WMI i CIM. Praca z właściwościami i metodami – info

Praca z obiektami WMI i CIM

Listowanie właściwości i metod tych obiektów

Odczytywanie właściwości

Uruchamianie metod obiektów

Uruchamianie metod klas

Wyszukiwanie metod i właściwości w obiektach klasy z obszaru nazw root cim v2 do których z poziomu PowerShell uzyskujemy dostęp przez Get-WmiObject lub Get-CimClass

Poniżej lista właściwości opisująca klasę Win32_OperatingSystem

```
1   Get-WmiObject -Class Win32_OperatingSystem | Get-Member
```

```
   TypeName: System.Management.ManagementObject#root\cimv2\Win32_OperatingSystem

Name                                MemberType     Definition
----                                ----------     ----------
PSComputerName                      AliasProperty  PSComputerName = __SERVER
Reboot                              Method         System.Management.Managem...
SetDateTime                         Method         System.Management.Managem...
Shutdown                            Method         System.Management.Managem...
Win32Shutdown                       Method         System.Management.Managem...
Win32ShutdownTracker                Method         System.Management.Managem...
BootDevice                          Property       string BootDevice {get;set;}
```

Kiedy zainstalowano system operacyjny na tym komputerze

```
1   Get-WmiObject -Class Win32_OperatingSystem | Get-Member
```

```
FreeVirtualMemory                   Property       uint64 FreeVirtualMemory ...
InstallDate                         Property       string InstallDate {get;s...
LargeSystemCache                    Property       uint32 LargeSystemCache {...
```

Pytanie i odpowiedz

```
3   Get-WmiObject -Class Win32_OperatingSystem | Select-Object -Property InstallDate
```

```
InstallDate
-----------
20240626100425.000000+120
```

W sposób bardziej czytelny

$installDate = Get-WmiObject -Class Win32_OperatingSystem | Select-Object -ExpandProperty InstallDate

$dateTime = [System.Management.ManagementDateTimeConverter]::ToDateTime($installDate)

$formattedDateTime = $dateTime.ToString("yyyy-MM-dd HH:mm:ss")

Write-Host "Data instalacji systemu: $formattedDateTime"

```
PS C:\Windows\system32> Get-Date | ForEach-Object { Write-Host $_ }

05.07.2024 15:33:39

PS C:\Windows\system32> $installDate = Get-WmiObject -Class Win32_OperatingSystem | Select-Object -ExpandProperty InstallDate
$dateTime = [System.Management.ManagementDateTimeConverter]::ToDateTime($installDate)
$formattedDateTime = $dateTime.ToString("yyyy-MM-dd HH:mm:ss")

Write-Host "Formatted InstallDate: $formattedDateTime"

Formatted InstallDate: 2024-06-26 10:04:25
```

Polecenie, poniższe, jest używane do pobierania informacji o klasie WMI Win32_OperatingSystem i wyświetlania jej właściwości i metod przy użyciu Get-Member.

Get-CimClass -ClassName Win32_OperatingSystem | Get-Member ma znacznie mniej wyników

```
2  Get-CimClass -ClassName Win32_OperatingSystem | Get-Member
```

```
   TypeName: Microsoft.Management.Infrastructure.CimClass

Name                   MemberType     Definition
----                   ----------     ----------
Dispose                Method         void Dispose(), void IDisposable.Dispose()
Equals                 Method         bool Equals(System.Object obj)
GetHashCode            Method         int GetHashCode()
GetType                Method         type GetType()
ToString               Method         string ToString()
CimClassMethods        Property       Microsoft.Management.Infrastructure.Generic.CimReadOnlyKeyedCo...
CimClassProperties     Property       Microsoft.Management.Infrastructure.Generic.CimReadOnlyKeyedCo...
CimClassQualifiers     Property       Microsoft.Management.Infrastructure.Generic.CimReadOnlyKeyedCo...
CimSuperClass          Property       cimclass CimSuperClass {get;}
CimSuperClassName      Property       string CimSuperClassName {get;}
CimSystemProperties    Property       Microsoft.Management.Infrastructure.CimSystemProperties CimSys...
CimClassName           ScriptProperty System.String CimClassName {get=[OutputType([string])]...
```

