NAMESPACE.Name="InventoryLogging"
Microsoft        \\WMI-SERVER\ROOT:__NAMESPACE.Name="Microsoft"
msdtc            \\WMI-SERVER\ROOT:__NAMESPACE.Name="msdtc"
PEH              \\WMI-SERVER\ROOT:__NAMESPACE.Name="PEH"
Policy           \\WMI-SERVER\ROOT:__NAMESPACE.Name="Policy"
RSOP             \\WMI-SERVER\ROOT:__NAMESPACE.Name="RSOP"
SECURITY         \\WMI-SERVER\ROOT:__NAMESPACE.Name="SECURITY"
ServiceModel     \\WMI-SERVER\ROOT:__NAMESPACE.Name="ServiceModel"
StandardCimv2    \\WMI-SERVER\ROOT:__NAMESPACE.Name="StandardCimv2"
subscription     \\WMI-SERVER\ROOT:__NAMESPACE.Name="subscription"
WMI              \\WMI-SERVER\ROOT:__NAMESPACE.Name="WMI"


PS C:\Users\Administrator> _
```

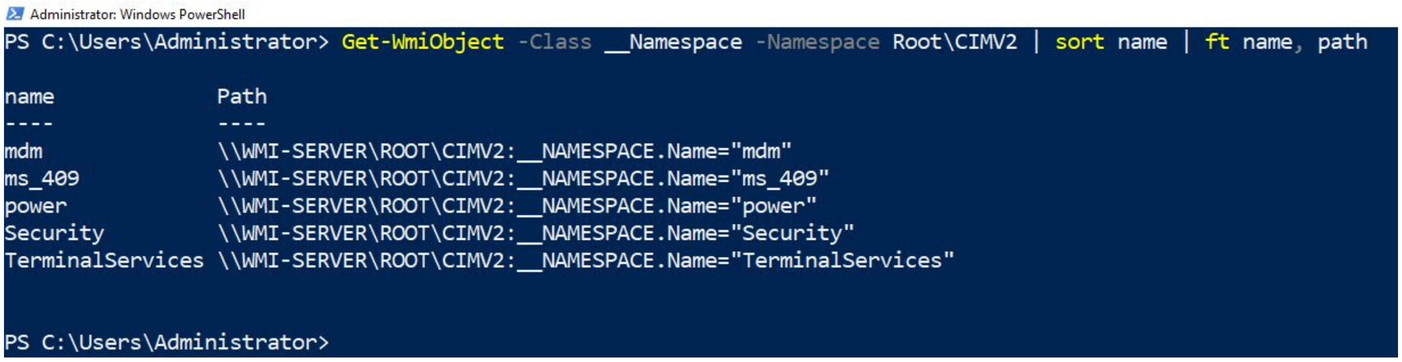
More in-depth namespaces

Command: Get-WmiObject -Class __Namespace -Namespace Root -List -Recurse | select __Namespace | sort __Namespace


```
PS C:\Users\Administrator> Get-WmiObject -Class __Namespace -Namespace Root -List -Recurse | select __Namespace | sort __Namespace

__NAMESPACE
-----------
ROOT
ROOT\AccessLogging
ROOT\Appv
ROOT\CIMV2
ROOT\CIMV2\mdm
ROOT\CIMV2\mdm\dmmap
ROOT\CIMV2\power
ROOT\CIMV2\Security
ROOT\CIMV2\Security\MicrosoftTpm
ROOT\CIMV2\TerminalServices
ROOT\Cli
ROOT\DEFAULT
ROOT\directory
ROOT\directory\LDAP
ROOT\Hardware
ROOT\Interop
ROOT\InventoryLogging
ROOT\Microsoft
ROOT\Microsoft\HomeNet
ROOT\Microsoft\protectionManagement
ROOT\Microsoft\SecurityClient
ROOT\Microsoft\Uev
ROOT\Microsoft\Windows
ROOT\Microsoft\Windows\AppBackgroundTask
ROOT\Microsoft\Windows\CI
ROOT\Microsoft\Windows\Defender
ROOT\Microsoft\Windows\DesiredStateConfiguration
ROOT\Microsoft\Windows\DesiredStateConfigurationProxy
ROOT\Microsoft\Windows\DeviceGuard
ROOT\Microsoft\Windows\Dns
ROOT\Microsoft\Windows\EventTracingManagement
```

Step 2: Check namespaces for Root/CIMV2 only

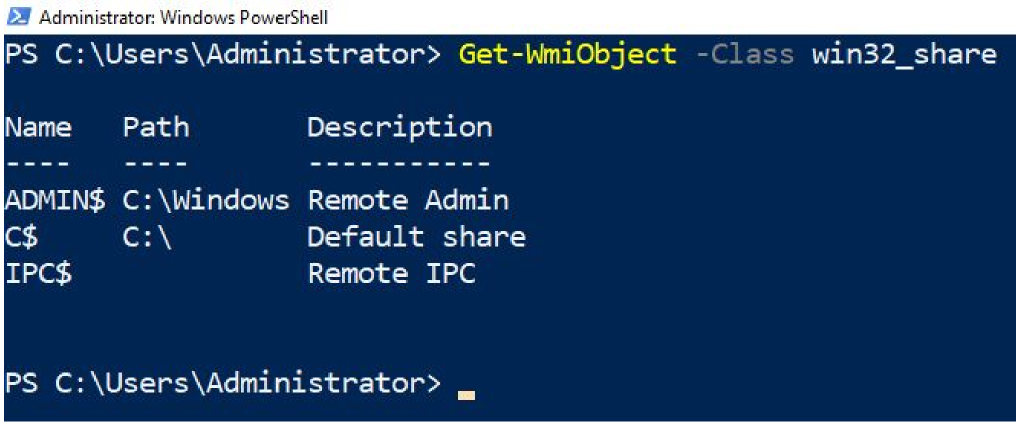
Command: Get-WmiObject -Class __Namespace -Namespace Root\CIMV2 | sort name | ft name, path



