WMI i CIM. Praca z klasami i obiektami

Pojęcia związane z zapytaniami WMI i CIM

Wyszukiwanie klasy WMI pozwalające na uzyskanie potrzebnej informacji

Podstawy budowy zapytań WMI

Zapytania języka WQL

Przegląd dokumentacji MSDN

WMI i CIM służą do wykonywania zapytań w systemach Windows.

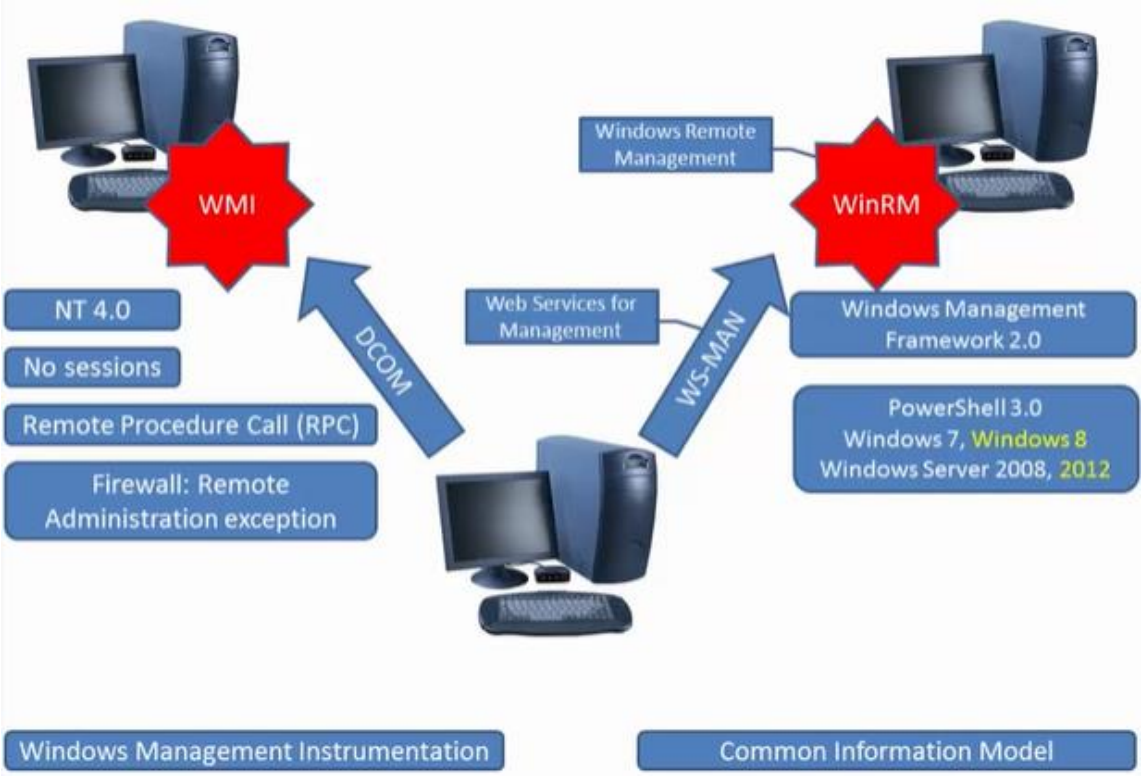
Zapytania mogą dotyczyć systemu i zainstalowanego oprogramowania.