Wszystkie metody zostały zgrupowane w kolekcji CimClassMethods a właściwości w kolekcji CimClassProperties

Kolekcja to taka właściwość której wartością jest lista

U nas to będzie lista nazw metod i lista nazw właćciwości które posiada obiekt klasy Win32_OperatingSystem

Aby wyświetlić informacje tylko o metodach posłuż się instrukcją

```
3  Get-CimClass -ClassName Win32_OperatingSystem | Select-Object CimClassMethods
```

```
CimClassMethods
---------------
{Reboot, Shutdown, Win32Shutdown, Win32ShutdownTracker...}
```

Odpowiedz nas nie zadowala, bo to informacja o to jakie metody wchodzą w skład kolekcji.

Kropki oznaczają, że jest tego więcej

Wybrana został a kolekcja a nie elementy kolekcji

Aby zobaczyć elementy kolekcji wpisz jak poniżej

```
4   Get-CimClass -ClassName Win32_OperatingSystem | Select-Object -ExpandProperty CimClassMethods
```

```
PS C:\WINDOWS\system32> Get-CimClass -ClassName Win32_OperatingSystem | Select-Object -ExpandProperty C

Name                ReturnType Parameters                            Qualifiers
----                ---------- ----------                            ----------
Reboot                  UInt32 {}                                    {Implemented, MappingString...
Shutdown                UInt32 {}                                    {Implemented, MappingString...
Win32Shutdown           UInt32 {Flags, Reserved}                     {Implemented, MappingString...
Win32ShutdownTracker    UInt32 {Comment, Flags, ReasonCode, Timeout} {Implemented, MappingString...
SetDateTime             UInt32 {LocalDateTime}                       {Implemented, Privileges, V...
```

-ExpandProperty - (parametr polecenia Select-Object) awansuje właściwość obiektu do rangi obiektu

Obiektem jest w wyniku powyższego kolekcja metod

Aby wyświetlić szczegółowych informacji tylko o metodzie Win32Shutdown klasy Win32_OperatingSystem za pomocą polecenia Get-CimClass wpisz w jednej linii jak poniżej

Get-CimClass -ClassName Win32_OperatingSystem | Select-Object -ExpandProperty CimClassMethods | Where Name -EQ Win32Shutdown | Select-Object -ExpandProperty Parameters