```
PS C:\Users\Administrator> Get-WmiObject -Class __Namespace -Namespace Root\CIMV2 | sort name | ft name, path

name             Path
----             ----
mdm              \\WMI-SERVER\ROOT\CIMV2:__NAMESPACE.Name="mdm"
ms_409           \\WMI-SERVER\ROOT\CIMV2:__NAMESPACE.Name="ms_409"
power            \\WMI-SERVER\ROOT\CIMV2:__NAMESPACE.Name="power"
Security         \\WMI-SERVER\ROOT\CIMV2:__NAMESPACE.Name="Security"
TerminalServices \\WMI-SERVER\ROOT\CIMV2:__NAMESPACE.Name="TerminalServices"


PS C:\Users\Administrator>
```

Step 3: Invoke win32_share class.

Command: Get-WmiObject -Class win32_share



Administrator: Windows PowerShell

```
PS C:\Users\Administrator> Get-WmiObject -Class win32_share

Name   Path       Description
----   ----       -----------
ADMIN$ C:\Windows Remote Admin
C$     C:\        Default share
IPC$              Remote IPC


PS C:\Users\Administrator> _
```

We can notice, we have received all the shares as an output. Also, when we don't define any specific namespace then it is by default using **Root/CIMV2** namespace. The **win32_share** class lives in **Root/CIMV2** namespace.

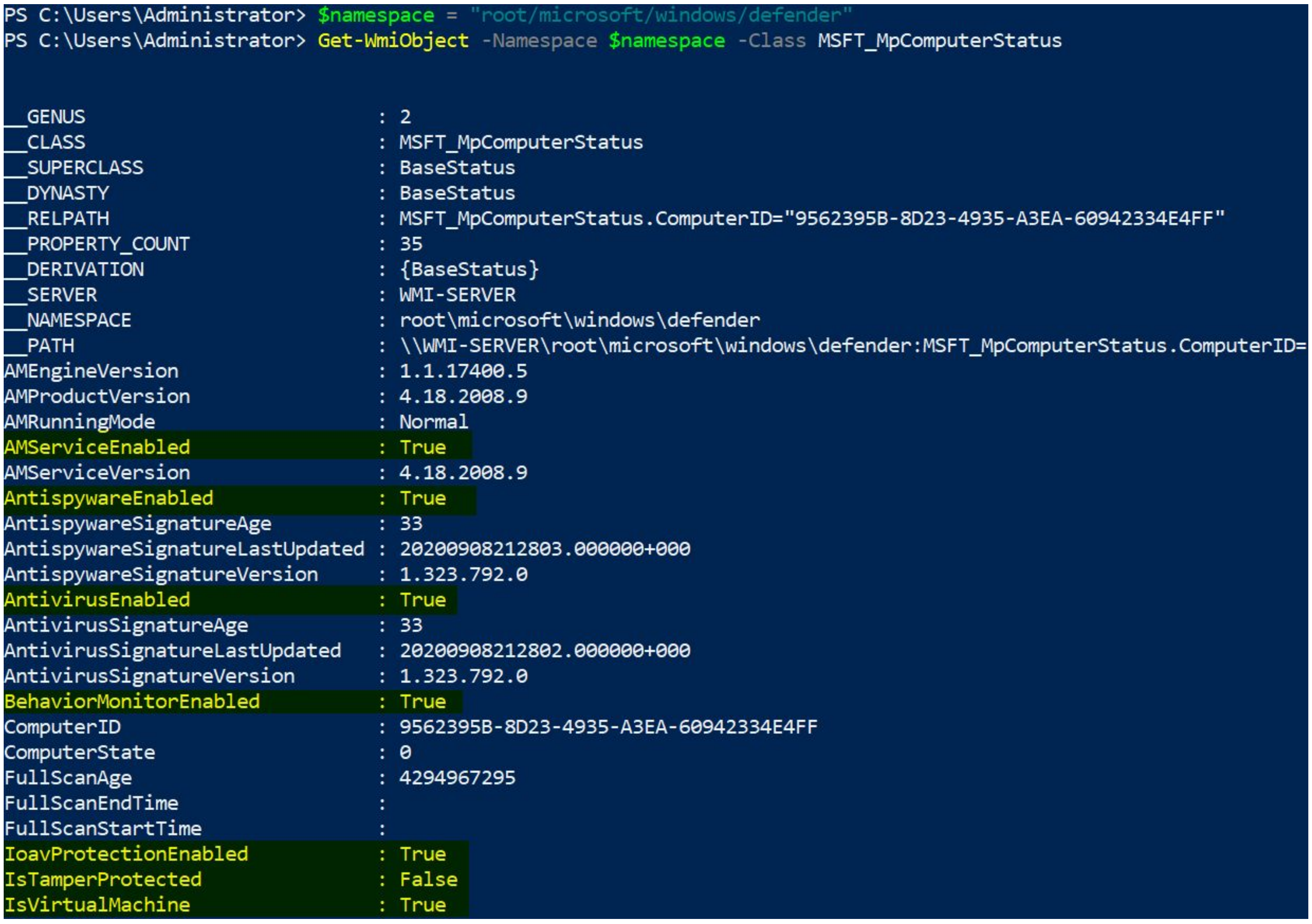
Step 4: Specify the variable as namespace and use the variable to run the commands.

Command: $namespace = "root/microsoft/windows/defender"

We are using "root/microsoft/windows/defender" namespace to query Windows Defender.

Checking the status of the Windows Defender.

Command: Get-WmiObject -Namespace $namespace -Class MSFT_MpComputerStatus

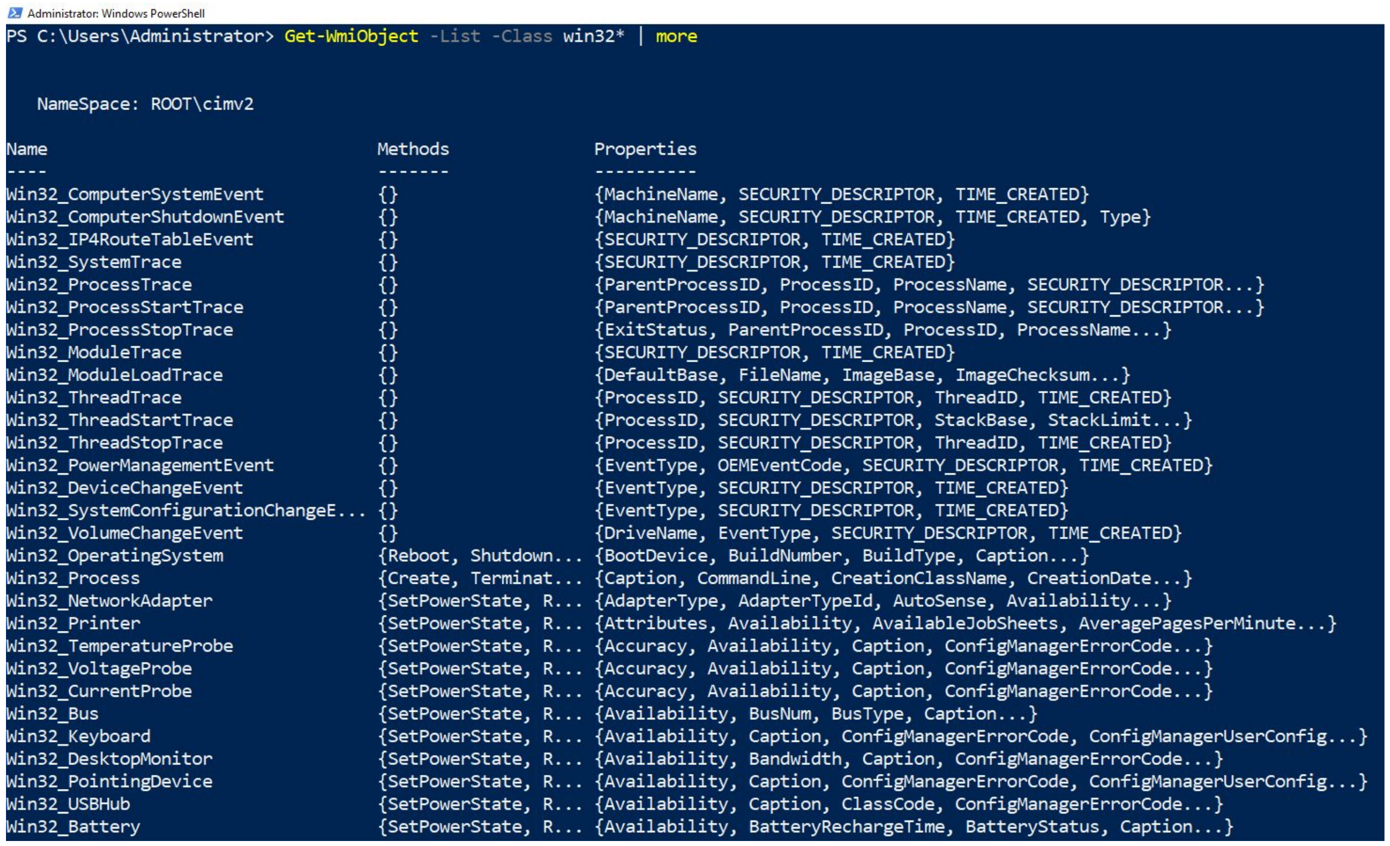