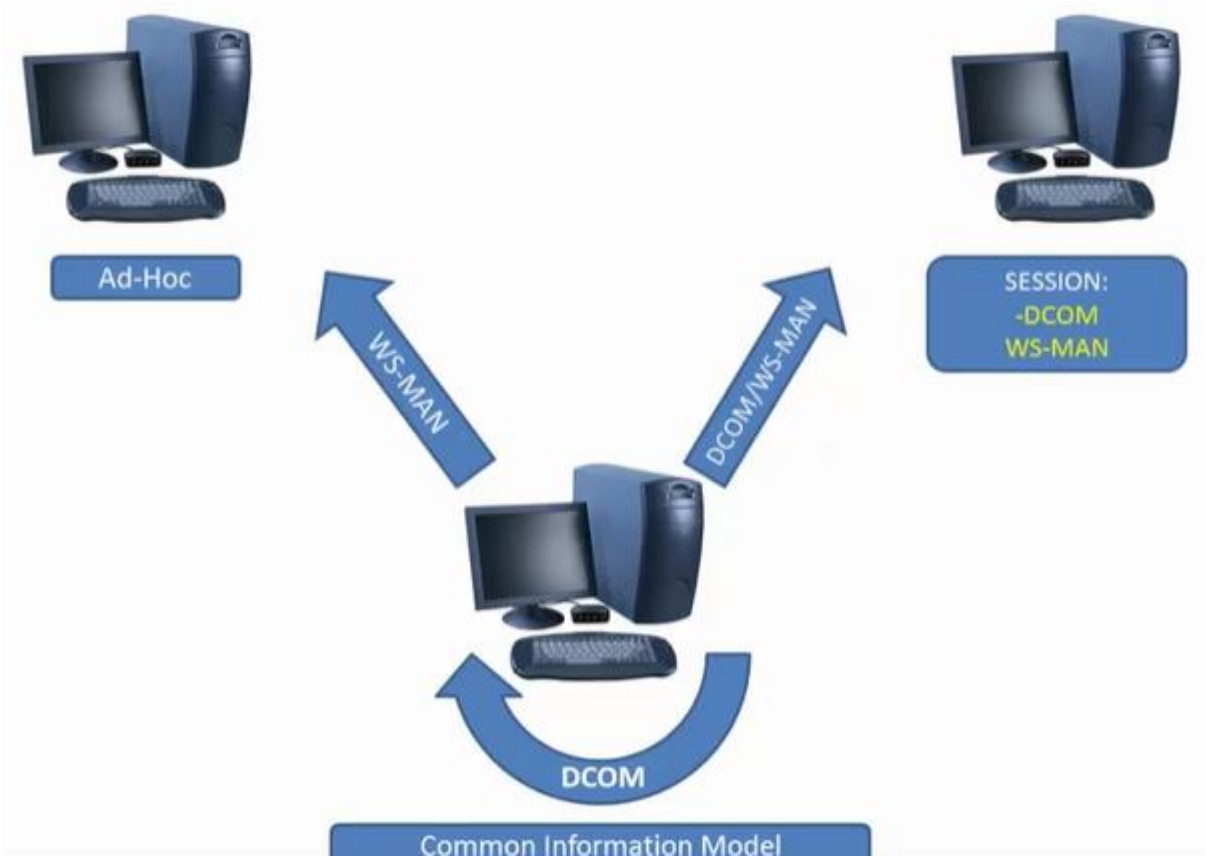
Można wysłać zapytania o dyski, procesy usługi drukarki, zapytania można wykonywać lokalnie i/lub zdalnie.

Wykonywanie zapytań na komputerach zdalnych możliwe jest za pomocą dwóch technik.

(WMI) - Windows Management Instrumentation - Instrumentacja zarządzania Windows to infrastruktura służąca do zarządzania danymi i operacjami w systemach operacyjnych opartych na systemie Windows. Możesz pisać skrypty lub aplikacje WMI, aby zautomatyzować zadania administracyjne na komputerach zdalnych, ale usługa WMI dostarcza również dane zarządzania do innych części systemu operacyjnego i produktów - na przykład System Center Operations Manager (dawniej Microsoft Operations Manager (MOM)) lub Zdalne Zarządzanie Windows (WinRM).

(CIM) - Common Information Model - obiektowy model informacji to standard określający zasady prezentowania i wymiany danych między aplikacjami zarządzającymi a zarządzanymi urządzeniami; pozwalający dostawcom oprogramowania tworzyć rozwiązania, jest standardem otwartym wieloplatformowy rozwijaniem tego standardu zajmuje się komitet normalizacyjny DMTF (Distributed Management Task Force).