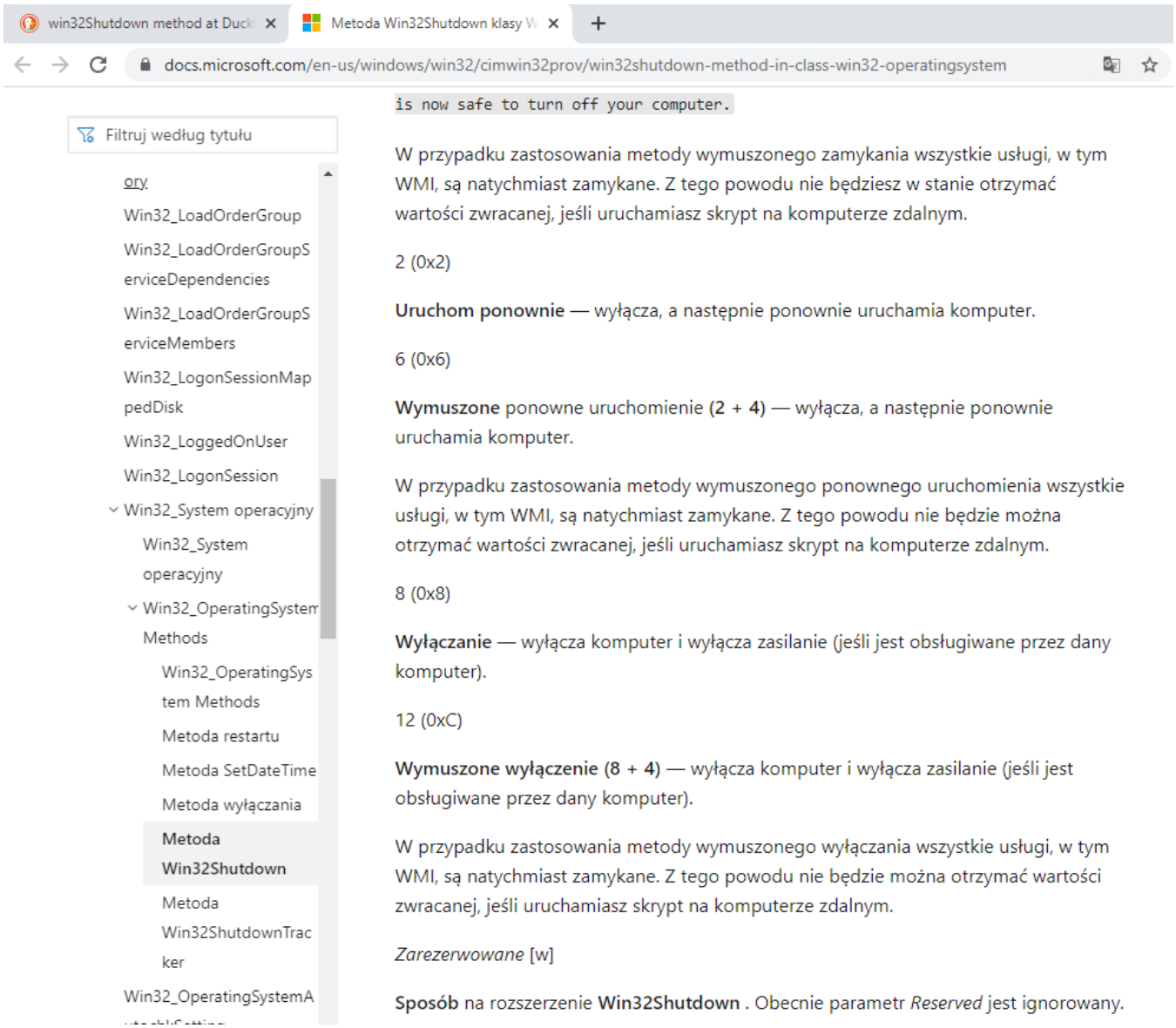
```
5   Get-CimClass -ClassName Win32_OperatingSystem | Select-Obj
```

```
PS C:\WINDOWS\system32> Get-CimClass -ClassName Win32_OperatingS

Name     CimType Qualifiers              ReferenceClassName
----     ------- ----------              ------------------
Flags     SInt32 {ID, in, MappingStrings}
Reserved  SInt32 {ID, in, MappingStrings}
```

Widzimy parametry Flags i Reserved oba typu Int32

Ale jakie wartości może przyjmować parametr Flags można to odnaleźć w helpie MS



Czas na dyski logiczne

```
6  Get-CimClass -ClassName Win32_LogicalDisk | Select-Object -ExpandProperty CimClassMethods
```

```
PS C:\WINDOWS\system32> Get-CimClass -ClassName Win32_LogicalDisk | Select-Object -ExpandProperty CimClassMethods

Name               ReturnType Parameters                                                          Qualifiers
----               ---------- ----------                                                          ----------
SetPowerState          UInt32 {PowerState, Time}                                                  {}
Reset                  UInt32 {}                                                                  {}
Chkdsk                 UInt32 {FixErrors, ForceDismount, OkToRunAtBootUp, RecoverBadSectors...} {Implemented, MappingStrings}
ScheduleAutoChk        UInt32 {LogicalDisk}                                                       {Implemented, MappingStrings, Static}
ExcludeFromAutochk     UInt32 {LogicalDisk}                                                       {Implemented, MappingStrings, Static}
```

A co znajduje się w klasie Win32_TimeZone

```
7  Get-CimClass -ClassName Win32_TimeZone | Select-Object -ExpandProperty CimClassMethods
```

```
PS C:\WINDOWS\system32> Get-CimClass -ClassName Win32_TimeZone | Select-Object -ExpandProperty CimClassMethods

PS C:\WINDOWS\system32>
```