```
PS C:\Users\Administrator> $namespace = "root/microsoft/windows/defender"
PS C:\Users\Administrator> Get-WmiObject -Namespace $namespace -Class MSFT_MpComputerStatus


__GENUS                          : 2
__CLASS                          : MSFT_MpComputerStatus
__SUPERCLASS                     : BaseStatus
__DYNASTY                        : BaseStatus
__RELPATH                        : MSFT_MpComputerStatus.ComputerID="9562395B-8D23-4935-A3EA-60942334E4FF"
__PROPERTY_COUNT                 : 35
__DERIVATION                     : {BaseStatus}
__SERVER                         : WMI-SERVER
__NAMESPACE                      : root\microsoft\windows\defender
__PATH                           : \\WMI-SERVER\root\microsoft\windows\defender:MSFT_MpComputerStatus.ComputerID=
AMEngineVersion                  : 1.1.17400.5
AMProductVersion                 : 4.18.2008.9
AMRunningMode                    : Normal
AMServiceEnabled                 : True
AMServiceVersion                 : 4.18.2008.9
AntispywareEnabled               : True
AntispywareSignatureAge          : 33
AntispywareSignatureLastUpdated  : 20200908212803.000000+000
AntispywareSignatureVersion      : 1.323.792.0
AntivirusEnabled                 : True
AntivirusSignatureAge            : 33
AntivirusSignatureLastUpdated    : 20200908212802.000000+000
AntivirusSignatureVersion        : 1.323.792.0
BehaviorMonitorEnabled           : True
ComputerID                       : 9562395B-8D23-4935-A3EA-60942334E4FF
ComputerState                    : 0
FullScanAge                      : 4294967295
FullScanEndTime                  :
FullScanStartTime                :
IoavProtectionEnabled            : True
IsTamperProtected                : False
IsVirtualMachine                 : True
```

Windows Defender is up and running.

Step 4: List all the class which starts from win32*

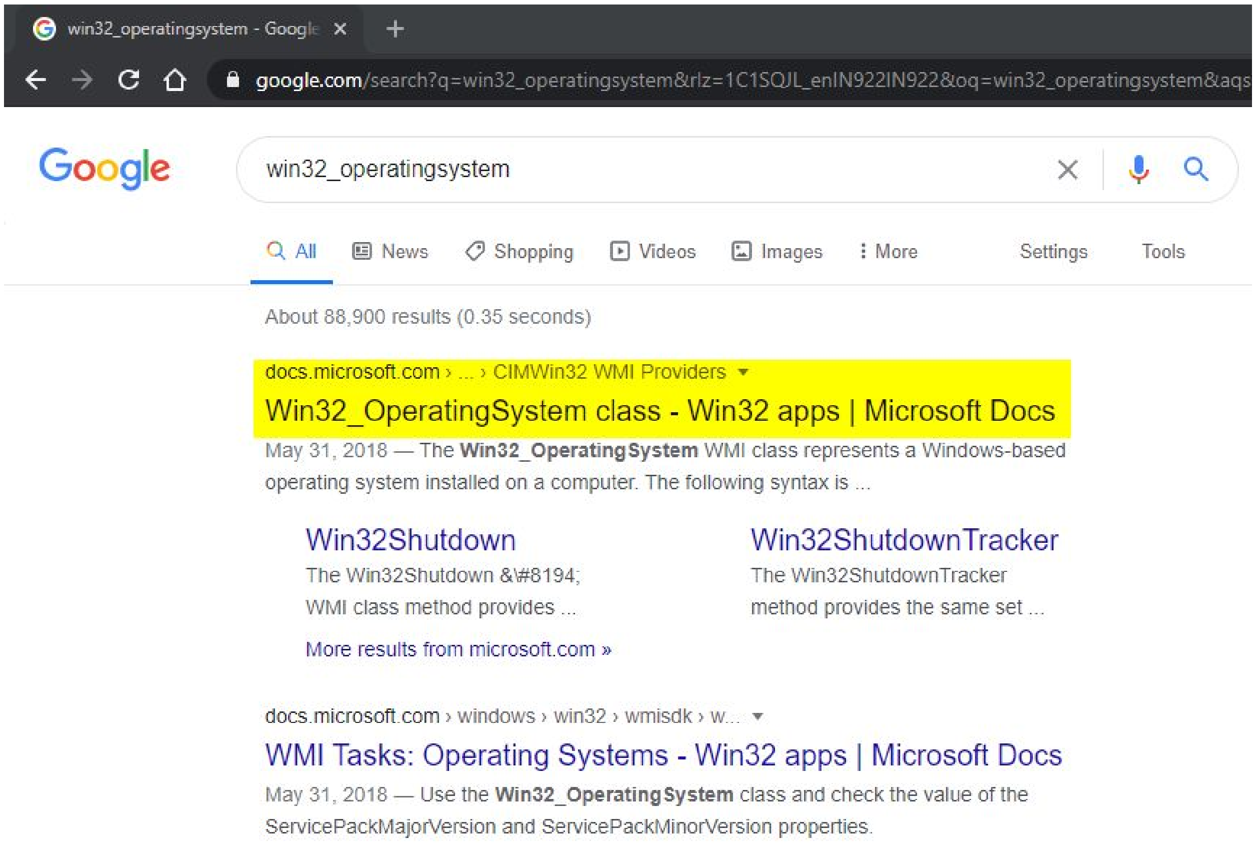
Command: Get-WmiObject -List -Class win32* | more



Administrator: Windows PowerShell