Kiedy chcesz poznać szczegóły konfiguracji systemu z udziałem WMI musisz znaleźć właściwą klasę, która zawiera w sobie takie informacje

W ramach WMI mamy ogromną ilość klas i to jest właśnie największa trudność

```
1   Get-WmiObject -Namespace root -List -recurse
```

```
CIM_InstIndication                {}
CIM_InstCreation                  {}
```

Dlatego klasy zostały pogrupowane na przestrzenie nazw

```
2   Get-WmiObject -Namespace root\cimv2 -List -recurse
```

```
__NAMESPACE                       {}                  {Name}
```

Jest mniej, ale jeszcze zbyt dużo – to tylko jedna podprzestrzeń

```
3   Get-WmiObject -Namespace root\cimv2 -List -recurse | Select -Unique __NAMESPACE
```

```
PS C:\WINDOWS\system32> Get-WmiObject -Namespace root\cimv2 -List -recurse | Select -Unique __NAMESPACE

__NAMESPACE
-----------
ROOT\CIMV2
ROOT\cimv2
root\cimv2
ROOT\cimv2\mdm
ROOT\ci
ROOT\ci
ROOT\cimv2\TerminalServices
ROOT\cimv2\mdm\dmmap
ROOT\cimv2\Security\MicrosoftTpm
ROOT\cimv2\Security\MicrosoftVolumeEncryption
```

Click to add search criteria.

Szukamy konfiguracji pulpitu Windows

```
3   Get-WmiObject -Namespace root\cimv2 -List -Recurse | where { $_.Name -like "*desktop*" }
```

```
   NameSpace: ROOT\cimv2

Name                                Methods              Properties
----                                -------              ----------
CIM_DesktopMonitor                  {SetPowerState, R... {Availability, Bandwidth, Caption, ConfigManagerErrorCode...}
Win32_DesktopMonitor                {SetPowerState, R... {Availability, Bandwidth, Caption, ConfigManagerErrorCode...}
Win32_Desktop                       {}                   {BorderWidth, Caption, CoolSwitch, CursorBlinkRate...}
Win32_UserDesktop                   {}                   {Element, Setting}
Win32_SystemDesktop                 {}                   {Element, Setting}


   NameSpace: ROOT\cimv2\TerminalServices

Name                                Methods              Properties
----                                -------              ----------
Win32_TSRemoteDesktop               {}                   {Alias, Caption, Description, IconContents...}


   NameSpace: ROOT\cimv2\mdm\dmmap

Name                                Methods              Properties
----                                -------              ----------
MDM_Policy_Result01_RemoteDeskto... {}                   {AllowUsersToConnectRemotely, ClientConnectionEncryptionLevel,
MDM_Policy_Config01_RemoteDeskto... {}                   {AllowUsersToConnectRemotely, ClientConnectionEncryptionLevel,
MDM_Policy_User_Config01_RemoteD... {}                   {AutoSubscription, InstanceID, ParentID}
MDM_Policy_Result01_RemoteDesktop02 {}                   {InstanceID, LoadAadCredKeyFromProfile, ParentID}
MDM_Policy_User_Config01_Desktop02  {}                   {InstanceID, ParentID, PreventUserRedirectionOfProfileFolders}
MDM_Policy_User_Result01_Desktop02  {}                   {InstanceID, ParentID, PreventUserRedirectionOfProfileFolders}
MDM_Policy_Config01_RemoteDesktop02 {}                   {InstanceID, LoadAadCredKeyFromProfile, ParentID}
MDM_Policy_User_Result01_RemoteD... {}                   {AutoSubscription, InstanceID, ParentID}
```

Jeżeli wiemy czego szukamy to przydatne może być wylistowanie nazw klas z określonej przestrzeni nazw i alfabetyczne posortowanie tych nazw

```
5   Get-WmiObject -Namespace root\cimv2 -List | Sort Name
```

```
Win32_SystemTimeZone                {}                   {Element, Setting}
Win32_SystemTrace                   {}                   {SECURITY_DESCRIPTOR, TIME_CREATED}
Win32_SystemUsers                   {}                   {GroupComponent, PartComponent}
```

```
6   Get-CimClass -Namespace root\cimv2 | Sort CimClassName
```

```
Win32_SystemTimeZone                {}                   {Element, Setting}
Win32_SystemTrace                   {}                   {SECURITY_DESCRIPTOR, TIME_CREATED}
```

Polecenie listujące klasy przekazujemy potokiem do where-object które odnajduje wyłącznie klasy o nazwie desktop a następnie sortujemy je alfabetycznie według nazwy

```
3   Get-WmiObject -Namespace root\cimv2 | where { $_.CimClassName -like "*desktop*" } | Sort CimClassName
```

```
   NameSpace: ROOT\cimv2

Name                                Methods              Properties
----                                -------              ----------
CIM_DesktopMonitor                  {SetPowerState, R... {Availability, Bandwidth, Caption, ConfigManagerErrorCode...}
Win32_Desktop                       {}                   {BorderWidth, Caption, CoolSwitch, CursorBlinkRate...}
Win32_DesktopMonitor                {SetPowerState, R... {Availability, Bandwidth, Caption, ConfigManagerErrorCode...}
Win32_SystemDesktop                 {}                   {Element, Setting}
Win32_UserDesktop                   {}                   {Element, Setting}
```