Brak metod

Odwołując się do klasy Win32_Group możemy wyświetlić lokalne grupy

```
8   Get-WmiObject -Class Win32_Group
```

```
Caption
-------
DESKTOP-VNM5DM7\Administratorzy
DESKTOP-VNM5DM7\Administratorzy funkcji Hyper-V
DESKTOP-VNM5DM7\Czytelnicy dzienników zdarzeń
DESKTOP-VNM5DM7\Goście
DESKTOP-VNM5DM7\Grupa kont zarządzana przez syste
```

Wyświetlam informacje o grupie

Get-WmiObject -Class Win32_Group -Filter "Name='Goście'"

```
12   Get-WmiObject -Class Win32_Group -Filter "Name='Goście'"
```

```
Caption        Domain Name   SID
-------        ------ ----   ---
STACJA\Goście STACJA Goście S-1-5-32-546
```

Zmieńmy nazwę grupy

```
12   Get-WmiObject -Class Win32_Group -Filter "Name='Goście'" | Invoke-WmiMethod -Name Rename -Argument Goscie
```

```
__GENUS          : 2
__CLASS          : __PARAMETERS
__SUPERCLASS     :
__DYNASTY        : __PARAMETERS
__RELPATH        :
__PROPERTY_COUNT : 1
__DERIVATION     : {}
__SERVER         :
__NAMESPACE      :
__PATH           :
ReturnValue      : 0
PSComputerName   :
```

ReturnValue = 0 to oznacza, że polecenie powiodło się

```
11   Get-WmiObject -Class Win32_Group
```

```
DESKTOP-VNM5DM7\Czytelnicy dzienników zdarzeń
DESKTOP-VNM5DM7\Goscie
DESKTOP-VNM5DM7\Grupa kont zarządzana przez system
```

Podobnie z Cim

```
12   Get-CimInstance -ClassName Win32_Group
```

```
PS C:\WINDOWS\system32> Get-CimInstance -ClassNa

SID          Name
---          ----
S-1-5-32-544 Administratorzy
S-1-5-32-578 Administratorzy funkcji Hyper-V
S-1-5-32-573 Czytelnicy dzienników zdarzeń
S-1-5-32-546 Goscie
S-1-5-32-581 Grupa kont zarządzana przez system
```

Dodam filtr, aby wyświetlić grupę

```
13  Get-CimInstance -ClassName Win32_Group -Filter "Name='Goscie'"

SID          Name   Caption       Domain
---          ----   -------       ------
S-1-5-32-546 Goscie STACJA\Goscie STACJA
```

Zmieńmy nawę grupy wpisz w jednej linii jak poniżej

Get-CimInstance -ClassName Win32_Group -Filter "Name='Goscie'" | Invoke-CimMethod -MethodName Rename -Arguments @{"Name"="Goście"}

```
14  Get-CimInstance -ClassName Win32_Group -Filter "Name='Goscie'" | Invoke-CimMethod -MethodName Rename -Arguments @{"N

PS C:\WINDOWS\system32> Get-CimInstance -ClassName Win32_Group -Filter "Name='Goscie'" | Invoke-CimMethod -MethodName Rena

ReturnValue PSComputerName
----------- --------------
          0
```

Mamy 0 a więc nazwa zmieniona

Dodatkowo sprawdzamy

```
15  Get-WmiObject -Class Win32_Group

DESKTOP-VNM5DM7\Czytelnicy dzienników zdarzeń
DESKTOP-VNM5DM7\Goście
DESKTOP-VNM5DM7\Grupa kont zarządzana przez system
```

Kolejny przykład

Programy można uruchamiać korzystając z metody start.proces to skorzystamy z metody Win32_Process mówię o uruchomieniu metody klasy a nie obiektu, proces powstanie dopiero gdy program zostanie uruchomiony, a teraz go niema, przed uruchomieniem procesu mamy klasę Win32_Process a wśród jej metod metodę Create która utworzy nowy proces w tym przypadku dla klasy wywołujemy metodę Create która jako parametr CommandLine przyjmuje wartość notepad.exe

Invoke-CimMethod -Class Win32_Process -MethodName Create -Arguments @{"CommandLine"="notepad.exe"}

```
15  Invoke-CimMethod -Class Win32_Process -MethodName Create -Arguments @{"CommandLine"="notepad.exe"}

PS C:\Windows\system32> Invoke-CimMethod -Class Win32_Process -MethodName Create -Arguments @{CommandLine = 'notepad.exe'}


ProcessId ReturnValue PSComputerName
--------- ----------- --------------
             9
```