```
PS C:\Users\Administrator> Get-WmiObject -List -Class win32* | more


   NameSpace: ROOT\cimv2

Name                                Methods              Properties
----                                -------              ----------
Win32_ComputerSystemEvent           {}                   {MachineName, SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_ComputerShutdownEvent         {}                   {MachineName, SECURITY_DESCRIPTOR, TIME_CREATED, Type}
Win32_IP4RouteTableEvent            {}                   {SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_SystemTrace                   {}                   {SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_ProcessTrace                  {}                   {ParentProcessID, ProcessID, ProcessName, SECURITY_DESCRIPTOR...}
Win32_ProcessStartTrace             {}                   {ParentProcessID, ProcessID, ProcessName, SECURITY_DESCRIPTOR...}
Win32_ProcessStopTrace              {}                   {ExitStatus, ParentProcessID, ProcessID, ProcessName...}
Win32_ModuleTrace                   {}                   {SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_ModuleLoadTrace               {}                   {DefaultBase, FileName, ImageBase, ImageChecksum...}
Win32_ThreadTrace                   {}                   {ProcessID, SECURITY_DESCRIPTOR, ThreadID, TIME_CREATED}
Win32_ThreadStartTrace              {}                   {ProcessID, SECURITY_DESCRIPTOR, StackBase, StackLimit...}
Win32_ThreadStopTrace               {}                   {ProcessID, SECURITY_DESCRIPTOR, ThreadID, TIME_CREATED}
Win32_PowerManagementEvent          {}                   {EventType, OEMEventCode, SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_DeviceChangeEvent             {}                   {EventType, SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_SystemConfigurationChangeE... {}                   {EventType, SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_VolumeChangeEvent             {}                   {DriveName, EventType, SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_OperatingSystem               {Reboot, Shutdown... {BootDevice, BuildNumber, BuildType, Caption...}
Win32_Process                       {Create, Terminat... {Caption, CommandLine, CreationClassName, CreationDate...}
Win32_NetworkAdapter                {SetPowerState, R... {AdapterType, AdapterTypeId, AutoSense, Availability...}
Win32_Printer                       {SetPowerState, R... {Attributes, Availability, AvailableJobSheets, AveragePagesPerMinute...}
Win32_TemperatureProbe              {SetPowerState, R... {Accuracy, Availability, Caption, ConfigManagerErrorCode...}
Win32_VoltageProbe                  {SetPowerState, R... {Accuracy, Availability, Caption, ConfigManagerErrorCode...}
Win32_CurrentProbe                  {SetPowerState, R... {Accuracy, Availability, Caption, ConfigManagerErrorCode...}
Win32_Bus                           {SetPowerState, R... {Availability, BusNum, BusType, Caption...}