Należy utworzyć obiekt, który pozwoli pracować z pulpitem lub dyskiem

```
8   Get-WmiObject -Class Win32_Desktop
9   Get-WmiObject -Class Win32_LogicalDisk
```

```
PS C:\WINDOWS\system32> Get-WmiObject -Class Win32_Desktop

Name                : ZARZĄDZANIE NT\SYSTEM
ScreenSaverActive   : False
ScreenSaverSecure   :
ScreenSaverTimeout  :
SettingID           :

Name                : ZARZĄDZANIE NT\USŁUGA LOKALNA
ScreenSaverActive   : False
ScreenSaverSecure   :
ScreenSaverTimeout  :
SettingID           :

Name                : ZARZĄDZANIE NT\USŁUGA SIECIOWA
ScreenSaverActive   : False
ScreenSaverSecure   :
ScreenSaverTimeout  :
SettingID           :

Name                : DESKTOP-VNM5DM7\admin
ScreenSaverActive   : False
ScreenSaverSecure   :
ScreenSaverTimeout  :
SettingID           :

Name                : .DEFAULT
ScreenSaverActive   : False
ScreenSaverSecure   :
ScreenSaverTimeout  :
SettingID           :
```

```
5   Get-WmiObject -Class Win32_LogicalDisk
```

```
DeviceID     : C:
DriveType    : 3
ProviderName :
FreeSpace    : 1064826626048
Size         : 1098571051008
VolumeName   :
```

Klsa – to zbiór właściwości, które posiadają opisywane przez tą klasę obiekty np. dysk logiczny

A nie dysk C lub D

Obiekt – to konkretna reprezentacja np. dysku C na komputerze

Obiekty cim

```
10  Get-CimInstance -Class Win32_Desktop
```

```
VolumeName   :

PS C:\WINDOWS\system32> Get-CimInstance -Class Win32_Desktop

SettingID Name                          ScreenSaverActive ScreenSaverSecure ScreenSaverTimeout
--------- ----                          ----------------- ----------------- ------------------
          ZARZĄDZANIE NT\SYSTEM         False
          ZARZĄDZANIE NT\USŁUGA LOKALNA False
          ZARZĄDZANIE NT\USŁUGA SIECIOWA False
          DESKTOP-VNM5DM7\admin         False
          .DEFAULT                      False
```

```
11  Get-CimInstance -Class Win32_LogicalDisk
12
```

```
          DESKTOP-VNM5DM7\admin          False
          .DEFAULT                       False

PS C:\WINDOWS\system32> Get-CimInstance -Class Win32_LogicalDisk

DeviceID DriveType ProviderName VolumeName Size          FreeSpace
-------- --------- ------------ ---------- ----          ---------
A:       2
C:       3                                1590611619840 1575861276672
D:       5
```

Click to add search criteria.

Do wyświetlania klas i obiektów WMI używamy tego samego polecenia Get-WmiObject

Kiedy szukamy klas CIM używamy Get-CimClass

Do pobrania obiektu CIM używamy Get-CimInstance

Aby wykonać zapytanie WMI

```
6   Get-WmiObject -Query "Select * from Win32_LogicalDisk"
```

```
DeviceID     : C:
DriveType    : 3
ProviderName :
FreeSpace    : 1064826142720
Size         : 1098571051008
VolumeName   :
```

```
13  Get-CimInstance -query "Select * from Win32_logicaldisk"
```

Click to add search criteria.

```
PS C:\WINDOWS\system32> Get-CimInstance -query "Select * from Win32_logicaldisk"

DeviceID DriveType ProviderName VolumeName Size          FreeSpace
-------- --------- ------------ ---------- ----          ---------
A:       2
C:       3                                1590611619840 1575860781056
D:       5
```

Uruchomię dwa procesy notpad.exe

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

Teraz przefiltrujemy obiekty, aby wyświetlić procesy notpad.exe

Get-WmiObject -Class Win32_Process -Filter "Name = 'notepad.exe'"