Sprawdzam, czy działa notepad

```
PS C:\Windows\system32> Get-Process -Name notepad
Get-Process : Cannot find a process with the name "notepad". Verify the process name and call the cmdlet again.
At line:1 char:1
+ Get-Process -Name notepad
+ ~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (notepad:String) [Get-Process], ProcessCommandException
    + FullyQualifiedErrorId : NoProcessFoundForGivenName,Microsoft.PowerShell.Commands.GetProcessCommand
```

Uruchom proces notpad.exe jeżeli nie działa

Start-Job -ScriptBlock { Start-Process notepad -WindowStyle Hidden } ;

```
PS C:\Windows\system32> Start-Job -ScriptBlock { Start-Process notepad -WindowStyle Hidden }

Id     Name            PSJobTypeName   State         HasMoreData     Location             Command
--     ----            -------------   -----         -----------     --------             -------
1      Job1            BackgroundJob   Running       True            localhost             Start-Process notepad...
```

Sprawdzam, czy działa notepad z Get-CimInstance

Get-CimInstance -ClassName Win32_Process -Filter 'Name="notepad.exe"'

```
PS C:\Windows\system32> Get-CimInstance -ClassName Win32_Process -Filter 'Name="notepad.exe"'

ProcessId Name        HandleCount WorkingSetSize VirtualSize
--------- ----        ----------- -------------- -----------
7556      Notepad.exe 512         46473216       4645212160
```

Wyłączenie notepad

```
19  Get-CimInstance -ClassName Win32_Process -Filter "Name='notepad.exe'" | Invoke-CimMethod -MethodName Terminate
```

```
ReturnValue PSComputerName
----------- --------------
          0
```

```
20  Get-Process -Name notepad
```

```
+ Get-Process -Name notepad
+ ~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (notepad:String) [Get-Process], ProcessCommandException
    + FullyQualifiedErrorId : NoProcessFoundForGivenName,Microsoft.PowerShell.Commands.GetProcessCommand
```

Uruchamiam notepad

Invoke-CimMethod -Class Win32_Process -MethodName Create -Arguments @{"CommandLine"="notepad.exe"}

```
PS C:\Windows\system32> Invoke-CimMethod -Class Win32_Process -MethodName Create -Arguments @{"CommandLine"="notepad.exe"}

ProcessId ReturnValue PSComputerName
--------- ----------- --------------
             9
```

Przesyłamy obiekty procesu do ForEach-Object a następnie dla każdego procesu wywołamy metode Terminate należy podać powód zamknięcia programu 0 – normalne zamknięcie procesu

```
22  Get-WmiObject -ClassName Win32_Process -Filter "Name='notepad.exe'" | ForEach-Object {$PSItem.Terminate(0)}
```

```
__GENUS          : 2
__CLASS          : __PARAMETERS
__SUPERCLASS     :
__DYNASTY        : __PARAMETERS
__RELPATH        :
__PROPERTY_COUNT : 1
__DERIVATION     : {}
__SERVER         :
__NAMESPACE      :
__PATH           :
ReturnValue      : 0
PSComputerName   :
```

ReturnValue : 0 – poprawne zakończenie

Sprawdzamy przez Get-Process -Name notepad

```
23   Get-Process -Name notepad
```

```
PS C:\WINDOWS\system32> Get-Process -Name notepad
Get-Process : Cannot find a process with the name "notepad". Verify the process name and call the cmdlet again.
At line:1 char:1
+ Get-Process -Name notepad
+ ~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (notepad:String) [Get-Process], ProcessCommandException
    + FullyQualifiedErrorId : NoProcessFoundForGivenName,Microsoft.PowerShell.Commands.GetProcessCommand
```

Brak procesu notatnika

Poznałaś sposoby pracy z obiektami WMI i CIM

Potrafisz Listować właściwości i metody tych obiektów

Umiesz odczytywać właściwości

Wiesz, jak uruchamiać metody obiektów oraz klas