Win32_Keyboard                      {SetPowerState, R... {Availability, Caption, ConfigManagerErrorCode, ConfigManagerUserConfig...}
Win32_DesktopMonitor                {SetPowerState, R... {Availability, Bandwidth, Caption, ConfigManagerErrorCode...}
Win32_PointingDevice                {SetPowerState, R... {Availability, Caption, ConfigManagerErrorCode, ConfigManagerUserConfig...}
Win32_USBHub                        {SetPowerState, R... {Availability, Caption, ClassCode, ConfigManagerErrorCode...}
Win32_Battery                       {SetPowerState, R... {Availability, BatteryRechargeTime, BatteryStatus, Caption...}
```

We have received all the classes which start from the **win32***. In this case, we are interested in **win32_operatingsystem** class. Also, we could always search for particular WMI classes online. eg: Search for “**win32_operatingsystem**” on google.


win32_operatingsystem - Google

google.com/search?q=win32_operatingsystem&rlz=1C1SQJL_enIN922IN922&oq=win32_operatingsystem&aqs

win32_operatingsystem

All · News · Shopping · Videos · Images · More · Settings · Tools

About 88,900 results (0.35 seconds)

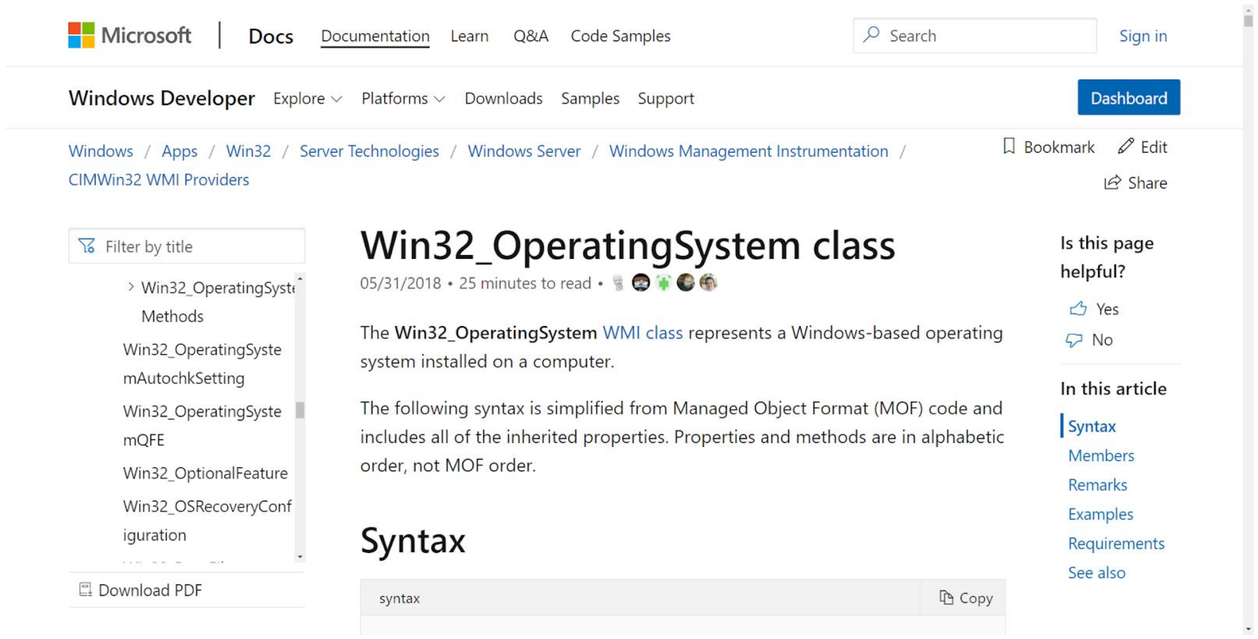
docs.microsoft.com › ... › CIMWin32 WMI Providers
Win32_OperatingSystem class - Win32 apps | Microsoft Docs
May 31, 2018 — The **Win32_OperatingSystem** WMI class represents a Windows-based operating system installed on a computer. The following syntax is ...

Win32Shutdown
The Win32Shutdown &\#8194; WMI class method provides ...

Win32ShutdownTracker
The Win32ShutdownTracker method provides the same set ...

More results from microsoft.com »

docs.microsoft.com › windows › win32 › wmisdk › w...
WMI Tasks: Operating Systems - Win32 apps | Microsoft Docs
May 31, 2018 — Use the **Win32_OperatingSystem** class and check the value of the ServicePackMajorVersion and ServicePackMinorVersion properties.


Microsoft | Docs Documentation Learn Q&A Code Samples Search Sign in

Windows Developer Explore Platforms Downloads Samples Support Dashboard

Windows / Apps / Win32 / Server Technologies / Windows Server / Windows Management Instrumentation / CIMWin32 WMI Providers

Bookmark Edit Share

Filter by title
- Win32_OperatingSystem Methods
- Win32_OperatingSystemAutochkSetting
- Win32_OperatingSystemQFE
- Win32_OptionalFeature
- Win32_OSRecoveryConfiguration

Download PDF

Win32_OperatingSystem class

05/31/2018 • 25 minutes to read •

The Win32_OperatingSystem WMI class represents a Windows-based operating system installed on a computer.

The following syntax is simplified from Managed Object Format (MOF) code and includes all of the inherited properties. Properties and methods are in alphabetic order, not MOF order.

Syntax

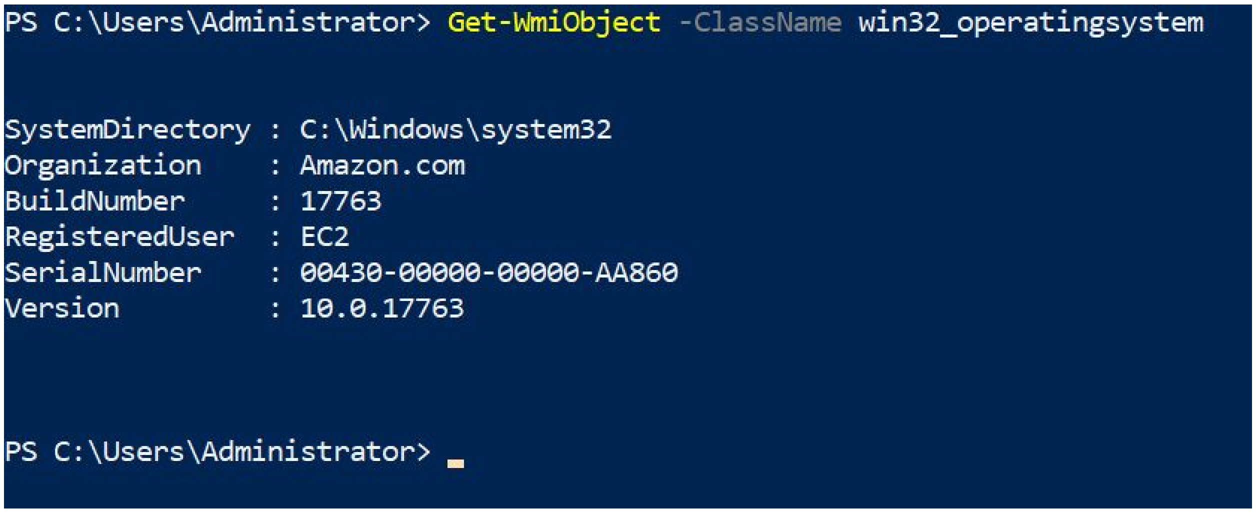
syntax Copy

Is this page helpful? Yes No

In this article: Syntax, Members, Remarks, Examples, Requirements, See also

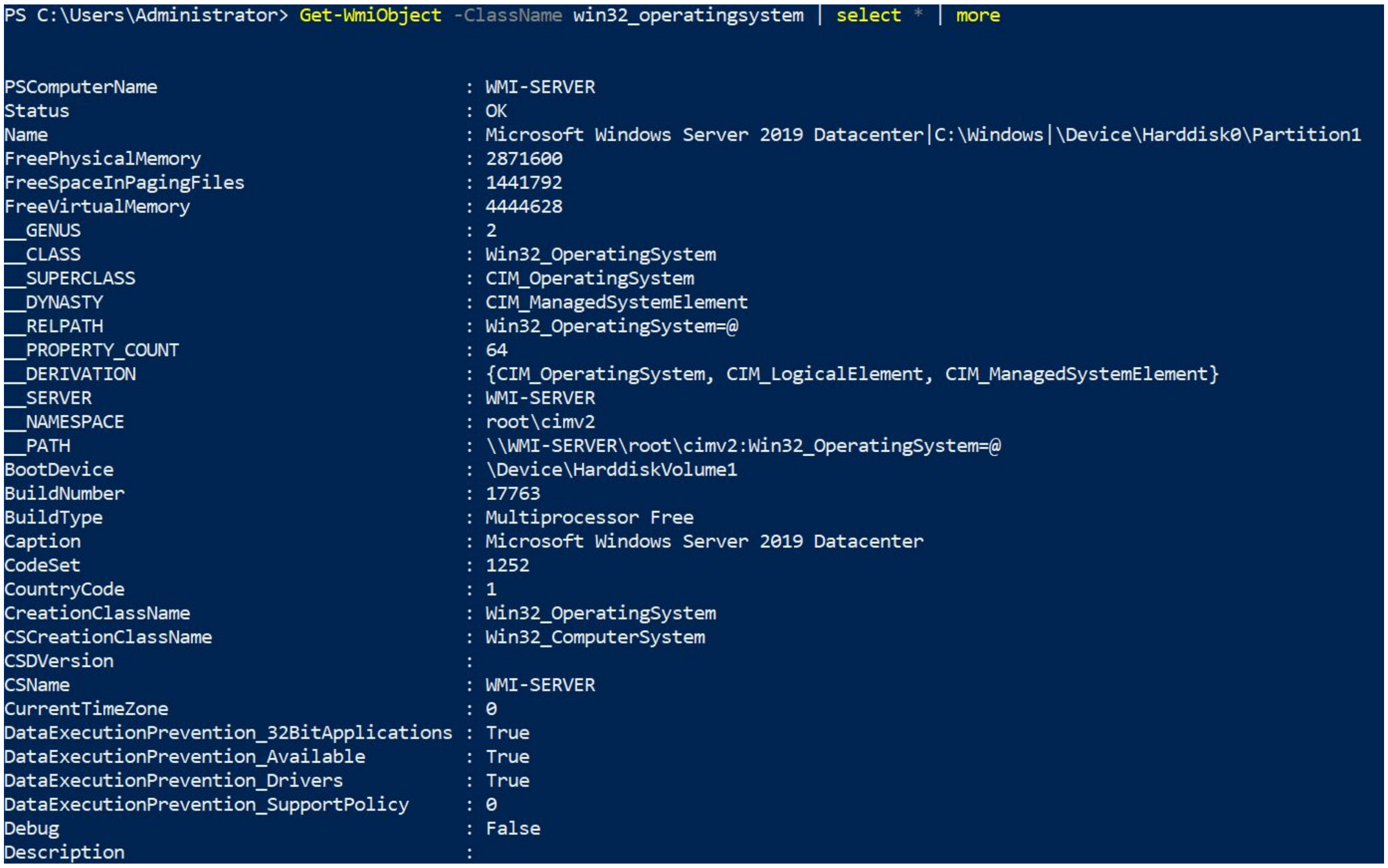
Running Win32_OperatingSystem class.

Command: Get-WmiObject -ClassName win32_operatingsystem