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

```
7  Get-WmiObject -Class Win32_Process -Filter "Name = 'notepad.exe'"
8  Get-CimInstance -Class  Win32_Process -Filter "Name = 'notepad.exe'"
```

```
WriteOperationCount        : 6
WriteTransferCount         : 71
PSComputerName             : STACJA
ProcessName                : Notepad.exe
Handles                    : 602
VM                         : 2203726438400
WS                         : 57507840
Path                       : C:\Program Files\WindowsApps\Microsoft.WindowsNotepad_11.2405.13.0_x64__8wekyb3d8bbwe\Notepad\Notepad.exe



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

ProcessId Name        HandleCount WorkingSetSize VirtualSize
--------- ----        ----------- -------------- -----------
6612      Notepad.exe 590         57569280       2203711631360
4080      Notepad.exe 199         12730368       2203459207168
4300      Notepad.exe 589         57462784       2203720216576
```

Jak filtrować obiekty, gdy korzystamy z WQL (Windows Management Instrumentation Query Language) to język zapytań używany do wykonywania zapytań na obiektach zarządzania Windows (WMI). Jest podobny do SQL, ale jest specyficzny dla WMI. WQL pozwala na pobieranie informacji o różnych aspektach systemu operacyjnego, takich jak procesy, usługi, ustawienia konfiguracyjne, itd. Aby filtrować obiekty za pomocą WQL, używasz klauzuli WHERE w zapytaniu, podobnie jak w SQL.

Get-WmiObject -Query "SELECT * FROM Win32_Process WHERE Name = 'notepad.exe'"

Get-CimInstance -Query "SELECT * FROM Win32_Process WHERE Name = 'notepad.exe'"

```
9  Get-WmiObject -Query "SELECT * FROM Win32_Process WHERE Name = 'notepad.exe'"
10 Get-CimInstance -Query "SELECT * FROM Win32_Process WHERE Name = 'notepad.exe'"
```

```
WriteOperationCount        : 6
WriteTransferCount         : 71
PSComputerName             : STACJA
ProcessName                : Notepad.exe
Handles                    : 583
VM                         : 2203719168000
WS                         : 57659392
Path                       : C:\Program Files\WindowsApps\Microsoft.WindowsNotepad_11.2405.13.0_x64__8wekyb3d8bbwe\Notepad\Notepad.exe



PS C:\Windows\system32> Get-CimInstance -Query "SELECT * FROM Win32_Process WHERE Name = 'notepad.exe'"

ProcessId Name        HandleCount WorkingSetSize VirtualSize
--------- ----        ----------- -------------- -----------
6612      Notepad.exe 580         57778176       2203709534208
4080      Notepad.exe 199         12779520       2203459207168
4300      Notepad.exe 579         57643008       2203718119424
```

Podsumowanie

Mamy dwa rodzaje poleceń

Get-WmiObject

Get-CimClass lub Get-CimInstance

Główne wyzwanie to odnalezienie właściwej klasy i filtrowanie wyszukanej nazwy w potoku.

Administratorzy najczęściej szukają danej klasy z udziałem wyszukiwarki internetowej.

Windows PowerShell Scriptomatic to narzędzie, które pomaga w tworzeniu skryptów Windows PowerShell wykorzystujących WMI (Windows Instrumentation Management) do zarządzania systemem i administrowania. Podobnie jak oryginalny Scriptomatic, Windows PowerShell Scriptomatic wyświetla listę klas WMI i umożliwia wybór klasy oraz tworzenie skryptu Windows PowerShell, który pobierze właściwości z danej klasy. Jeśli chcesz dowiedzieć się więcej, polecam zajrzeć na stronę Microsoft Community Hub lub Hey, Scripting Guy!.

Strona microsoft.com o np -ClassName Win32_Desktop

Przykład zbierania informacji o pulpitach: Aby wyświetlić informacje o komputerach stacjonarnych na komputerze lokalnym, użyj polecenia: Get-CimInstance -ClassName Win32_Desktop

Poznałeś pojęcia związane z zapytaniami WMI i CIM

Wiesz, jak wyszukiwać odpowiednie klasy WMI

Znasz podstawy budowy zapytań WMI

Wiesz, co to zapytania języka WQL

Widziałeś' przykładową dokumentację klas WMI na stronach MSDN