```
PS C:\Users\Administrator> Get-WmiObject -ClassName win32_operatingsystem


SystemDirectory : C:\Windows\system32
Organization    : Amazon.com
BuildNumber     : 17763
RegisteredUser  : EC2
SerialNumber    : 00430-00000-00000-AA860
Version         : 10.0.17763



PS C:\Users\Administrator>
```

We can run the below command to get in-depth details of the class.

Command: Get-WmiObject -ClassName win32_operatingsystem | select * | more

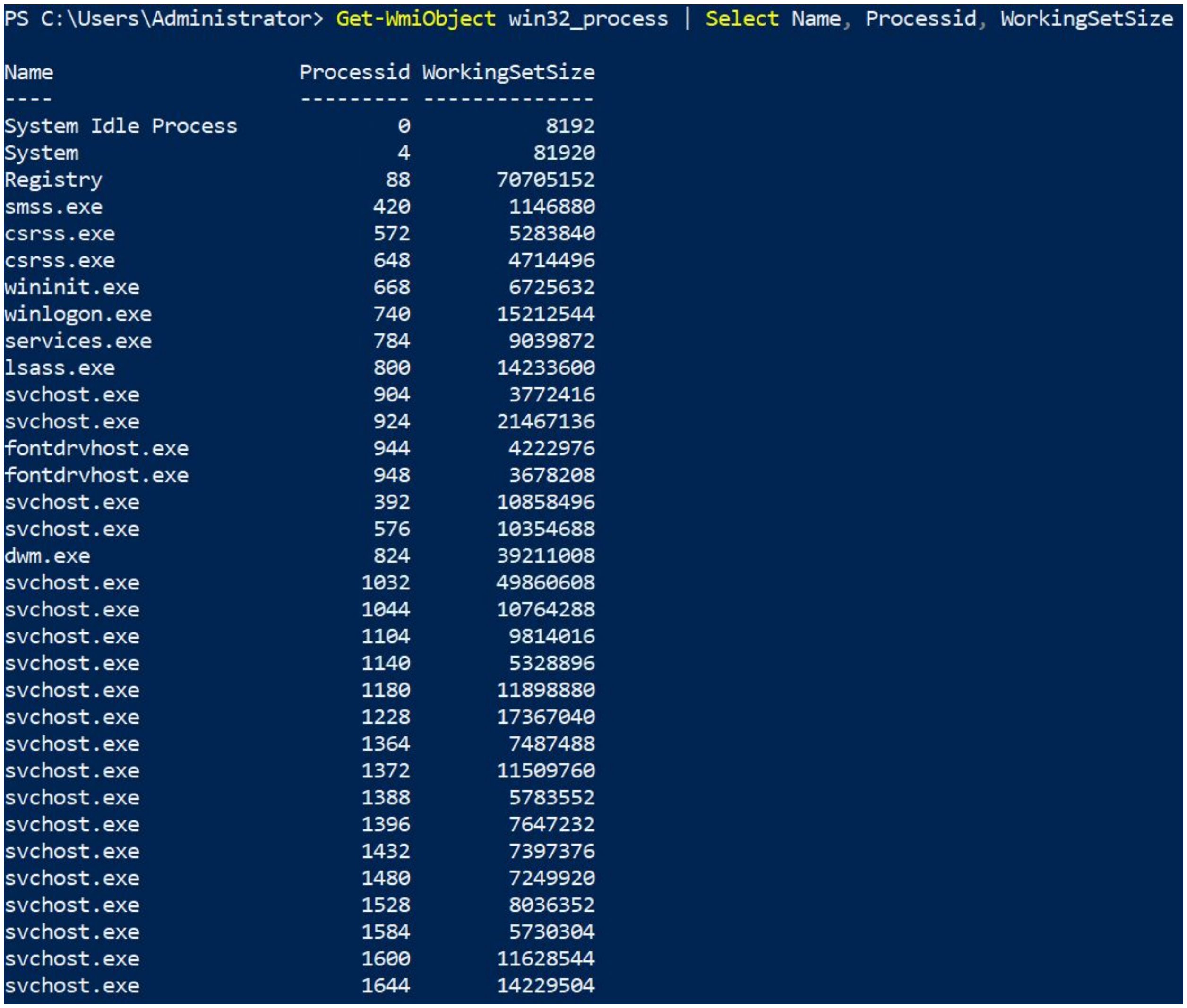


```
PS C:\Users\Administrator> Get-WmiObject -ClassName win32_operatingsystem | select * | more


PSComputerName                            : WMI-SERVER
Status                                    : OK
Name                                      : Microsoft Windows Server 2019 Datacenter|C:\Windows|\Device\Harddisk0\Partition1
FreePhysicalMemory                        : 2871600
FreeSpaceInPagingFiles                    : 1441792
FreeVirtualMemory                         : 4444628
__GENUS                                   : 2
__CLASS                                   : Win32_OperatingSystem
__SUPERCLASS                              : CIM_OperatingSystem
__DYNASTY                                 : CIM_ManagedSystemElement
__RELPATH                                 : Win32_OperatingSystem=@
__PROPERTY_COUNT                          : 64
__DERIVATION                              : {CIM_OperatingSystem, CIM_LogicalElement, CIM_ManagedSystemElement}
__SERVER                                  : WMI-SERVER
__NAMESPACE                               : root\cimv2
__PATH                                    : \\WMI-SERVER\root\cimv2:Win32_OperatingSystem=@
BootDevice                                : \Device\HarddiskVolume1
BuildNumber                               : 17763
BuildType                                 : Multiprocessor Free
Caption                                   : Microsoft Windows Server 2019 Datacenter
CodeSet                                   : 1252
CountryCode                               : 1
CreationClassName                         : Win32_OperatingSystem
CSCreationClassName                       : Win32_ComputerSystem
CSDVersion                                :
CSName                                    : WMI-SERVER
CurrentTimeZone                           : 0
DataExecutionPrevention_32BitApplications : True
DataExecutionPrevention_Available         : True
DataExecutionPrevention_Drivers           : True
DataExecutionPrevention_SupportPolicy     : 0
Debug                                     : False
Description                               :
```

We have learned about WMI namespaces and their classes

Step 5: Get all the running processes.

Command: Get-WmiObject win32_process | Select Name, Processid, WorkingSetSize

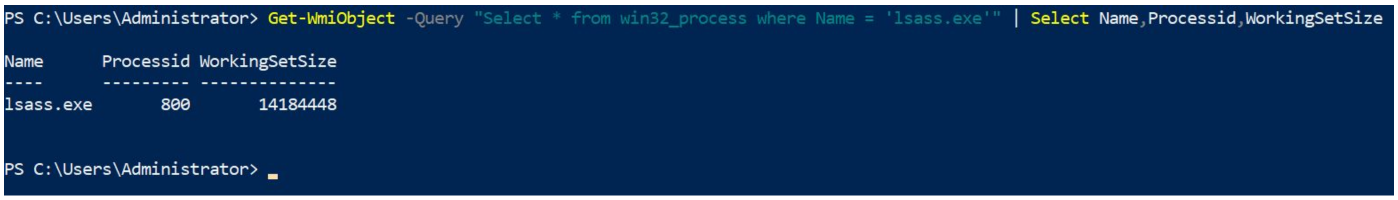


```
PS C:\Users\Administrator> Get-WmiObject win32_process | Select Name, Processid, WorkingSetSize

Name                    Processid WorkingSetSize
----                    --------- --------------
System Idle Process             0           8192
System                          4          81920
Registry                       88       70705152
smss.exe                      420        1146880
csrss.exe                     572        5283840
csrss.exe                     648        4714496
wininit.exe                   668        6725632
winlogon.exe                  740       15212544
services.exe                  784        9039872
lsass.exe                     800       14233600
svchost.exe                   904        3772416
svchost.exe                   924       21467136
fontdrvhost.exe               944        4222976
fontdrvhost.exe               948        3678208
svchost.exe                   392       10858496
svchost.exe                   576       10354688
dwm.exe                       824       39211008
svchost.exe                  1032       49860608
svchost.exe                  1044       10764288
svchost.exe                  1104        9814016
svchost.exe                  1140        5328896
svchost.exe                  1180       11898880
svchost.exe                  1228       17367040
svchost.exe                  1364        7487488
svchost.exe                  1372       11509760
svchost.exe                  1388        5783552
svchost.exe                  1396        7647232
svchost.exe                  1432        7397376
svchost.exe                  1480        7249920
svchost.exe                  1528        8036352
svchost.exe                  1584        5730304
svchost.exe                  1600       11628544
svchost.exe                  1644       14229504
```

We have received all the running processes. Also, we have filtered the processes with only Name, Process ID, and Working Size.

Step 6: Running WMI query to filter the lsass.exe process only.

Command: Get-WmiObject -Query "Select * from win32_process where Name = 'lsass.exe'" | Select Name,Processid,WorkingSetSize


```
PS C:\Users\Administrator> Get-WmiObject -Query "Select * from win32_process where Name = 'lsass.exe'" | Select Name,Processid,WorkingSetSize

Name      Processid WorkingSetSize
----      --------- --------------
lsass.exe       800       14184448


PS C:\Users\Administrator> _
```

We have successfully used multiple namespaces and classes. Similarly, we could use any namespace and their classes as per the goal.

The objective of this exercise is to get familiar with WMI Namespace and Classes for pentesting use cases. We can perform tons of things using the available WMI classes or custom classes.

References:

1. WMI Arch (https://docs.microsoft.com/en-us/windows/win32/wmisdk/wmi-architecture)
2. WMI Namespace (https://docs.microsoft.com/en-us/windows/win32/wmisdk/gloss-n)
3. WMI Glossary (https://docs.microsoft.com/en-us/windows/win32/wmisdk/wmi-glossary)